Ryan Young
817447547
CompE375 – Section 2

Final Project: CANBUS Demo

This project communicates between two microcontrollers using the canbus protocol and two canbus breakout boards. From a serial terminal on a pc it can receive a byte over UART, then it sends that bite over SPI to the MCP2515 canbus transceiver, which in turn sends it to the other canbus breakout board. The second microcontroller then reads the data over SPI and transmits it over UART to another serial terminal. This process works exactly the same in the opposite direction.

I began the project by establishing direct SPI communication between my Xplained-mini and an Arduino nano while I waited for my canbus devboards to arrive. I had to research the timing of SPI quite a bit, and still struggled with the slow speed of UART compared to SPI, which would cause random lost data packets. This ended up being more related to microcontrollers being poor SPI slaves, and I not timing trouble once connected to the MCP2515.

Once the devboards arrived I then began the process of interpreting the library for the MCP2515 canbus transceiver chip and trying to establish communication just between the mcu and the MCP2515. At this point my Xplained-mini stopped being able to be programmed. I fortunately had just acquired an AVRISP mkII ISP programmer, and was able to erase one my Arduino Nanos and program it "baremetal" since the AVRISP mkII works with Atmel Studio. From there I was able to get the MCP2515 to initialize and read back the contents of its registers. My next process was to setup the IDs and filters for the MCP2515 so I would be able to address the correct receiver. Once my IDs were established I had to build the data frame required for canbus. The final part was to send out my UART data over canbus and setup my receive interrupt to read the canbus receive buffer. I burned the firmware onto two Nanos and was able to establish communication fairly quickly.

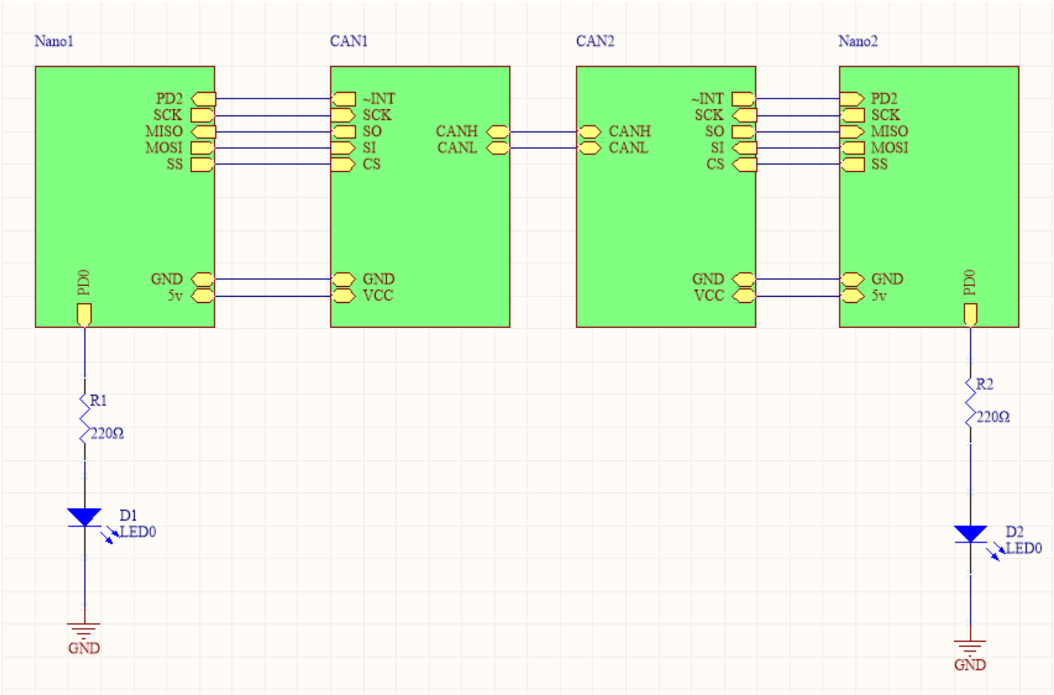
The project currently only uses a single ID, but this will be expanded to use a block of IDs and filters so there can be multiple devices on the bus as well as the ID's being connected to specific sensor data on each device and establishing an ID priority chain. This project is a prototype for use in this year's Mechatronics autonomous submarine. We plan to have 3 daughter cards on a backplane motherboard that all communicate over canbus. The motherboard then passes data and commands in between the daughter cards and the main CPU over USB. This project taught me a lot about various communication protocols, as well as building a data frame, which I will need to determine how we wish to communicate between the backplane and main CPU of the submarine.

