Software code

HCS12

.c code files

Delay.c
iic.c
main.c
PLL.c
Port_b.c
pwm.c
RTI.C
SCI.C
timer.c

```
 1: //************************************************************
 2: //*                                                          *
 3: //* ---------------------------------------------------------- *
 4: //*  Filename: delay.c                                       *
 5: //*  Compiler: Imagecraft ICC12(V6.16A) for freescale HCS12  *
 6: //*  Made by:  Eric Halmans                                  *
 7: //*  For:      Fontys Highschool Eindhoven, Mechatronica     *
 8: //*  Date:     April 2006                                    *
 9: //*  Version:  1.0 beta test version                         *
10: //*                                                          *
11: //*  Description:                                            *
12: //*    This file is part of a set of files,                  *
13: //*    which are used in an universal AC & DC motor control. *
14: //*    (Hbridge with PWM on PP0 & PP1, SCI1 input,IIC Output) *
15: //* ---------------------------------------------------------- *
16: //*                                                          *
17: //************************************************************
18:
19:
20: // include header files
21: #include "ASM.h"
22:
23:
24: // function declarations
25:
26: // -------------------------------------------------------------------------
27: // Short description:
28: // this function is a simple iterative delay
29:
30: // pre:  None
31: // post: Program is bussy for 100 times with a No Operation Instruction
32: // -------------------------------------------------------------------------
33: void TimingShort(void)
34: {
35:   int x;
36:   for (x=0;x<255;x++)
37:   {
38:     NOP();
39:   }
40: }
41:
42:
43: // -------------------------------------------------------------------------
44: // Short description:
45: // this function is an adjustable delay bassed on
46: // the simple iterative delay wich is multiplied with the integer value temp.
47:
48: // pre:  None
49: // post: Program is bussy for (100 times with a No Operation Instruction) * temp
50: // -------------------------------------------------------------------------
51: void TimingAdjustable(int temp)
52: {
53:   int i;
54:   for(i=0;i<=temp;i++)TimingShort();
55: }
56:
57: //*************************************************************************
58: // end of Delay.c
59:
60:
```

```
 1: //************************************************************
 2: //*                                                          *
 3: //* ---------------------------------------------------------- *
 4: //*  Filename: iic.c                                         *
 5: //*  Compiler: Imagecraft ICC12(V6.16A) for freescale HCS12  *
 6: //*  Made by:  Eric Halmans                                  *
 7: //*  For:      Fontys Highschool Eindhoven, Mechatronica     *
 8: //*  Date:     April 2006                                    *
 9: //*  Version:  1.0 beta test version                         *
10: //*                                                          *
11: //*  Description:                                            *
12: //*    This file is part of a set of files,                  *
13: //*    which are used in an universal AC & DC motor control. *
14: //*    (Hbridge with PWM on PP0 & PP1, SCI1 input,IIC Output) *
15: //* ---------------------------------------------------------- *
16: //*                                                          *
17: //************************************************************
18:
19:
20: // include header files
21: #include "iic.h"
22:
23:
24: // function declarations
25:
26: // ------------------------------------------------------------------------
27: // Short description:
28: // this function initialises IIC ONCHIP hardware to 100khz bus speed,
29: // with no interrupt
30:
31: // Important: IIC hardware depends on the busspeed
32: //            this means it depends on PLL clock...
33: //            It is nearly impossible to find a correct speed for IIC
34: //            when PLL = 25 Mhz speed
35:
36: // Pre:  PLL should be set to a 24 Mhz busspeed
37: //       IIC harware registers must be defined (mc9s12dp512.h)
38:
39: // Post: IIC hardware registers are changed
40: //       ICC is set to 100 Khz and no interrupt
41: // ------------------------------------------------------------------------
42: void InitIIC(void)
43: {
44:  // PLL has to be set to 24 Mhz bus!!!!
