```
001
002 //------------------------------------------------------------------------
003 // Title      : AVR C Grundgerüst für ATmega8
004 //------------------------------------------------------------------------
005 // Funktion   : ...
006 // Schaltung  : ...
007 //------------------------------------------------------------------------
008 // Prozessor  : ATmega8
009 // Takt       : 16 MHz
010 // Sprache    : C
011 // Date       : ...
012 // Version    : ...
013 // Autor      : ...
014 //------------------------------------------------------------------------
015 #define F_CPU 16000000
016 #include <avr\io.h>
017 #include "main.h"
018 #include <stdio.h>
019 //------------------------------------------------------------------------
020
021 //===Befehlsdefintionen===//
022
023 //#define sbi(var,mask) ((var)|=(uint_8)(1<<mask)) //verkürzte Schreibwei
024 //#define cbi(var,mask) ((var)&=(uint_8)~(1<<mask))
025
026 //===IO-Definitionen===//
027
028 //IO Pins Atmega8
029
030 #define LED PB5
031
032 //BMA defines
033
034 #define CSB PB0     //Pin for Chip-Select
035 #define INT PB1     //Pin for Interrupt (PD2 ist INT0 beim Atmega8)
036
037 //===Prototypen-Allgemein===//
038
039 void ioinit(void);
040 void delay_ms(uint16_t x);
041 void delay_us(uint16_t x);
042
043 //===UART-Prototypen===//
044
045 static int uart_putchar(char c, FILE *stream);
046 uint8_t uart_getchar(void);
047 static FILE mystdout= FDEV_SETUP_STREAM(uart_putchar, NULL, _FDEV_SETUP_W
048
049 //===SPI/BMA Prototypen===//
050
051 void init_SPI(void);
052 void txdata(char data);
053 char rxdata(void);
054 char read(uint8_t address);
055 void write(uint8_t address,char data);
056 int init_BMA180(uint8_t range,uint8_t bw);
057 int check_interrupt(void);
058 int test_interrupt(void);
059
060 //=========================//
061
```

```
062 //Globale Variablen
063
064 int main (void)
065 {
066    char temp;
067    signed short temp2;
068    char inkey;
069
070    ioinit();
071    init_SPI();
072    sbi(PORTB,CSB);
073    printf("\f");
074    printf("\n***************BMA180 Test*****************\n\n");
075
076    //Init BMA at +-1,5g and 10Hz, return error if not connected
077
078    while(init_BMA180(0x01,0x00)!=0)     //init_BMA() bekommt die Initiali
079    {
080         printf("Error connecting to BMA180...\n");
081         delay_ms(1000);
082    }
083
084    printf("Successfully connected to BMA180...\n");
085
086    while(test_interrupt()!=0)
087    {
088         printf("Interrupt-Test fehlgeschlagen...\n");
089         delay_ms(1000);
090    }
091
092    printf("Interrupt-Test erfolgreich...\n");
093
094    printf("\nPress any button to begin acceleration output...\n");
095    inkey=uart_getchar();
096    printf("\nACCx \tACCy \tACCz \tTEMP\n\n");
097
098    while (true)      // Mainloop
099    {
100       temp=0;
101
102       while(temp!=1)
103       {
104            temp=read(ACCXLSB)&0x01;    //Abfrage von new_data_int
105       }
106
107       temp=read(ACCXMSB);              //speichern vom MSB der X-Daten i
108       temp2=temp<<8;                   //verschieben des X-MSB um 8Bit n
109       temp2|=read(ACCXLSB);
110       temp2=temp2>>2;                  /*LSB+MSB um 2 Stellen nach recht
111                                          benötigt werden*/
112       printf("%d\t",temp2);            //Ausgabe der Daten in X-Richtung
113
114       while(temp!=1)
115       {
116            temp=read(ACCYLSB)&0x01;
117       }
118
119       temp=read(ACCYMSB);
120       temp2=temp<<8;
121       temp2|=read(ACCYLSB);
122       temp2=temp2>>2;
```

```
123       printf("%d\t",temp2);
124
125       while(temp!=1)
126       {
127             temp=read(ACCZLSB)&0x01;
128       }
129
130       temp=read(ACCZMSB);
131       temp2=temp<<8;
132       temp2|=read(ACCZLSB);
133       temp2=temp2>>2;
134       printf("%d\n",temp2);
135    }
136 }
137
138 //init_BMA180
139 //Input: range is a 3-Bit value between 0x00 and 0x06 and will set the ra
140 //       bw is a 4-Bit value between 0x00 and 0x09. Again described on pg
141 //Output: -1 on error,0 on success
142
143 int init_BMA180(uint8_t range, uint8_t bw)
144 {
145     char temp,temp1;
146     //if connected correctly, ID register schould be 3
147     if(read(Chip_ID)!=3)
148         return -1;
149     //-----------------------------------------------
150     //Set ee_w bit
151     temp=read(CTRLREG0);
152     temp|=0x10;
