// Header-Dateien aufrufen für Befehle

#include "avr/io.h" // Globale Header-Datei für In-Output-Belegung

#include "avr/interrupt.h" // Globale Header-Datei für Interrupt

// Intitialsieren von Variablen (Deklarieren mach ich im main() weils in Simulation hier nicht funktionierte)

*int8\_t* i;

*int8\_t* data;

// UNTERPROGRAMME DEKLARIEREN

void init\_variabs(void);

void init\_pin(void);

void init\_can\_generell(void);

//UNTERPROGRAMME ############################################################################################################################################################################

// Startwert für Variablen festlegen

void init\_variabs(void)

{

i = 0;

}

// Pinbelegung festlegen

void init\_pin(void)

{

// Pin PB2 zur LED-Ansteuerung

DDRB = 0b00000100; // Pin B2 also Ausgang, Rest-Pins also Input, da nicht benötigt.

PORTB = 0b11111011; // Pin B2 ausschalten. Rest-Pins eingeschalten, da diese als Input deklariert sind, wurde jetzt Pull-Up-Resistor aktiviert.

DDRD = 0b10000000;

PORTD = 0b01111111;

DDRC = 0b00010000;

PORTC = 0b11101111;

}

// Generelle Einstellungen für CAN um Daten zu senden

void init\_can\_generell(void)

{

CANGCON = (1<<SWRES); // Can-Controller mit Hardware Reseted.

CANGSTA = (1<<ENFG); // Enable CAN-Controller

for (i=0; i<6; i++) // alle MOb's zurücksetzen.

{

CANPAGE = (i<<4); // select MOb-Number, um 4 shiften, da MOBNB0 das 5. Bit ist.

CANCDMOB = 0x00; // disable MOb, transmission and reception

CANSTMOB = 0x00; // clear status

CANIDT1 = 0x00; // clear ID

CANIDT2 = 0x00;

CANIDT3 = 0x00;

CANIDT4 = 0x00;

CANIDM1 = 0x00; // clear mask

CANIDM2 = 0x00;

CANIDM3 = 0x00;

CANIDM4 = 0x00;

CANMSG = 0x00; // Daten zurücksetzen

}

CANBT1 = 0x0E; // Baudrate mit 250 KBaud bei fCLKio = 16MHz festgelegt

CANBT2 = 0x04; // ...

CANBT3 = 0x13; // ...

/\*CANTCON = 0x00; // CAN-Timer mit 0 geprescaled\*/

CANGCON = (1<<ENASTB); // // CAN-Channel wechselt in enable mode wenn 11 recessive Bits gelesen wurden

CANPAGE = (1<<MOBNB0) ; // MOb1 ausgewählt. Dies ist etwas unübersichtlich, siehe hierzu Erklärung in Doku unter Probleme.

CANIDM1 = 0x00; // Alle ID's passieren lassen

CANIDM2 = 0x00;

CANIDM4 = (1<<IDEMSK); // Muss nicht gesetzt werden, ´wird aber unter 16.5.2.3 RX Data and Remote Frame aufgeführt

CANCDMOB = (1<<CONMOB1) | (1<<DLC0); // set to receive, 1-byte in message

CANGIE = (1<<ENIT) | (1<<ENRX); // enable receive interrupt

CANIE1 = 0x00;

CANIE2 = (1<<IEMOB1); // enable ISR wenn MOB1 beschrieben wird

CANSIT1 = 0x00; // Wenn mehrere MOb's interrupt auslösen sollen

CANSIT2 = 0x00; //...

}

//HAUPTPROGRAMM ############################################################################################################################################################################

int main(void)

{

init\_variabs();

init\_pin();

init\_can\_generell();

sei(); //Erlaube Globale Interrupts

while(1)

{/\* PROGRAMMSCHLEIFE\*/

}

return(0);

}

ISR(CAN\_INT\_vect)

{

data = CANMSG ;

/\*PORTB = 0b11111111;\*/

// ROTE LED (PB2) leuchtet, Orangene LED (PC4) aus, Blaue LED (PD7) aus.

if (data == 0x55)

{

DDRD = 0b00000000; // Alle=Eingang

PORTD = 0b00000000; // Alle=Aus

DDRC = 0b00000000; // Alle=Eingang

PORTC = 0b00000000; // Alle=Aus

DDRB = 0b00000100; // PB2=Ausgang, Rest=Eingang

PORTB = 0b11111111; // PB2=High, Rest=PullUp

}

// BLAUE LED (PD7) leuchtet, Orangene LED (PC4) aus, Rote LED (PB2) aus.

else if (data == 0xCC)

{

DDRB = 0b00000000; // Alle=Eingang

PORTB = 0b00000000; // Alle=Aus

DDRC = 0b00000000; // Alle=Eingang

PORTC = 0b00000000; // Alle=Aus

DDRD = 0b10000000; // PD7=Ausgang, Rest=Eingang

PORTD = 0b11111111; // PD7=High, Rest= PullUp

}

// ORANGENE LED (PC4) leuchtet, Rote LED (PB2) aus, Blaue LED (PD7) aus.

else

{

DDRD = 0b00000000; // Alle=Eingang

PORTD = 0b00000000; // Alle=Aus

DDRB = 0b00000000; // Alle=Eingang

PORTB = 0b00000000; // Alle=Aus

DDRC = 0b00010000; // PC4=Ausgang, Rest=Eingang

PORTC = 0b11111111; // PC4=High, Rest=PullUp

}

for (i=0; i<6; i++) // alle MOb's zurücksetzen.

{

CANPAGE = (i<<4); // select MOb-Number, um 4 shiften, da MOBNB0 das 5. Bit ist.

CANCDMOB = 0x00; // disable MOb, transmission and reception

CANSTMOB = 0x00; // clear status

CANIDT1 = 0x00; // clear ID

CANIDT2 = 0x00;

CANIDT3 = 0x00;

CANIDT4 = 0x00;

CANIDM1 = 0x00; // clear mask

CANIDM2 = 0x00;

CANIDM3 = 0x00;

CANIDM4 = 0x00;

}

CANSTMOB =(0<<RXOK); // RXOK zurücksetzen

}

// PROGRAMMENDE ############################################################################################################################################################################