45:  // IICCLK = 100 kbit/s
46:  // IICCLK DIVIDER = BUSCLK / ICCCLK = 24M/100K = 240
47:  // IIC Adresregister -> NO value <- Master mode
48:  // IIC Freq Register -> Look for 240 in table -> one value 0x1F
49:  IBFD = 0x1f;
50:
51:  // IBCR control register:
52:  // 7 - IBEN  I-bus Enable                       1 (I bus enable)
53:  // 6 - IBIE  I-bus Interrupt Enable             0 (Interrupt disable)
54:  // 5 - MS/NOT SL Master slave select            1 (Master select)
55:  // 4 - TX / NOT RX Transmit receive mode select 1 (TX mode)
56:  // 3 - TXAK Transmit acknowledge enable         1 (TXAC enable)
57:  // 2 - Repeat Start                             0 (NOT repeat start)
58:  // 1 - RESERVED                                 0
59:  // 0 - IBSWAI  wait modus                       0 (NO wait modus)
60:  IBCR= 0xB8;
```

```
 61:
 62: }
 63:
 64:
 65: // ------------------------------------------------------------------------
 66: // Short description:
 67: // This function sends 0x50, the adress of the LCD, to establish a conection
 68:
 69: // Pre:  IIC should be initialised
 70: // Post: The character 0x50 is send to the LCD module.
 71: //       This is the default adresvalue stored in the LCD Module
 72: // ------------------------------------------------------------------------
 73: void SendIICadress(void)
 74: {
 75:   // wait for IICbus to be free
 76:   while ((IBSR&0x20)==0);
 77:   // send character 0x50 (Adres character)
 78:   IBDR= 0x50;
 79:   // wait for a while, in wich at least the character is send over.
 80:   TimingShort();
 81: }
 82:
 83:
 84: // ------------------------------------------------------------------------
 85: // Short description:
 86: // This function sends the codes through IIC bus to clear the LCD
 87:
 88: // Pre:  Adres has to be send to LCD
 89: // Post: LCD is cleared, cursor = at first lcd Character
 90: // ------------------------------------------------------------------------
 91: void ClearIIC(void)
 92: {
 93:   // wait for IICbus to be free
 94:   while ((IBSR&0x20)==0);
 95:   // send character 0xfe (Character which inits special mode)
 96:   IBDR= 0xfe;
 97:   // wait for a while, in wich at least the character is send over.
 98:   TimingShort();
 99:   // wait for IICbus to be free
100:   while ((IBSR&0x20)==0);
101:   // send character 0x58 (special mode character which codes for clear LCD)
102:   IBDR= 0x58;
103:   // wait for a while, in wich at least the character is send over.
104:   TimingShort();
105: }
106:
107:
108: // ------------------------------------------------------------------------
109: // Short description:
110: // this function sends a character to the LCD
111:
112: // pre:  IIC Adres has to be send
113: // post: unsigned character a is send so LCD
114: // ------------------------------------------------------------------------
115: void SendIICbyte(unsigned char a)
116: {
117:    while ((IBSR&0x20)==0);
118:    IBDR = a;
119:    TimingShort();
120: }
```

```
121:
122: //*****************************************************************************
123: // end of iic.c
124:
```

```
1: //************************************************************
2: //*                                                          *
3: //* ---------------------------------------------------------- *
4: //*  Filename: main.c                                        *
5: //*  Compiler: Imagecraft ICC12(V6.16A) for freescale HCS12  *
6: //*  Made by:  Eric Halmans                                  *
7: //*  For:      Fontys Highschool Eindhoven, Mechatronica     *
8: //*  Date:     April 2006                                    *
9: //*  Version:  1.0 beta test version                         *
10: //*                                                         *
11: //*  Description:                                           *
12: //*    This file is part of a set of files,                 *
13: //*    which are used in an universal AC & DC motor control.*
14: //*    (Hbridge with PWM on PP2 & PP3, SCI1 input,IIC Output)*
15: //* ---------------------------------------------------------- *
16: //* History:                                                *
17: //*                                                         *