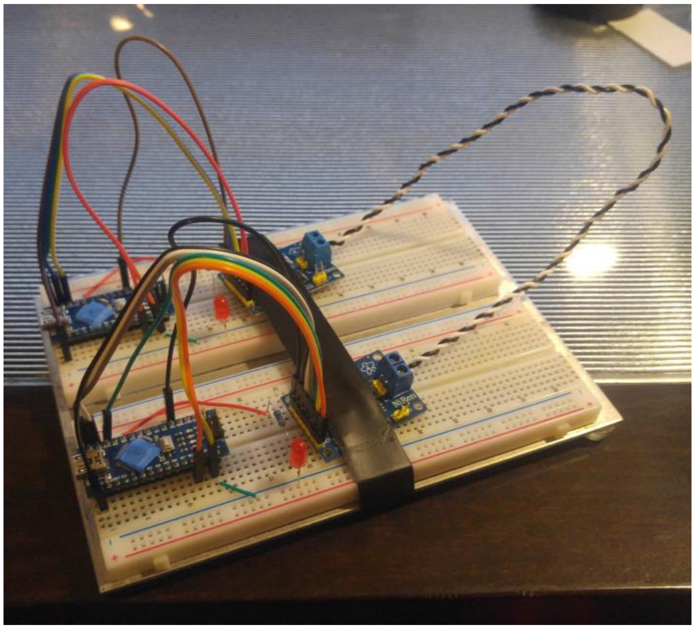
Pseudocode:

```
 //initialization

initialize GPIO pins
        set input and output pins for MCP2515 interrupt and status LED.
        Can’t use the board’s LED as it’s attached to the SCK port.

enable interrupts
        external interrupt needed for when MCP2515 has data in receive buffer

initialize UART
        set speed and character size

initialize SPI
        setup input and output pins and pull up/down resistors
        enable SPI
        set clock speed of SCK

initialize the MCP2515
        //adjust registers of the transceiver over SPI
        Put into configuration mode
                set speed
                enable interrupts
                set input/output pins
        Read and verify value in register
                return status over serial and turn on error LED if read fails.
        Setup ID values and filters
Setup transmit frame
        put together the basic transmit word for this demo

main while loop
{
        wait for serial to be available
                if available
                        store it
                        transmit the data over can
}

External interrupt INT0 ISR
        when the mcp2515 has received data in its buffer it brings the
                INT pin low.  This triggers an interrupt on the mcu.
        when the interrupt it triggered get the data out of the buffer
        transmit the data over UART.
```

Operation Instructions:

Using an atmega328p based microcontroller, an MCP2515 based canbus dev board, two LEDs and two 220Ω resistors connect the following circuit.





Using either the mcu's own debugger or an external ISP programmer flash the mcu's with the firmware, making sure to switch the TxID and RxID so that they're not trying to transmit and receive on the same ID. Then open up two serial terminals with your preferred application (I used PuTTy) and connect them to the two mcu's respectively.

Upon startup, the mcu's will attempt to initialize the canbus. If they are successful they will print "canbus Init SUCCESS!" on the terminal. If they fail, they will print "canbus Init FAIL!" and light the status LED hooked to PB2. Once successful, typing into one terminal will transmit and display the text in the other terminal and vice versa.

Estimated time: 30hrs

Demo video link:

https://youtu.be/rwBOggfKPX4

Resources used:

- CHENBO Canbus devboard:
    - http://a.co/dAPAGak
    - Although any MCP2515 based design will work.
- MCP2515 Datasheet:
    - http://ww1.microchip.com/downloads/en/DeviceDoc/20001801H.pdf
- Example Arduino Library from the SPARKFUN canbus devboard:
- https://www.sparkfun.com/products/13262
- Microchip's sample PIC to canbus node write up:
    - http://ww1.microchip.com/downloads/en/AppNotes/00215c.pdf
- A very clear breakdown of how the MCP2515 IDs, filters, and masks work that I found on the microchip forums:
    - http://www.microchip.com/forums/FindPost/172790
- Arduino nano pinout diagram:
    - http://www.pighixxx.com/test/pinouts/boards/nano.pdf
- This website's well written and visualized explanation of the SPI protocol:
    - http://maxembedded.com/2013/11/serial-peripheral-interface-spi-basics/
- And this website's well written AVR SPI code examples:
    - http://www.gammon.com.au/spi
- A very good explanation of the canbus data frame and how the canbus operates
    - https://youtu.be/RRbrk3SdSKA

Target Device:

The Xplained-mini or any ATMEGA328P microcontroller.

Tools used:

