```
main.c:
 2
 3
       4
 5
       Formål: Main er hvor funktionerne bliver kaldt til at udfører programmet.
 6
       04-08-2020
       Udarbejdet af:
 8
       Rolf J. Godfrey s190813, Ulrik Hansen s195091, Holger Bregnhøi Weise s195118, Victor Strauss s190811
 9
       */
10
11
12
       #include <avr/io.h>
13
       #include <math.h>
14
       #include <string.h>
15
       #include "Uart1.h"
16
       #include "TIMER.h"
17
       #include "ADC.h"
18
       #include "SPI.h"
19
       #include <util/delay.h>
20
       #include <avr/pgmspace.h>
21
       #include <avr/interrupt.h>
22
       #define F_CPU 16000000UL
23
       #define USART_BAUDRATE 115200
24
       #define MYBRRD F_CPU/8/USART_BAUDRATE-1
25
26
```

```
27
28
        /* Variabel initialising */
29
        volatile int record_length = 500;
30
        volatile int comparevalue = 249;
31
        volatile char indicator = 0;
32
        volatile char shape = 0;
33
        volatile char amplitude = 0;
34
        volatile char frequency = 0;
35
        volatile char Hexarray[11];
36
        volatile char LENGTH = 15;
37
        char buffer0 = 1;
38
        char buffer1 = 0;
39
        char rx_comp = 0;
40
        int lab_flag = 1;
41
        int samplrate = 500;
42
        int i = 0;
43
        int p = 0;
44
       int l = 0;
45
        int k = 0;
46
        volatile char adc_s[1000];
47
        volatile char adc_s1[1000];
48
        volatile char adc_done = 0;
49
        volatile char adc_done1 = 0;
50
51
52
        2
```

```
53
54
55
        ISR(TIMER1_COMPB_vect){}
56
        //interrupt service rutine for ADC
57
        ISR (ADC_vect){
58
                                        //Læser ind på den første buffer når Buffer0 flag er højt
         if (buffer0 == 1){
59
           adc_s[l] = ADCH;
60
           l++;
61
62
         if (buffer1 == 1){
                                       // Læser ind på den anden buffer når Buffer1 flag er sat højt
63
           adc_s1[p] = ADCH;
64
           p++;
65
         }
66
         if (l == record_length){
                                          // Når den første buffer er fuld, så bliver Buffer0 flaget sat lavt og
67
           l = 0;
                                   // buffer1 flaget bliver sat højt så de hele siden skifter.
68
           adc_done = 1;
69
           buffer0 = 0;
70
           buffer1 = 1;
71
72
         if (p == record_length){
73
           p = 0;
74
           adc_done1 = 1;
75
           buffer0 = 1;
76
           buffer 1 = 0;
77
78
        3
```

```
79
        //interrupt service rutine for uart
 80
        ISR(USART1_RX_vect){
                                           //pakken fra UDR1 bliver fyldt på et array
 81
          Hexarray[i] = UDR1;
 82
                                  // Tæller i op til brug i main funktion
          i++;
 83
 84
        void Hex_generator(void){
 85
            if (k == 0 \&\& Hexarray[5] == 0x00){
 86
             shape = Hexarray[6];
 87
 88
            if (k == 1 \&\& Hexarray[5] == 0x00){
 89
             amplitude = Hexarray[6];
 90
 91
            if (k == 2 \&\& Hexarray[5] == 0x00){
 92
             frequency = Hexarray[6];
 93
 94
            putchUSART1(0x55);
 95
            putchUSART1(0xAA);
 96
            putchUSART1(0x00);
 97
            putchUSART1(0x0B);
 98
            putchUSART1(0x01);
 99
            putchUSART1(indicator);
100
            putchUSART1(shape);
101
            putchUSART1(amplitude);
102
            putchUSART1(frequency);
103
            putchUSART1(0x00);
104
            putchUSART1(0x00);
        4
```

```
105
106
        void check_type(void){
107
          char type = Hexarray[4];
108
          switch(type){
109
           // if type == 0x01
110
           case 0x01:
111
           //BTN0
112
           if (Hexarray[5] == 0x00){
113
             Hex_generator();
114
             //fpga send new info
115
             putcSPI_master(0x55);
116
             putcSPI_master(shape);
117
             putcSPI_master(frequency);
118
             putcSPI_master(amplitude);
119
             putcSPI_master(0xFF);
120
             putcSPI_master(0x00);
121
           }
122
           //BTN1
123
           if (Hexarray[5] == 0x01){
124
             k++;
125
             indicator = k;
126
             if(k == 3){
127
              k = 0;
128
129
             Hex_generator();
130
        5
```

```
131
         //BTN3, code and labview reset
132
           if (Hexarray[5] == 0x03){
133
             k = 0;
134
             indicator = 0;
135
             shape = 0;
136
             amplitude = 0;
137
             frequency = 0;
138
             putchUSART1(0x55);
139
             putchUSART1(0xAA);
140
             putchUSART1(0x00);
141
             putchUSART1(0x0B);
142