18: //* Aditions to version 1.0:                                *
19: //* - Deleted unused variables                              *
20: //* - made improvements in the AC part:                     *
21: //*      - finetuned the sinewave shape by using either     *
22: //*        a 1 value or a 0 value to get symetry in         *
23: //*        the wave shape                                   *
24: //*      - improved readability of AC part by using         *
25: //*        better variables                                 *
26: //* - scanned the c & h files for readabilty                *
27: //*                                                         *
28: //************************************************************
29:
30:
31: // include header files
32: #include "main.h"
33:
34:
35: // interrupt routine declarations
36:
37: // ------------------------------------------------------------------------------
38: // Short description:
39: // SCI1 Interrupt Routine
40:
41: // Pre:  SCI1 is waiting for an incomming character,
42: //       which will invoke this routine
43:
44: // Post: if incomming character is:
45: //        - 1st char: Determines AC or DCleft or DCRight or Stop story
46: //                    or it determines RTI speed set to either 15,625 hz or khz
47: //        - Dc 2nd char: sets DC Duty value for either PWM2 or PWM3
48: //        - AC 2nd char: Awaits for AC 3th char
49: //        - AC 3th char: Awaits for AC 4th char
50: //        - AC 4nd char: Awaits for AC 5th char
51: //        - AC 5nd char: Enables main routine to calculate a new AC table
52: // ------------------------------------------------------------------------------
53: #pragma interrupt_handler SCI1_Handler
54:  void SCI1_Handler(void)
55: {
56: //_none:    (label as where to go to, only used in comment)
57:   Status = SCI1SR1;
58:   if  (Status & 0x20)
59:        {
60:          // SCI1CR2=0x0c;
```

```
 61:          Temp=SCI1DRL;
 62:          switch (StateMachine)
 63:            {
 64:
 65:           // determines if going into ac or dc mode,
 66:           // if a change in RTI frequency is been made or go into stopmode
 67:           case FirstChar:
 68:                {
 69:                 StateFirstChar();
 70:                }
 71:                break;
 72:
 73:           // if DC left: 2nd char will set PWMDTY2
 74:           // if DC right: 2nd char will set PWMDTY3
 75:           case DCSecondChar:
 76:                {
 77:                 StateDCSecondChar();
 78:                }
 79:                break;
 80:
 81:           // get first digit for AC frequency (0..9)
 82:           // and go to ACThirdChar state
 83:           // elso go back to the FirstCar state
 84:           case ACSecondChar:
 85:                {
 86:                 StateACSecondChar();
 87:                }
 88:                break;
 89:
 90:           // get seccond digit for AC frequency (0..9)
 91:           // and go to ACForthChar state
 92:           // elso go back to the FirstCar state
 93:           case ACThirdChar:
 94:                {
 95:                 StateACThirdChar();
 96:                }
 97:                break;
 98:
 99:           // get third digit for AC frequency (0..9)
100:           // and go to ACFifthChar state
101:           // elso go back to the FirstCar state
102:           case ACForthChar:
103:                {
104:                 StateACForthChar();
105:                }
106:                break;
107:
108:           // get forth digit for AC frequency (0..9)
109:           // if digit is good: enable calculate table1 in main while loop
110:           // elso go back to the FirstCar state
111:           case ACFifthChar:
112:                {
113:                 StateACFifthChar();
114:                }
115:
116:            break;
117:
118:           // in all other cases go back to the FirstCar state
119:           default:
120:                {
```

```
121:                 StateDefault();
122:                }
123:             break;
124:           }
125:         }
126: }
127:
128: // ------------------------------------------------------------------------
129: // Short description:
130: // Real Time Interrupt Routine
131: // This Routine sets and runs through the AC frequencies.