- Atmel Studio 7
- Notepad++
- AVRISP mkII http://www.atmel.com/tools/AVRISPMKII.aspx

```
/******************************************************************************
       can_bus.c

       Description:
              Communicate between two microcontrollers over CAN-BUS protocol using
              the MCP2515 CAN transceiver chip.

       Created:       12/6/2016 8:08:43 AM
       Author:        Ryan Young
       RedID:         817447547
*****************************************************************************///template for
ATmega328p
#define F_CPU 16000000UL // 16MHz clock from the debug processor
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <string.h>


/******************************************************************************
       CANBUS ID definition|
              change values for different devices so that M can talk to S and
              visa versa
******************************************************************************/
//RxID is your device ID that you allow messages to receive
//uint8_t RxID = 0x10;  //M
uint8_t RxID = 0x20;  //S

//TxID is the target ID you're transmitting to
//uint8_t TxID = 0x20;       //M
uint8_t TxID = 0x10;  //S
/******************************************************************************/


#include "headers/global.h"                //general define header pulled from net
#include "headers/defines.h"        //Pin name definitions
#include "headers/functions.h"             //general functions
#include "headers/spi_ry.h"                //SPI protocol implementation
#include "headers/usart_ry.h"       //serial communication with PC
#include "headers/mcp2515_ry_def.h" //MCP2515 register and bit definitions
#include "headers/mcp2515_ry.h"            //MCP2515 functions

tCAN usart_char;      //transmit package
tCAN spi_char;        //receive package

/******************************************************************************
       start of main()|
******************************************************************************/
int main(void)
{
              //initialization functions
       GPIO_init();
       INTERRUPT_init();
       USART_Init(103);//103 sets baud rate at 9600
       SPI_masterInit();

       //MCP2515 initialization
       if(mcp2515_init(CANSPEED_500))
       {
              USART_Transmit_TX("Can Init SUCCESS!");
       }else
```

```
    {
        USART_Transmit_TX("Can Init FAILURE!");
    }
    USART_Transmit(10);//New Line
    USART_Transmit(13);//Carriage return

    //setup the transmit frame
    usart_char.id = TxID;               //set target device ID
    usart_char.header.rtr = 0;          //no remote transmit(i.e. request info)
    usart_char.header.length = 1;//single byte(could be up to 8)

    while (1)
    {
        if(!(UCSR0A & (1<<RXC0)))//if data in serial buffer
        {
            //get serial data
            usart_char.data[0] = USART_Receive();

            //transmit usart_char over canbus
            mcp2515_send_message(&usart_char);
        }
    }
}/*****end of main()**********************************************************/

/******************************************************************************
    RECEIVE interrupt on pin PD2|
******************************************************************************/
ISR(INT0_vect)
{
    mcp2515_get_message(&spi_char);//get canbus message
    USART_Transmit(spi_char.data[0]); //transmit message over uart
}
```