             putchUSART1(0x01);
143
             putchUSART1(0x00);
144
             putchUSART1(0x00);
145
             putchUSART1(0x00);
146
             putchUSART1(0x00);
147
             putchUSART1(0x00);
148
             putchUSART1(0x00);
149
             //fpga reset
150
             putcSPI_master(0x55);
151
             putcSPI_master(0x00);
152
             putcSPI_master(0x00);
153
             putcSPI_master(0x00);
154
             putcSPI_master(0x00);
155
             putcSPI_master(0x00);
156
        6
```

```
157
            break
158
            //Gemmer samplerate, udregner comparevalue og gemmer Record length
159
            case 0x02:
160
              samplrate = (Hexarray[5]<<8)|(Hexarray[6] & 0xFF);</pre>
161
              record_length = (Hexarray[7]<<8)|(Hexarray[8] & 0xFF);
162
              comparevalue = (250000UL/(samplrate))-1;
163
              Timer1(comparevalue);
164
              break;
165
166
167
         int main (void){
168
          //call of functions
169
                     //enable globalt interrupt
          sei();
170
          enableReceice_Itr1();
171
          uart1_Init(MYBRRD);
172
          ADCinit(1);
173
          Timer1(249);
174
          SPI_master_init ();
175
176
          while(1){
177
            if(rx_comp == 0){
178
                                           //check synch byte
              if(Hexarray[0]!=0x55){
179
                 i = 0;
180
181
              if (Hexarray[0] == 0x55 \&\& i > 2){ //save length of hexadecimal package
182
              LENGTH = Hexarray[3];
         7
```

```
183
184
185
             if (i > LENGTH-2){
                                       //Done reading and storing information
186
             i = 0;
187
             rx_{comp} = 1;
188
             }
189
           }
190
191
            if(rx_comp == 1){
192
             _delay_ms(10);
193
             check_type();
194
195
196
            // Data pakke sendes til LabView
197
            else{
198
             if (adc_done == 1 && rx_comp == 0){
199
               putchUSART1(0x55);
200
               putchUSART1(0xAA);
201
               putchUSART1((record_length+7)>>8);
202
               putchUSART1((char)(record_length+7));
203
               putchUSART1(0x02);
204
               for (int p = 0; p < record_length; p++){</pre>
205
                putchUSART1(adc_s[p]);
206
207
               putchUSART1(0x00);
208
               putchUSART1(0x00);
        8
```

```
209
               adc_done = 0;
210
211
212
              if (adc_done1 == 1 && rx_comp == 0){
213
               putchUSART1(0x55);
214
               putchUSART1(0xAA);
215
               putchUSART1(((record_length+7)>>8));
216
               putchUSART1((char)(record_length+7));
217
               putchUSART1(0x02);
218
               for (int p = 0; p < record_length; p++){</pre>
219
               putchUSART1(adc_s1[p]);
220
221
               putchUSART1(0x00);
222
               putchUSART1(0x00);
223
               adc_done1 = 0;
224
225
            }
226
            rx_{comp} = 0;
227
228
        ADC.c:
229
230
231
         //Formål: Dette modul initialisere ADC'en.
232
         //Created: 04-08-2020
233
         //Udarbejdet af:
234
        //Rolf J. Godfrey s190813, Ulrik Hansen s195091, Holger Bregnhøi Weise s195118, Victor Strauss s190811
         9
```

```
235
236
237
        #include <avr/io.h>
238
        #include <avr/interrupt.h>
239
        #include "ADC.h"
240
        #include "TIMER.h"
241
        #include "Uart1.h"
242
243
        void ADCinit(int channel){
244
          ADCSRA|=(1<<ADEN); // Enabler ADC'en
245
          ADCSRA|=(1<<ADATE)|(1<<ADIE); //enabler Auto trigger mode og ADC Interrupt Enable
246
          ADCSRB|=(1<<ADTS2)|(1<<ADTS0); // timer compare match B
247
          ADMUX = channel;
248
          ADMUX|=(1<<ADLAR);
249
          DIDR0 = (1<<channel);
250
          DIDR0 = 0b111111101;
251
          DIDR1 = 0xff;
252 }
253
254
255
        ADC.h:
256
257
        // Created: 04-08-2020
258
        // Rolf J. Godfrey s190813, Ulrik Hansen s195091, Holger Bregnhøi Weise s195118, Victor Strauss s190811
259
        #ifndef ADC_H_
260
        #define ADC_H_
        10
```

```
261
         extern char call_adc_done();
262
         extern void ADCinit(int channel);
263
         #endif
264
265
         SPI.c:
266
          // Created: 04-08-2020
267
          // Rolf J. Godfrey s190813, Ulrik Hansen s195091, Holger Bregnhøi Weise s195118, Victor Strauss s190811
268
         #include <avr/io.h>
269
         void SPI_master_init (){
270
           DDRB|=(1<<DDB2)|(1<<DDB1)|(1<<DDB0); //opens needed ports
271
           SPCR|=(1<<SPE)|(1<<MSTR)|(1<<CPOL)|(1<<CPHA); // sets clock rate start, sample on rising, setup on falling.