132:
133: // Pre:  RTI is waiting for an interupt, wich can be set to either 15,625 hz
134: //       or 15,625 Khz this interrupt then will invoke a proper routine
135: //       wich is of course depending on the preliminary state
136:
137: // Post: every interrupt is another step in running through the AC table1
138: // ------------------------------------------------------------------------
139: #pragma interrupt_handler RTI_Handler
140: void RTI_Handler(void)
141: {
142: if (Return==NONE)RTI_NONE();
143: else if (Return==SetTable1)RTI_SetTable1();
144: else if (Return==ForwardPositive)RTI_ForwardPositive();
145: else if (Return==BackwardPositive)RTI_BackwardPositive();
146: else if (Return==ForwardNegative)RTI_ForwardNegative();
147: else if (Return==BackwardNegative)RTI_BackwardNegative();
148: CRGFLG |=0x80;
149: }
150:
151:
152: // MAIN
153: void main(void)
154: {
155:
156: // Setup SCI interrupt
157:    *( ( unsigned char * ) 0x3fc1 ) = 0x06;
158:    *( ( void (**)(void) ) 0x3fc2 ) = SCI1_Handler;
159:
160: // Setup RTI interrupt
161:    *( ( unsigned char * ) 0x3feb ) = 0x06;
162:    *( ( void (**)(void) ) 0x3fec ) = RTI_Handler;
163:
164: // Initialisation sequence
165: // Where we are during initialisation is shown
166: // by the LEDs on the Electronicladen HCS12 T-Board
167: // Set Port B as output: enable LEDS
168:    InitPortB();
169: // Turns first LED on
170:    PORTB=0xfe;
171: // Wait
172:    TimingAdjustable(255);
173: // Initializes SCI1 with 19200 bit/s busspeed and receive interrupt
174:    InitSCI1();
175: // Turns seccond LED on
176:    PORTB=0xFD;
177: // Wait
178:    TimingAdjustable(255);
179: // Initializes Phase Lock Loop to 48 Mhz
180:    InitPLL48();
```

```
181: // Turns 3th LED on
182:    PORTB=0xFB;
183: // Wait
184:    TimingAdjustable(255);
185: // Initializes PWM sets freqency to 31,25 hz
186:    InitPWM();
187:    // Disables PWM outputs on all channels: PP0..PP7
188:    PWME=0x00;
189: // Turns 4th LED on
190:    PORTB=0xEF;
191: // Wait
192:    TimingAdjustable(255);
193: // Turns 5th LED on
194:    PORTB=0xDF;
195: // Wait
196:    TimingAdjustable(255);
197: // Initializes Real time interrupt to 15.625 Khz
198:    InitRTI();
199: // Turns 6th LED on
200:    PORTB=0xBF;
201: // Wait
202:    TimingAdjustable(255);
203: // Initializes IIC, busspeed 100Khz
204:    InitIIC();
205: // Turns 7th LED on
206:    PORTB=0x7F;
207: // Wait
208:    TimingAdjustable(255);
209: // Set while loop switch to write opening txt to LCD
210:    MainStateMachine=SendCharIIC;
211:    IICModus=IICNONE;
212: // enable interrupt
213:    CLI();
214: // Main loop forever
215:    while(1)
216:     {
217:      switch (MainStateMachine)
218:      {
219:       // this case builds thew ac table
220:       // it combines the four digits received through SCI1 into an float
221:       // and it starts calculating integer numbers, wich correspond with
222:       // how many RTI interrupts only a counter is upgraded with 1.