153     write(CTRLREG0,temp);   //Have to set ee_w to write any other registe
154     //-----------------------------------------------
155     //Set Bandwidth
156     temp=read(BWTCS);
157     temp1=bw;
158     temp1=(temp1<<BWSHIFT); /*Inhalt von Temp1 muss um 4 Bit nach links v
159                             wird. Nur im High-Byte wird die Bandbreite ei
160     temp&=(~BWMASK);        //Bitmaske löscht nur das High-Byte in temp
161     temp|=temp1;            //gewünschte Bandbreite BW wird in temp im Hi
162     write(BWTCS,temp);      //keep tcs<3:0> in BWCTS, but write new BW
163     //-----------------------------------------------
164     //Set Range
165     temp=read(OLSB1);
166     temp1=range;
167     temp1=(temp1<<RANGESHIFT);
168     temp&=(~RANGEMASK);
169     temp|=temp1;
170     write(OLSB1,temp);      //Write new range data, keep other bits the s
171     //-----------------------------------------------
172
173     return 0;
174 }
175
176 //int check_interrupt()
177 //returns: 0 or 1 depending on status of PB1
178
179 int check_interrupt(void)
180 {
181     if(PINB&(1<<1)==0)
182         return 0;
183
```

```
184     else
185         return 1;
186 }
187 
188 //Check interrupt functionallity on PB1
189 //Output: 0 on success, -1 on error
190 
191 int test_interrupt(void)
192 {
193     while(check_interrupt()!=0)
194         ;
195 
196     write(CTRLREG3,0x12);   //set new_data_int
197     delay_ms(100);
198 
199     while(check_interrupt()!=1)
200         ;
201 
202     //write reset_int control bit to 1
203     write(CTRLREG3,0x10);
204     write(CTRLREG0,0x50);
205 
206     while(check_interrupt()!=0)
207         ;
208 
209     return 0;
210 }
211 
212 void write(uint8_t address, char data)
213 {
214     //write any data byte to any single address
215     //adds a "0" to the MSB of the address byte (Write Mode)
216 
217     address&=0x7F;
218 
219     //printf("\nWriting 0x%x to 0x%x\n",data,address);
220 
221     cbi(PORTB,CSB);
222     delay_ms(1);
223     txdata(address);
224     delay_ms(1);
225     txdata(data);
226     delay_ms(1);
227     sbi(PORTB,CSB);
228 }
229 
230 char read(uint8_t address)
231 {
232     //returns the contents of any One-Byte-Register from any address
233     //sets the MSB for every address byte (READ Mode)
234 
235     char byte;
236 
237     address|=0x80;
238 
239     cbi(PORTB,CSB);
240     txdata(address);
241     byte=rxdata();
242     sbi(PORTB,CSB);
243 
244     return byte;
```

```
}

char rxdata(void)
{
    //while((SPSR&0x80)==0x80);
    SPDR=0x55;
    while((SPSR&0x80)==0x00);

    return SPDR;
}

void txdata(char data)
{
    //while((SPSR&0x80)==0x80);
    SPDR = data;
    while((SPSR&0x80)==0x00);
}

void init_SPI(void)
{
    sbi(SPCR,MSTR);     //make SPI Master
    sbi(SPCR,CPOL);
    sbi(SPCR,CPHA);     //SCL idle high, sample data on rising edge
    cbi(SPCR,SPR1); cbi(SPCR,SPR0); cbi(SPSR,SPI2X); //Fosc/4 is SPI freq
    sbi(SPCR,SPE);      //enable SPI
}

static int uart_putchar(char c, FILE *stream)
{
    if(c=='\n')

        uart_putchar('\r',stream);

    loop_until_bit_is_set(UCSRA,UDRE);
    UDR=c;

    return 0;
}

uint8_t uart_getchar(void)
{
    while(!(UCSRA&(1<<RXC)));
    return(UDR);
}

void ioinit(void)
{
    int MYUBRR=103; //Results in 9600bps @ 8MHz or 19200bps @ 16MHz

    //1=output, 0=input
    //DDRA=0b00000000; //ADC Inputs
    DDRB=0b11101101; //MISO input, PB1 input
    DDRC=0b11111111; //All outputs
    DDRD=0b11111110; //PORTD(RX on PD0)

    stdout=&mystdout;   //required for printf init

    UBRRH = (MYUBRR)>>8;
    UBRRL = MYUBRR;
    UCSRB = (1<<RXEN)|(1<<TXEN);
    UCSRC = (3<<UCSZ0);
```

```
306     UCSRA = (1<<U2X);
307 
308     TCCR2 = (1<<CS21);
309 }
310 
311 //General short delays
312 
313 void delay_ms(uint16_t x)
314 {
315     for(;x>0;x--)
316         delay_us(1000);
317 }
318 
319 void delay_us(uint16_t x)
320 {
321     while(x>256)
322     {
323         TIFR=(1<<TOV2); //Clear any interrupt flags on Timer2
324         TCNT2=0;            //256-125=131: Preload timer2 for x clicks. S
325 
326         while((TIFR&(1<<TOV2))==0);
327 
328         x-=256;
329     }
330 
331     TIFR=(1<<TOV2);     //Clear any interrupt flags on Timer2
332     TCNT2=256-x;            //256-125=131: Preload timer2 for x clicks. S
333 
334     while(TIFR&(1<<TOV2)==0);
335 }
336 
337 //-------------------------------------------------------------------------
338 
```