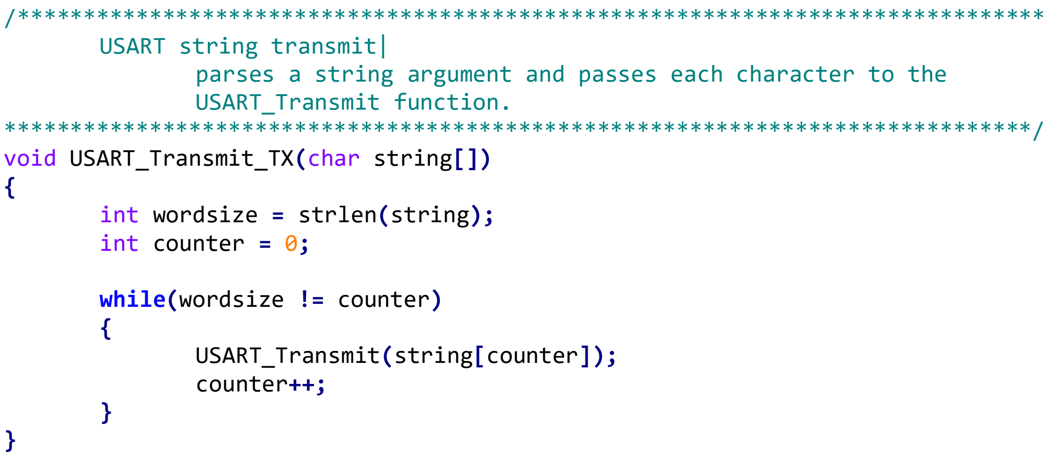
```
/*******************************************************************************
RYAN YOUNG
RyanAYoung81@gmail.com

       This is a simple define header I found floating around the internet of
               useful GPIO commands and bitwise operations that I've found useful
               for making my code more legible.

*******************************************************************************/

#ifndef GLOBAL_H
#define GLOBAL_H

// ----------------------------------------------------------------------------

#define true    1
#define false   0

#define True    1
#define False   0

//typedef      _Bool bool;
//typedef      boolean Bool;

// ----------------------------------------------------------------------------

#define SET_L(x)              _XRS(x)
#define SET_H(x)              _XS(x)
#define TOGGLE(x)             _XT(x)
#define SET_OUTPUT(x)  _XSO(x)
#define SET_INPUT(x)   _XSI(x)
#define IS_SET(x)             _XR(x)

#define PORT(x)               _port2(x)
#define DDR(x)                _ddr2(x)
#define PIN(x)                _pin2(x)

#define _XRS(x,y)      PORT(x) &= ~(1<<y)
#define _XS(x,y)       PORT(x) |= (1<<y)
#define _XT(x,y)       PIN(x) |= (1<<y)

#define _XSO(x,y)      DDR(x) |= (1<<y)
#define _XSI(x,y)      DDR(x) &= ~(1<<y)

#define _XR(x,y)       ((PIN(x) & (1<<y)) != 0)

#define _port2(x)      PORT ## x
#define _ddr2(x)       DDR ## x
#define _pin2(x)       PIN ## x

#endif // GLOBAL_H
```

```
/******************************************************************************
       defines.h

       Description:

       Created:       12/3/2016 3:11:57 PM
       Author:        Ryan Young
       RedID:         817447547

******************************************************************************/
//gpio pins formatted for using global.h
#define SS B,2
#define MOSI B,3
#define SCK B,5
#define INT D,2

//status light
#define LED2                 B,0

#define RxIDLow       ((RxID & 0x07) << 5)
#define RxIDHi (uint8_t)(RxID >> 3)
```

```
/*****************************************************************************
       functions.h

       Created:      12/3/2016 2:19:19 PM
       Author:       Ryan Young
       RedID:        817447547

*****************************************************************************/

/*****************************************************************************
       GPIO initiation|
              enabling inputs, outputs, and pull-up resistors
*****************************************************************************/
void GPIO_init(void)
{
       //SPI GPIO set in spi_ry.h

       //set input for INT line PD2
       SET_INPUT(INT);

       //set output for status LED on PB0
       SET_OUTPUT(LED2);
}

/*****************************************************************************
       interrupt initiation|
*****************************************************************************/
void INTERRUPT_init(void)
{
       //enable external interrupt for INT line from mcp2515
       EIMSK |= (1<<INT0);//enable
       EICRA &= ~(3<<ISC00);//low level interrupt for INT0

       sei(); //global interrupt enable
}
```

```
/******************************************************************************
       spi_ry.h

       Created:      12/3/2016 3:54:03 PM
       Author:       Ryan Young
       RedID:        817447547

******************************************************************************/
void SPI_masterInit(void)
{
       //set SS, MOSI, & SCK OUTPUT
       SET_OUTPUT(SS);
       SET_OUTPUT(MOSI);
       SET_OUTPUT(SCK);

       //SS high
       SET_H(SS);
       // MOSI & SCK low
       SET_L(MOSI);
       SET_L(SCK);

       //enable SPI
       SPCR |= (1<<SPE) | (1<<MSTR);

       //set SCK divider to f_osc/8
       SPCR |= 0b11; //sets divider to f_osc/16
       //SPSR |= 1; //f_osc * 2, results in f_osc/8
}

char SPI_txrx(char val)
{
       SPDR = val; //send value to buffer
       while(!(SPSR & (1<<SPIF))); //wait until complete
       _delay_us(50);
       return(SPDR);  //return received value
}
```

```
/******************************************************************************
       usart_ry.h

       Created:      12/3/2016 3:36:01 PM
       Author:       Ryan Young
       RedID:        817447547

******************************************************************************/

/******************************************************************************
       USART initialization|
******************************************************************************/
void USART_Init( unsigned int ubrr)
{
       /*Set baud rate */
       UBRR0H = (unsigned char)(ubrr>>8);
       UBRR0L = (unsigned char)ubrr;

       UCSR0B |= (1<<RXEN0)|(1<<TXEN0);
              /*Enable receiver and transmitter */

       UCSR0C |= (3<<UCSZ00);
              //(3<<UCSZ00) shifts 0b11 left into the UCSZ[1:0] position
              //     to enable an 8-bit character size
}

/******************************************************************************
       USART receive function|
              currently not used
******************************************************************************/
uint8_t USART_Receive( void )
{
       /* Wait for data to be received */
       while (!(UCSR0A & (1<<RXC0)));
       /* Get and return received data from buffer */
       return UDR0;
}

/******************************************************************************
       USART transmit function|
              transmits a character across the uart tx/rx pins