223:       // if the counter equals the value in the table,
224:       // the next dutystep is been made
225:       // it sets the PWMMODUS1 &
226:       // it also enables new values to be written to IIC
227:       case BuildAC:
228:       {
229:          MainStateBuildAC();
230:       }
231:       break;
232:
233:       // this case looks at what and how to be writen to LCD
234:       case SendCharIIC:
235:       {
236:          MainStateSendCharIIC();
237:       }
238:       break;
239:
240:      case PWMNone:
```

```
241:      {
242:         NOP();
243:      }
244:      break;
245:
246:      case TimerRead:
247:      {
248:      //MainStateMachine=TimerRead;
249:      //Timer_ON();
250:      //Timer_OFF();
251:      // SendTimer();
252:      // MainStateMachine=PWMNone;
253:      }
254:      break;
255:
256:      default:
257:      {
258:         NOP();
259:      }
260:      break;
261:      }
262:    }
263: }
264:
265: //*****************************************************************************
266: // end of main.c
267:
```

```
 1: //************************************************************
 2: //*                                                          *
 3: //* ---------------------------------------------------------- *
 4: //*  Filename: pll.c                                         *
 5: //*  Compiler: Imagecraft ICC12(V6.16A) for freescale HCS12  *
 6: //*  Made by:  Eric Halmans                                  *
 7: //*  For:      Fontys Highschool Eindhoven, Mechatronica     *
 8: //*  Date:     April 2006                                    *
 9: //*  Version:  1.0 beta test version                         *
10: //*                                                          *
11: //*  Description:                                            *
12: //*    This file is part of a set of files,                  *
13: //*    which are used in an universal AC & DC motor control. *
14: //*    (Hbridge with PWM on PP0 & PP1, SCI1 input,IIC Output) *
15: //* ---------------------------------------------------------- *
16: //*                                                          *
17: //************************************************************
18:
19:
20: // include header files
21: #include "PLL.h"
22:
23:
24: // function declarations
25:
26: // -------------------------------------------------------------------------
27: // Short description:
28: // this sets the PLL Phase Lock loop to 48 Mhz
29: // This is part of the CRG module
30:
31: // pre:  PLL harware registers must be defined (mc9s12dp512.h)
32: // post: Busclock is set to 48 Mhz
33: // -------------------------------------------------------------------------
34: void InitPLL48(void)
35: {
36:   // PLLCLK = 2 * OSCCLK * ((SYNR+1)/(REFDV+1)) = 48 Mhz...
37:   // -> BUSCLK = 24 Mhz
38:   SYNR = 0x02;
39:   REFDV = 0x01;
40:
41:   // Enable PLL bit in CLKSEL
42:   CLKSEL = 0x80;
43: }
44:
45: /*
46: // -------------------------------------------------------------------------
47: // Short description:
48: // this sets the PLL Phase Lock loop to 50 Mhz
49:
50: // pre:  PLL harware registers must be defined (mc9s12dp512.h)
51: // post: Busclock is set to 50 Mhz (Max Bus Speed)
52: // -------------------------------------------------------------------------
53: void InitPLL50(void)
54: {
55:   // PLLCLK = 2 * OSCCLK * ((SYNR+1)/(REFDV+1)) = 48 Mhz...
56:   SYNR = 0x18;
57:   REFDV = 0x0F;
58:
59:   // Enable PLL bit in CLKSEL
60:   CLKSEL = 0x80;
```

```
61: }
62: */
63:
64: //*****************************************************************************
65: // end of PLL.c
66:
```

```
 1: //************************************************************
 2: //*                                                          *
 3: //* ---------------------------------------------------------- *
 4: //*  Filename: Port_b.c                                       *
 5: //*  Compiler: Imagecraft ICC12(V6.16A) for freescale HCS12    *
 6: //*  Made by:  Eric Halmans                                   *
 7: //*  For:      Fontys Highschool Eindhoven, Mechatronica      *
 8: //*  Date:     April 2006                                     *
 9: //*  Version:  1.0 beta test version                          *
10: //*                                                          *
11: //*  Description:                                            *
12: //*    This file is part of a set of files,                  *