272
           SPCR|=(1<<SPR1)|(1<<SPI2X); // 500k baud
273
           PORTB = (1 < PB0); //pin b0 = 1
274
275
         void putcSPI_master(unsigned char DATA){
276
           PORTB &=\sim(1<<PB0);
277
           SPDR=DATA;// transmits data
278
          while(!(SPSR&(1<<SPIF))); // waits for data to complete
279
           PORTB|=(1<<PB0);
280
281
         SPI.h:
282
          // Created: 04-08-2020
283
          // Rolf J. Godfrey s190813, Ulrik Hansen s195091, Holger Bregnhøi Weise s195118, Victor Strauss s190811 Author: victo
284
         #ifndef SPI_H_
285
         #define SPI_H_
286
         extern void SPI_master_init ();
         11
```

```
287
         extern void SPI_slave_init();
288
         extern void putcSPI_master(unsigned char DATA);
289
         unsigned char getcSPI master(void);
290
         #endif /* INCFILE1_H_ */
291
         TIMER.c:
292
293
          // Created: 04-08-2020
294
          // Rolf J. Godfrey s190813, Ulrik Hansen s195091, Holger Bregnhøi Weise s195118, Victor Strauss s190811
295
         #include <avr/io.h>
296
         #include "ADC.h"
297
         #include "TIMER.h"
298
         #include "Uart1.h"
299
         int Comparevalue;
300
         void Timer1 (unsigned int Comparevalue){
301
          TCCR1B |=(1<<CS11)|(1<<CS10); // Sætter prescaler factor til 64 så vi får den ønskede compare match value
302
          TCCR1B |=(1<<WGM02);
                                        // Indstiller CTC mode
303
          OCR1B = Comparevalue;
                                       // Compare match value sat til at ændre sig i forhold til samplrate modtaget fra LabView
304
          OCR1A = Comparevalue;
305
          TIMSK1|=(1<<0CIE1B);
                                       // enabler timer interrupt
306
         TIMER.h:
307
308
            // Created: 04-08-2020
309
           // Rolf J. Godfrey s190813, Ulrik Hansen s195091, Holger Bregnhøi Weise s195118, Victor Strauss s190811
310
         #ifndef TIMER_H_
311
         #define TIMER_H_
312
         volatile char state;
         12
```

```
313
        //int samplrate;
314
        extern void Timer1 (unsigned int Comparevalue);
315
        extern void TimskEnable ();
316
        extern void TimskClear ();
317
       extern int Compvalcalc(char Hexarray[]);
318
        #endif
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
```

```
UART1.c:
339
340
                  // created: 04-08-2020
341
                  // Rolf J. Godfrey s190813, Ulrik Hansen s195091, Holger Bregnhøi Weise s195118, Victor Strauss s190811
342
         #include <avr/io.h>
343
         #include <avr/interrupt.h>
344
         #include "uart1.h"
345
         #define USART_BAUDRATE 115200
346
         #define baud F_CPU/8/USART_BAUDRATE-1
347
         void uart1_Init(unsigned int ubrr){
348
          UCSR1B|=(1<<RXEN1)|(1<<TXEN1)|(1<<RXCIE1); // enable receive and transmit and receive complete interrupt
349
          UCSR1C|=(1<<UCSZ10)|(1<<UCSZ11); // frame: 1 start bit, 8 data bit, no parity:
350
          UBRR1H = (unsigned char)(ubrr>>8); //baud rate values up to 16 bit therefore to registers
351
          UBRR1L = (unsigned char)ubrr;
352
          UCSR1A=(1<<U2X1); //full duplex
353
354
         char getchUSART1(void){
                                              //modtager et bit oretunerer det
355
          while (!(UCSR1A &(1<<RXC1)));
                                                 // venter til karakter er modtaget
356
          return UDR1;
357
358
         void putchUSART1(char tx){
                                                //transmiterer et byte
359
          while (!(UCSR1A&(1<<UDRE1)));
360
          UDR1 = tx;
361
362
363
         void enableReceice_Itr1(){
364
          UCSR1B|=(1<<RXCIE1);
         14
```

```
UART1.h:
366
367
                 // created: 04-08-2020
368
                 // Rolf J. Godfrey s190813, Ulrik Hansen s195091, Holger Bregnhøi Weise s195118, Victor Strauss s190811
369
370
         #ifndef UART_H_
371
         #define UART_H_
372
         #define BAUD 115200
373
         #define MYUBRRF F_CPU/8/BAUD-1 //full duplex
374
         #define MYUBRRH F_CPU/16/BAUD-1 //half duplex
375
         #define max 20 // number of data in the receive array
376
         extern void putchUSART1 (char tx);
377
         extern char getchUSART1(void);
378
         extern void uart1_Init(unsigned int ubrr);
379
         extern void enableReceice_Itr1();
380
         #endif /* UART_H_ */
381
382
383
384
385
386
387
```