******************************************************************************/
void USART_Transmit( uint8_t data )
{
       /* Wait for empty transmit buffer */
       while ( !( UCSR0A & (1<<UDRE0)) );
       /* Put data into buffer, sends the data */
       UDR0 = data;
}
```



```
/*****************************************************************************
        USART string transmit|
                parses a string argument and passes each character to the
                USART_Transmit function.
*****************************************************************************/
void USART_Transmit_TX(char string[])
{
        int wordsize = strlen(string);
        int counter = 0;

        while(wordsize != counter)
        {
                USART_Transmit(string[counter]);
                counter++;
        }
}
```

```
// ----------------------------------------------------------------------------
/*
 * Copyright (c) 2007 Fabian Greif, Roboterclub Aachen e.V.
 *  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * $Id: mcp2515_defs.h 6611 2008-07-14 09:58:16Z fabian $
 */
// ----------------------------------------------------------------------------

/******************************************************************************
RYAN YOUNG
RyanAYoung81@gmail.com

       This header file was acquired from the SPARKFUN can-bus Arduino library.
       It has been modified for clarity and use in my own code

       I've also rewritten all comments in English(as prior was in German)
              and added details.

       location of SPARKFUN library:
       https://github.com/sparkfun/SparkFun_CAN-Bus_Arduino_Library
******************************************************************************/

#define CANSPEED_125  7              // CAN speed at 125 kbps
#define CANSPEED_250         3              // CAN speed at 250 kbps
#define CANSPEED_500  1              // CAN speed at 500 kbps

//from the MCP2515 DATA SHEET SPI instruction set
//TABLE 12-1
#define CAN_RESET                   0xC0
#define CAN_READ                    0x03
#define CAN_READ_RX_BUFF     0x90
#define CAN_WRITE                   0x02
#define CAN_LOAD_TX_BUFF     0x40
#define CAN_RTS                     0x80
#define CAN_READ_STATUS             0xA0
#define CAN_RX_STATUS        0xB0
#define CAN_BIT_MODIFY       0x05
/******************************************************************************
FORMAT:
```

```
send three words
       1)     COMMAND(from this list)
       2)     REGISTER ADDRESS
       3)     VALUE TO BE STORED IN REGIESTER

******************************************************************************/


/******************************************************************************
       Register addresses for the MCP2515|
              Call these address over SPI then write your values
******************************************************************************/
#define RXF0SIDH      0x00
#define RXF0SIDL      0x01
#define RXF0EID8      0x02
#define RXF0EID0      0x03
#define RXF1SIDH      0x04
#define RXF1SIDL      0x05
#define RXF1EID8      0x06
#define RXF1EID0      0x07
#define RXF2SIDH      0x08
#define RXF2SIDL      0x09
#define RXF2EID8      0x0A
#define RXF2EID0      0x0B
#define BFPCTRL              0x0C
#define TXRTSCTRL     0x0D
#define CANSTAT              0x0E
#define CANCTRL              0x0F

#define RXF3SIDH      0x10
#define RXF3SIDL      0x11
#define RXF3EID8      0x12
#define RXF3EID0      0x13
#define RXF4SIDH      0x14
#define RXF4SIDL      0x15
#define RXF4EID8      0x16
#define RXF4EID0      0x17
#define RXF5SIDH      0x18
#define RXF5SIDL      0x19
#define RXF5EID8      0x1A
#define RXF5EID0      0x1B
#define TEC                  0x1C
#define REC         0x1D

#define RXM0SIDH      0x20
#define RXM0SIDL      0x21
#define RXM0EID8      0x22
#define RXM0EID0      0x23
#define RXM1SIDH      0x24
#define RXM1SIDL      0x25
#define RXM1EID8      0x26
#define RXM1EID0      0x27
#define CNF3          0x28
#define CNF2          0x29
#define CNF1          0x2A
#define CANINTE              0x2B
#define CANINTF              0x2C
#define EFLG          0x2D

#define TXB0CTRL      0x30
#define TXB0SIDH      0x31
```

```
#define TXB0SIDL      0x32
#define TXB0EID8      0x33
#define TXB0EID0      0x34
#define TXB0DLC              0x35
#define TXB0D0        0x36
#define TXB0D1        0x37
#define TXB0D2        0x38
#define TXB0D3        0x39
#define TXB0D4        0x3A
#define TXB0D5        0x3B
#define TXB0D6        0x3C
#define TXB0D7        0x3D

#define TXB1CTRL      0x40
#define TXB1SIDH      0x41
#define TXB1SIDL      0x42
#define TXB1EID8      0x43
#define TXB1EID0      0x44
#define TXB1DLC              0x45
#define TXB1D0        0x46
#define TXB1D1        0x47
#define TXB1D2        0x48
#define TXB1D3        0x49
#define TXB1D4        0x4A
#define TXB1D5        0x4B
#define TXB1D6        0x4C
#define TXB1D7        0x4D

#define TXB2CTRL      0x50
#define TXB2SIDH      0x51
#define TXB2SIDL      0x52
#define TXB2EID8      0x53
#define TXB2EID0      0x54
#define TXB2DLC              0x55
#define TXB2D0        0x56
#define TXB2D1        0x57
#define TXB2D2        0x58
#define TXB2D3        0x59
#define TXB2D4        0x5A
#define TXB2D5        0x5B
#define TXB2D6        0x5C
#define TXB2D7        0x5D

#define RXB0CTRL      0x60
#define RXB0SIDH      0x61
#define RXB0SIDL      0x62
#define RXB0EID8      0x63
#define RXB0EID0      0x64
#define RXB0DLC              0x65
#define RXB0D0        0x66
#define RXB0D1        0x67
#define RXB0D2        0x68
#define RXB0D3        0x69