13: //*    which are used in an universal AC & DC motor control.  *
14: //*    (Hbridge with PWM on PP0 & PP1, SCI1 input,IIC Output)  *
15: //* ---------------------------------------------------------- *
16: //*                                                          *
17: //************************************************************
18:
19:
20: // include header files
21: #include "Port_b.h"
22:
23:
24: // ------------------------------------------------------------------------
25: // Short description:
26: // this sets Port B as an output and gives it the value: 11111111
27:
28: // pre:  PortB harware registers must be defined (mc9s12dp512.h)
29: // post: PortB is set as output and is given the value 0xFF
30: //       On the HCS12 TBoard, PortB is connected to a Led Array
31: // ------------------------------------------------------------------------
32: void InitPortB(void)
33: {
34:  // Port B register set -> Port B is now an output
35:  DDRB  |= 0xFF;
36:  // value 0xff written to Port b -> All lights are on
37:  PORTB |= 0xFF;
38: }
39:
40: //************************************************************************
41: // end of Port_b.c
```

```
 1: //************************************************************
 2: //*                                                          *
 3: //* ---------------------------------------------------------- *
 4: //*  Filename: pwm.c                                         *
 5: //*  Compiler: Imagecraft ICC12(V6.16A) for freescale HCS12  *
 6: //*  Made by:  Eric Halmans                                  *
 7: //*  For:      Fontys Highschool Eindhoven, Mechatronica     *
 8: //*  Date:     April 2006                                    *
 9: //*  Version:  1.0 beta test version                         *
10: //*                                                          *
11: //*  Description:                                            *
12: //*    This file is part of a set of files,                  *
13: //*    which are used in an universal AC & DC motor control. *
14: //*    (Hbridge with PWM on PP0 & PP1, SCI1 input,IIC Output) *
15: //* ---------------------------------------------------------- *
16: //*                                                          *
17: //************************************************************
18:
19:
20: // include header files
21: #include "pwm.h"
22:
23:
24: // function declarations
25:
26: // -------------------------------------------------------------------------
27: // Short description:
28: // this function initialises all the PWM registers  to a freqency of 31,25 Khz
29: // and 256 steps
30:
31: // pre:  PWM harware registers must be defined (mc9s12dp512.h)
32: // post: PWM registers are set:
33: //          - polarity is positive (0x00)
34: //          - Clock is Clock a for PWM0..PWM3 and Clock is clock b for PWM4..PWM7
35: //          - Clock prescaler is set to 128 for clock a and b
36: //          - PWM are all left aligned
37: //          - 8 x 8 bit clock (No 16 bit clock)
38: //          - PWM Periods are set to 256 steps for all clocks
39: // -------------------------------------------------------------------------
40: void InitPWM(void)
41: {
42:   // PWM polarity = 1 => all the same (0xff = 100 %, 0x00 = 0 %)
43:   // do not change this!!!!!! Gives big trouble!!!!
44:   PWMPOL=0xff;
45:
46:   // Selection of SA/SB or A/B clock (All SA & SB)
47:   // All Clock A & B = PWMCLK=0x00;
48:   PWMCLK=0x00;
49:
50:   // PWM prescaler clock A & B  x-b-b-b-x-a-a-a
51:   // THERE is also a scale register used for SA and SB !!!!!
52:   // ( move down )
53:     // 000 => Busclk
54:     // 001 => Busclk /   2
55:     // 010 => Busclk /   4
56:     // 011 => Busclk /   8
57:     // 100 => Busclk /  16
58:     // 101 => Busclk /  32
59:     // 110 => Busclk /  64
60:     // 111 => Busclk / 128
```

```
61:   // 1 / F counter = (1/busclock) * (dutysteps+1) * prescaler * 2
62:   PWMPRCLK=0x00;
63:
64:   // PWM Centre or left align
65:   // All left align = 0x00
66:   PWMCAE=0x00;
67:
68:   // PWM 8 bit 16 bit selection
69:   // All 8 bit x 8 channels
70:   PWMCTL=0x00;
71:
72:   // set port P as an output (used in hc11) should be disabled in HCS12)
73:   //DDRP=0xff;
74:
75:   // PWM Scale register
76:   // Clock SA = Clock A / (2 * PWMSCLA)
77:   // Clock SB = Clock B / (2 * PWMSCLB)
78:   // PWMSCLA=0x ;
79:   // PWMSCLB=0x ;
80:   // we use clock a and clock b so no scaling!
81:   // PWM countervalues
82:   // used to force reset PWM not used in setup PWMCNT0..7
83:   // PWM Channel period registers
84:   // PWMPER0..7
85:   PWMPER0=0xff;
86:   PWMPER1=0xff;
87:   PWMPER2=0xff;
88:   PWMPER3=0xff;
89:   PWMPER4=0xff;
90:   PWMPER5=0xff;
91:   PWMPER6=0xff;
92:   PWMPER7=0xff;
93: }
94:
95: //*****************************************************************************
96: // end of pwm.c
97:
```

```
 1: //************************************************************
 2: //*                                                          *
 3: //* ---------------------------------------------------------- *
 4: //*  Filename: RTI.c                                         *
 5: //*  Compiler: Imagecraft ICC12(V6.16A) for freescale HCS12  *
 6: //*  Made by:  Eric Halmans                                  *
 7: //*  For:      Fontys Highschool Eindhoven, Mechatronica     *
 8: //*  Date:     April 2006                                    *
 9: //*  Version:  1.0 beta test version                         *
10: //*                                                          *
11: //*  Description:                                            *
12: //*    This file is part of a set of files,                  *
13: //*    which are used in an universal AC & DC motor control. *
14: //*    (Hbridge with PWM on PP0 & PP1, SCI1 input,IIC Output) *
15: //* ---------------------------------------------------------- *
16: //*                                                          *
17: //************************************************************
18:
19:
20: // include header files
21: #include "RTI.h"
22:
23: // function declarations
24:
25: // function declarations
26:
27: // ------------------------------------------------------------------------------
28: // Short description:
29: // this initializes the Real Time Interrupt to 15,625 hz
30: // (Maximal frequency for RTI /1000)
31: // This is part of the CRG module
32:
33: // pre:  RTI harware registers must be defined (mc9s12dp512.h)
34: // post: RTI registers are set:
35: //         - RTI COntrol register is set to 0x75
36: //         - RTI interupt is enabled  (optional)
37: // ------------------------------------------------------------------------------
38:
39: void InitRTI(void)
40: {
41:   // sets frequency of Real time Interupt to 15.625 hz
42:   // see for othert values the table in the CRG documentation
43:   // factor 1000 bigger : RTICTL |= 0x10  (15.625 Khz)
44:   RTICTL = 0x10;
45:
46:   // enables RTI Interrupt
47:   // CRGINT = 0x80;
48: }
49:
50: //*****************************************************************************
51: // end of RTI.c
52:
53: void StopRTI(void)
54: {
55:   // disables RTI Interrupt
56:   CRGINT = 0x00;
57: }
58:
```

```
1: //************************************************************
2: //*                                                          *
3: //* ---------------------------------------------------------- *
4: //*  Filename: sci.c                                          *
5: //*  Compiler: Imagecraft ICC12(V6.16A) for freescale HCS12    *
6: //*  Made by:  Eric Halmans                                    *
7: //*  For:      Fontys Highschool Eindhoven, Mechatronica       *
8: //*  Date:     April 2006                                      *
9: //*  Version:  1.0 beta test version                           *
10: //*                                                          *
11: //*  Description:                                            *
12: //*    This file is part of a set of files,                  *
13: //*    which are used in an universal AC & DC motor control.  *
14: //*    (Hbridge with PWM on PP0 & PP1, SCI1 input,IIC Output)  *
15: //* ---------------------------------------------------------- *
16: //*                                                          *
17: //************************************************************
18:
19:
20: // include header files
21: #include "SCI.H"
22:
23:
24: // function declarations
25:
26: // ------------------------------------------------------------------------------
27: // Short description:
28: // this initialises de SCI1 Port to interupt on receive
29:
30: // pre:  SCI harware registers must be defined (mc9s12dp512.h)
31: // post: RTI registers are set:
32: //       - Baudrate is set  to 19200 bit/s
33: //       - Receiver interupt, receiver enable and transmitter enable are set
34: //       - Interrupt flag is cleared (OPTIONAL)
35: // ------------------------------------------------------------------------------
36: void InitSCI1(void)
37: {
38:   // local character definition used for resetting the SCI interrupt status
39:   unsigned char StatusLocal;
40:
41:   // sets baudrate to 19200 bit/s
42:
43:   // BusCLK =16 Mhz (not 6 Mhz)
44:   // SCICLK = 8Mhz
45:   // Define baudrate SCI Baud Rate = SCICLK / (16*BR)
46:   // PCLK = 16 Mhz -> Baudrate = 8M/(16*br) bit/s.