#define RXB0D4        0x6A
#define RXB0D5        0x6B
#define RXB0D6        0x6C
#define RXB0D7        0x6D

#define RXB1CTRL      0x70
#define RXB1SIDH      0x71
#define RXB1SIDL      0x72
```

```
#define RXB1EID8      0x73
#define RXB1EID0      0x74
#define RXB1DLC              0x75
#define RXB1D0        0x76
#define RXB1D1        0x77
#define RXB1D2        0x78
#define RXB1D3        0x79
#define RXB1D4        0x7A
#define RXB1D5        0x7B
#define RXB1D6        0x7C
#define RXB1D7        0x7D
//end of address names
/******************************************************************************/

/******************************************************************************
       Bit Definition for registers|
******************************************************************************/

/******************************************************************************
       BFPCTRL|
              RXnBF PIN CONTROL AND STATUS
******************************************************************************/
#define B1BFS         5
#define B0BFS         4
#define B1BFE         3
#define B0BFE         2
#define B1BFM         1
#define B0BFM         0

/******************************************************************************
       TXRTSCTRL|
              TXnRTS PIN CONTROL AND STATUS REGISTER
******************************************************************************/
#define B2RTS         5
#define B1RTS         4
#define B0RTS         3
#define B2RTSM        2
#define B1RTSM        1
#define B0RTSM        0

/******************************************************************************
       CANSTAT|

******************************************************************************/
#define OPMOD2        7
#define OPMOD1        6
#define OPMOD0        5
#define ICOD2         3
#define ICOD1         2
#define ICOD0         1

/******************************************************************************
       CANCTRL|

******************************************************************************/
#define REQOP2        7
#define REQOP1        6
#define REQOP0        5
#define ABAT          4
#define CLKEN         2
#define CLKPRE1              1
```

```
#define CLKPRE0              0

/******************************************************************************
       CNF3|

******************************************************************************/
#define WAKFIL        6
#define PHSEG22              2
#define PHSEG21              1
#define PHSEG20              0

/******************************************************************************
       CNF2|

******************************************************************************/
#define BTLMODE              7
#define SAM                  6
#define PHSEG12              5
#define PHSEG11              4
#define PHSEG10              3
#define PHSEG2        2
#define PHSEG1        1
#define PHSEG0        0

/******************************************************************************
       CNF1|

******************************************************************************/
#define SJW1          7
#define SJW0          6
#define BRP5          5
#define BRP4          4
#define BRP3          3
#define BRP2          2
#define BRP1          1
#define BRP0          0

/******************************************************************************
       CANINTE|

******************************************************************************/
#define MERRE         7
#define WAKIE         6
#define ERRIE         5
#define TX2IE         4
#define TX1IE         3
#define TX0IE         2
#define RX1IE         1
#define RX0IE         0
```

```
/*******************************************************************************
       CANINTF|

*******************************************************************************/
#define MERRF        7
#define WAKIF        6
#define ERRIF        5
#define TX2IF        4
#define TX1IF        3
#define TX0IF        2
#define RX1IF        1
#define RX0IF        0


/*******************************************************************************
       EFLG|

*******************************************************************************/
#define RX1OVR       7
#define RX0OVR       6
#define TXB0         5
#define TXEP         4
#define RXEP         3
#define TXWAR        2
#define RXWAR        1
#define EWARN        0

/*******************************************************************************
       TXBnCTRL|
               (n = 0, 1, 2)
*******************************************************************************/
#define ABTF         6
#define MLOA         5
#define TXERR        4
#define TXREQ        3
#define TXP1         1
#define TXP0         0

/*******************************************************************************
       RXB0CTRL|
               This and RXB1CTRL are the receive buffer control register use to
               control message      masks and filters.
*******************************************************************************/
#define RXM1         6
#define RXM0         5
#define RXRTR        3
#define BUKT         2
#define BUKT1        1
#define FILHIT0             0

/*******************************************************************************
       TXBnSIDL|
               (n = 0, 1)
*******************************************************************************/
#define EXIDE        3
```

```
/*******************************************************************************
       RXB1CTRL|
              Uses some of the same defines from RXB0CTRL