47:      //   1200 bit/s = 416 dec.      -> 0x01A0
48:      //   2400 bit/s = 208 dec.      -> 0x00D0
49:      //   4800 bit/s = 104 dec.      -> 0x0068
50:      //   9800 bit/s =  51 dec.      -> 0x0033
51:      //  19200 bit/s =  26 dec.      -> 0x001A
52:      //  38400 bit/s =  13 dec.      -> 0x000D
53:      //  56000 bit/s =   9 dec.      -> 0x0009
54:      // 128000 bit/s =   4 dec.      -> 0x0004
55:      // 256000 bit/s =   2 dec.      -> 0x0002
56:      // 512000 bit/s =   1 dec.      -> 0x0001
57:   SCI1BDH= 0x00;
58:   SCI1BDL= 0x1A;
59:
60:   // SCI1CR2:
```

```
  // Transmitter Interrupt enable          SCI1CR2[7]=0
  // Transmision complete Interrupt enable SCI1CR2[6]=0
  // Receiver Full Interrupt enable        SCI1CR2[5]=1
  // Idle line interrupt enable            SCI1CR2[4]=0
  // Transmitter enable                    SCI1CR2[3]=1
  // Receiver enable                       SCI1CR2[2]=1
  // Reciever wake up bit                  SCI1CR2[1]=0
  // Send break bit                        SCI1CR2[0]=0
  SCI1CR1=0x00;
  SCI1CR2=0x2c;

  //clear TDRE flag two step proces
  //   (1) read SCI1SR1+(SCI1SR2)
  //   (2) write (SC1DRH)+SC1DRL
  StatusLocal = SCI1SR1;
  SCI1DRL = 0x00;
}

// ------------------------------------------------------------------------------
// Short description:
// this function sends a character through SCI1

// pre:  SCI harware registers must be defined (mc9s12dp512.h)
//       SCI registers must have been set
//       TDRE (bit7) must have been declared
// post: sends character X to SCI1
// ------------------------------------------------------------------------------
void SendChar(char X)
{
  //wait for the SCI1 port to be inactive
  while((SCI1SR1&TDRE)==0){};

  // send character
  SCI1DRL=X;
}

void ClearReceive(void)
{
  char Temper=0;
  Temper=SCI1SR1;
  Temper=SCI1DRL;
}


void SendInt(int *X)
{
  int Y,i;
  Y=*X;
  SendChar(Read_Low(Y));
  for (i=0;i<255;i++)NOP();
  SendChar(Read_High(Y));
}

// this function takes an int and reads out its lower byte.
char Read_Low(int x)
{
char y;
int z;
z=x;
y=(char)z;
```

```
121: return y;
122: }
123:
124: // this function takes an int and reads out its higher byte.
125: char Read_High(int x)
126: {
127: char y;
128: int z;
129: z=x;
130: z >>= 8;
131: y=(char)z;
132: return y;
133: }
134:
135: //***************************************************************************
136: // end of SCI.c
137:
138:
```

```c
#include "timer.h"
#include "mc9s12dp512.h"

// this function turns on the timer.
void Timer_ON(void)
{
TSCR1=0x80;
}

// this function turns off the timer.
void Timer_OFF(void)
{
TSCR1=0x00;
}

// this function gets the timer value.
int Get_Time(void)
{
int x;
x=TCNT;
return x;
}

// this function takes an int and reads out its lower byte.
char Timer_Read_Low(int x)
{
char y;
y=(char)x;
return y;
}

// this function takes an int and reads out its higher byte.
char Timer_Read_High(int x)
{
char y;
x >>= 8;
y=(char)x;
return y;
}
```