              RXM1, RXM0, RXRTR, and FILHIT0
*******************************************************************************/
#define FILHIT2              2
#define FILHIT1              1


/*******************************************************************************
       RXBnSIDL|
              (n = 0, 1)
*******************************************************************************/
#define SRR                  4
#define IDE                  3


/*******************************************************************************
       RXBnDLC|
       TXBnDLC|
              (n = 0, 1)
              same bit names for both registers
*******************************************************************************/
/**
 * \brief      Bitdefinition von RXBnDLC (n = 0, 1)
 * \see           (gleiche Bits)
 */
#define RTR                  6
#define DLC3          3
#define DLC2          2
#define DLC1          1
#define DLC0          0
```

```c
/* Copyright (c) 2007 Fabian Greif
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS "AS IS" AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 */
// ----------------------------------------------------------------------------

/******************************************************************************
RYAN YOUNG
       This header file was acquired from the SPARKFUN can-bus arduino library.
       It has been HEAVILY modified for clarity and use in my own code,
       as well as most of the original comments were in German.

       location of SPARKFUN library:
       https://github.com/sparkfun/SparkFun_CAN-Bus_Arduino_Library
******************************************************************************/


/******************************************************************************
       tCAN|
              this is a structure to hold a canbus message frame.
              It contains:
                     the message ID
                     if it's a "remote transmit receive" frame
                     the length: from 1 to 8 bytes
                     the 8 bytes
******************************************************************************/
typedef struct
{
       uint16_t id;
       struct {
              int8_t rtr : 1;
              uint8_t length : 4;
       } header;
       uint8_t data[8];
} tCAN;
```

```
void mcp2515_write_register( uint8_t adress, uint8_t data )
{
        SET_L(SS);//enable slave

        SPI_txrx(CAN_WRITE);//send write instruction
        SPI_txrx(adress);//send address
        SPI_txrx(data);//send value

        SET_H(SS);//disable slave
}

uint8_t mcp2515_read_register(uint8_t adress)
{
        uint8_t data;

        SET_L(SS);//enable slave

        SPI_txrx(CAN_READ);
        SPI_txrx(adress);

        data = SPI_txrx(0xff);

        SET_H(SS);//disable slave

        return data;
}

// -------------------------------------------------------------------------
void mcp2515_bit_modify(uint8_t adress, uint8_t mask, uint8_t data)
{
        SET_L(SS);

        SPI_txrx(CAN_BIT_MODIFY);
        SPI_txrx(adress);
        SPI_txrx(mask);
        SPI_txrx(data);

        SET_H(SS);
}

// ----------------------------------------------------------------------------
uint8_t mcp2515_read_status(uint8_t type)
{
        uint8_t data;

        SET_L(SS);

        SPI_txrx(type);
        data = SPI_txrx(0xff);

        SET_H(SS);

        return data;
}
```

```
/*****************************************************************************
        MCP2515 initialization|
                sets up speed, initial conditions, interrupts, GPIO,
                        and receive filters for the canbus transceiver.

                If error, PB0 is set high to turn on an error LED.
****************************************************************************/
uint8_t mcp2515_init(uint8_t speed)
{

        // resets MCP2515 and puts it into configuration mode.
        SET_L(SS);
        SPI_txrx(CAN_RESET);
        SET_H(SS);

        //I had to increase this from 10 to 20, as otherwise it would fail.
        //              The MCP2515 needed more time to reset.
        _delay_us(20);

        // load CNF1..3 Register
        SET_L(SS);
        SPI_txrx(CAN_WRITE);
        SPI_txrx(CNF3);  //address 0x28

        SPI_txrx((1<<PHSEG21));             // Bitrate 500 kbps at 16 MHz

        /*after tx the MCP2515 increments the address automatically, so you can
                continue to write into adjacent registers, so the next write goes into
                0x29, the location of CNF2*/
        SPI_txrx((1<<BTLMODE)|(1<<PHSEG11));//CFN2
    SPI_txrx(speed);//writes to BRPn of CNF1

        // activate interrupts
        SPI_txrx((1<<RX1IE)|(1<<RX0IE));
        SET_H(SS);

        // test if we could read back the value => is the chip accessible?
        if (mcp2515_read_register(CNF1) != speed) {
                SET_H(LED2);

                return false;
        }

        // deactivate the RXnBF Pins (High Impedance State)
        mcp2515_write_register(BFPCTRL, 0);

        // set TXnRTS as inputs
        mcp2515_write_register(TXRTSCTRL, 0);
        /*
        // turn off filters => receive any message
        mcp2515_write_register(RXB0CTRL, (1<<RXM1)|(1<<RXM0));
        mcp2515_write_register(RXB1CTRL, (1<<RXM1)|(1<<RXM0));
        */
/******************************************************************************
        CANBUS ID|
                Most of these have adjacent registers so we can address the register
                pairs in a single write session.

                The original code didn't include this as it was setup to receive all
                messages on the bus.
******************************************************************************/
        //enable filtering
        mcp2515_write_register(RXB0CTRL, (1<<RXM0));//buffer0
        mcp2515_write_register(RXB1CTRL, (1<<RXM0));//buffer1

        //Receive Masking:
        //block all ID's other than the exact RxID for buffer0
        SET_L(SS);
        SPI_txrx(CAN_WRITE);
        SPI_txrx(RXM0SIDH);
        SPI_txrx(0xFF);
        SPI_txrx(0xE0);
        SET_H(SS);
//      mcp2515_write_register(RXM0SIDH, 0xFF);
//      mcp2515_write_register(RXM0SIDL, 0xE0);
        //block all ID's other than the exact RxID for buffer1
        SET_L(SS);
        SPI_txrx(CAN_WRITE);
        SPI_txrx(RXM1SIDH);
        SPI_txrx(0xFF);
        SPI_txrx(0xE0);
        SET_H(SS);
//      mcp2515_write_register(RXM1SIDH, 0xFF);
//      mcp2515_write_register(RXM1SIDL, 0xE0);

        //Set RxID
        SET_L(SS);
        SPI_txrx(CAN_WRITE);
        SPI_txrx(RXF0SIDH);
        SPI_txrx(RxIDHi);
        SPI_txrx(RxIDLow);
        SET_H(SS);
//      mcp2515_write_register(RXF0SIDH, RxIDHi);//filter0
//      mcp2515_write_register(RXF0SIDL, RxIDLow);

        SET_L(SS);
        SPI_txrx(CAN_WRITE);
        SPI_txrx(RXF1SIDH);
        SPI_txrx(RxIDHi);
        SPI_txrx(RxIDLow);
        SET_H(SS);
//      mcp2515_write_register(RXF1SIDH, RxIDHi);//filter1
//      mcp2515_write_register(RXF1SIDL, RxIDLow);
```

```
       SET_L(SS);
       SPI_txrx(CAN_WRITE);
       SPI_txrx(RXF2SIDH);
       SPI_txrx(RxIDHi);
       SPI_txrx(RxIDLow);
       SET_H(SS);
//     mcp2515_write_register(RXF2SIDH, RxIDHi);//filter2
//     mcp2515_write_register(RXF2SIDL, RxIDLow);
       SET_L(SS);
       SPI_txrx(CAN_WRITE);
       SPI_txrx(RXF3SIDH);
       SPI_txrx(RxIDHi);
       SPI_txrx(RxIDLow);
       SET_H(SS);
//     mcp2515_write_register(RXF3SIDH, RxIDHi);//filter3
//     mcp2515_write_register(RXF3SIDL, RxIDLow);
       SET_L(SS);
       SPI_txrx(CAN_WRITE);
       SPI_txrx(RXF4SIDH);
       SPI_txrx(RxIDHi);
       SPI_txrx(RxIDLow);
       SET_H(SS);
//     mcp2515_write_register(RXF4SIDH, RxIDHi);//filter4
//     mcp2515_write_register(RXF4SIDL, RxIDLow);

/******************************************************************************/

       // reset device to normal mode
       mcp2515_write_register(CANCTRL, 0);
       SET_L(LED2);
       return true;
}

// ----------------------------------------------------------------------------
uint8_t mcp2515_get_message(tCAN *message)
{
       // read status
       uint8_t status = mcp2515_read_status(CAN_RX_STATUS);
       uint8_t addr;
       uint8_t t;
       if (bit_is_set(status,6)) {
              // message in buffer 0
              addr = CAN_READ_RX_BUFF;
       }
       else if (bit_is_set(status,7)) {
              // message in buffer 1
              addr = CAN_READ_RX_BUFF | 0x04;
       }
       else {
              // Error: no message available
              return 0;
       }

       SET_L(SS);
       SPI_txrx(addr);

       // read id
       message->id  = (uint16_t) SPI_txrx(0xff) << 3;
       message->id |=            SPI_txrx(0xff) >> 5;

       SPI_txrx(0xff);
```

```
        SPI_txrx(0xff);

        // read DLC
        uint8_t length = SPI_txrx(0xff) & 0x0f;

        message->header.length = length;
        message->header.rtr = (bit_is_set(status, 3)) ? 1 : 0;

        // read data
        for (t=0;t<length;t++) {
                message->data[t] = SPI_txrx(0xff);
        }
        SET_H(SS);

        // clear interrupt flag
        if (bit_is_set(status, 6)) {
                mcp2515_bit_modify(CANINTF, (1<<RX0IF), 0);
        }
        else {
                mcp2515_bit_modify(CANINTF, (1<<RX1IF), 0);
        }

        return (status & 0x07) + 1;
}

// ----------------------------------------------------------------------------
uint8_t mcp2515_send_message(tCAN *message)
{
        uint8_t status = mcp2515_read_status(CAN_READ_STATUS);

        /*  status info from data sheet:
          Bit   Function
           2    TXB0CNTRL.TXREQ
           4    TXB1CNTRL.TXREQ
           6    TXB2CNTRL.TXREQ
         */
        uint8_t address;
        uint8_t t;

        if (bit_is_clear(status, 2)) {
                address = 0x00;
        }
        else if (bit_is_clear(status, 4)) {
                address = 0x02;
        }
        else if (bit_is_clear(status, 6)) {
                address = 0x04;
        }
        else {
                // all buffer used => could not send message
                return 0;
        }

        SET_L(SS);
        SPI_txrx(CAN_LOAD_TX_BUFF | address);

        //split 11bit ID into it's respective register positions
        SPI_txrx(message->id >> 3);
    SPI_txrx(message->id << 5);

        SPI_txrx(0);
        SPI_txrx(0);

        uint8_t length = message->header.length & 0x0f;

        if (message->header.rtr) {
                // a rtr-frame has a length, but contains no data
                SPI_txrx((1<<RTR) | length);
        }
        else {
                // set message length
                SPI_txrx(length);

                // data
                for (t=0;t<length;t++) {
                        SPI_txrx(message->data[t]);
                }
        }
        SET_H(SS);

        //Wait for message to "settle" in register
        _delay_us(1);

        // send message
        SET_L(SS);
        address = (address == 0) ? 1 : address;
        SPI_txrx(CAN_RTS | address);
        SET_H(SS);

        return address;
}
```