Software code

HCS12

.h header files

ASM.h
Delay.h
Floattable.h
iic.h
main.h
PLL.h
Port_b.h
pwm.h
RTI.h
SCI.h
timer.h

```
 1: //*************************************************************
 2: //*                                                           *
 3: //* ----------------------------------------------------------*
 4: //*  Filename: ASM.h                                          *
 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: // Assembler operations defines
21:
22: // No Operation as an assembler statement
23: #define NOP() asm("nop");
24:
25: // Enable all Interrupts as an assembler statement
26: #define CLI() asm("cli");
27:
28: // Disable all Interruptsas an assembler statement
29: #define SEI() asm("sei");
30:
31: //*****************************************************************************
32: // end of ASM.h
```

```
 1: //************************************************************
 2: //*                                                          *
 3: //* ---------------------------------------------------------- *
 4: //*  Filename: Delay.h                                         *
 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: // function declarations
21:
22: // -------------------------------------------------------------------------------
23: // Short description:
24: // this function is a simple iterative delay
25:
26: // pre:  None
27: // post: Program is bussy for 100 times with a No Operation Instruction
28: // -------------------------------------------------------------------------------
29: void TimingShort(void);
30:
31: // -------------------------------------------------------------------------------
32: // Short description:
33: // this function is an adjustable delay bassed on
34: // the simple iterative delay wich is multiplied with the integer value temp.
35:
36: // pre:  None
37: // post: Program is bussy for (100 times with a No Operation Instruction) * temp
38: // -------------------------------------------------------------------------------
39: void TimingAdjustable(int temp);
40:
41: //*******************************************************************************
42: // end of Delay.h
43:
```

```
 1:
 2: //*************************************************************
 3: //*                                                           *
 4: //* ----------------------------------------------------------*
 5: //*  Filename: Floattable.h                                   *
 6: //*  Compiler: Imagecraft ICC12(V6.16A) for freescale HCS12   *
 7: //*  Made by:  Eric Halmans                                   *
 8: //*  For:      Fontys Highschool Eindhoven, Mechatronica      *
 9: //*  Date:     April 2006                                     *
10: //*  Version:  1.0 beta test version                          *
11: //*                                                           *
12: //*  Description:                                             *
13: //*    This file is part of a set of files,                   *
14: //*    which are used in an universal AC & DC motor control.  *
15: //*    (Hbridge with PWM on PP0 & PP1, SCI1 input,IIC Output) *
16: //* ----------------------------------------------------------*
17: //*                                                           *
18: //*************************************************************
19:
20:
21: // constant declarations
22:
23: // ------------------------------------------------------------------------------
24: // Short description:
25: // This file declares constant tables to be stored in the eeprom area
26:
27: // pre:  Project->Options->Target->Other Options: -bmyarea:0x400.0x0fff
28: // post: tables are to be found in the eeprom, and the program code knows
29: // ------------------------------------------------------------------------------
30: // define absolute start adres to store
31: // (Wich is the start adres of the eeprom on chip 0x0400..0x0fff)
32: #pragma abs_address:0x0400
33:
34: // these float tables generate a sine like signal on the PWM output,
35: // if correctly used
36: // the idea is like this:
37: // The floatvalue times the total amounts of steps, gives a step value,
38: // wich corresponds with a sine if thew PWM stays the same for this stepvalue
39: // after that the next step is entered and again has to wait again for:
40: //  total amount of steps * float value
41:
42: float array4bit[5]=
43: {0.160861,0.172472,0.206560,0.460107,NULL};
44:
45: float array8bit[9]=
46: {0.079786,0.081075,0.083853,0.088619,0.096469,0.110091,0.138384,0.321722,NULL};
47:
48: float array16bit[17]=
49: {0.039815,0.039971,0.040291,0.040784,0.041472,0.042382,0.043557,0.045061,
50:  0.046987,0.049481,0.052782,0.057309,0.063895,0.074490,0.095454,0.226268,NULL};
51:
52: float array32bit[33]=
53: {0.019898,0.019917,0.019956,0.020015,0.020095,0.020196,0.020319,0.020465,
54:  0.020637,0.020835,0.021062,0.021320,0.021613,0.021944,0.022319,0.022742,
55:  0.023222,0.023766,0.024386,0.025096,0.025915,0.026867,0.027987,0.029322,
56:  0.030942,0.032953,0.035526,0.038964,0.043858,0.051597,0.066696,0.159572,NULL};
57:
58: float array64bit[65]=
59: {0.009948,0.009950,0.009955,0.009962,0.009972,0.009984,0.009999,0.010016,
60:  0.010036,0.010059,0.010084,0.010112,0.010143,0.010176,0.010213,0.010253,
```

```
 0.010295,0.010341,0.010391,0.010444,0.010501,0.010561,0.010626,0.010694,
 0.010768,0.010845,0.010928,0.011016,0.011110,0.011209,0.011315,0.011427,
 0.011547,0.011675,0.011810,0.011955,0.012110,0.012276,0.012453,0.012643,
 0.012847,0.013067,0.013305,0.013562,0.013841,0.014146,0.014478,0.014844,
 0.015247,0.015695,0.016195,0.016758,0.017397,0.018129,0.018980,0.019983,
 0.021188,0.022670,0.024551,0.027046,0.030574,0.036121,0.046886,0.112687,NULL};

float array128bit[129]=
{0.004974,0.004974,0.004975,0.004975,0.004977,0.004978,0.004980,0.004982,
 0.004985,0.004987,0.004990,0.004994,0.004997,0.005002,0.005006,0.005010,
 0.005015,0.005021,0.005026,0.005032,0.005039,0.005045,0.005052,0.005060,
 0.005067,0.005075,0.005084,0.005093,0.005102,0.005111,0.005121,0.005131,
 0.005142,0.005153,0.005165,0.005177,0.005189,0.005202,0.005215,0.005229,
 0.005243,0.005258,0.005273,0.005288,0.005304,0.005321,0.005338,0.005356,
 0.005374,0.005393,0.005413,0.005433,0.005453,0.005475,0.005497,0.005519,
 0.005543,0.005567,0.005592,0.005617,0.005644,0.005671,0.005699,0.005728,
 0.005758,0.005789,0.005821,0.005854,0.005888,0.005923,0.005959,0.005996,
 0.006035,0.006075,0.006116,0.006159,0.006203,0.006249,0.006297,0.006346,
 0.006397,0.006450,0.006505,0.006562,0.006622,0.006683,0.006748,0.006815,
 0.006884,0.006957,0.007033,0.007113,0.007196,0.007283,0.007374,0.007470,
 0.007571,0.007677,0.007788,0.007906,0.008031,0.008164,0.008304,0.008454,
 0.008613,0.008784,0.008966,0.009163,0.009375,0.009605,0.009855,0.010128,
 0.010428,0.010760,0.011128,0.011541,0.012009,0.012542,0.013160,0.013886,
 0.014754,0.015820,0.017169,0.018953,0.021469,0.025416,0.033057,0.079629,NULL};

float array256bit[257]=
{0.002487,0.002487,0.002487,0.002487,0.002487,0.002487,0.002488,0.002488,
 0.002488,0.002489,0.002489,0.002489,0.002490,0.002490,0.002491,0.002491,
 0.002492,0.002493,0.002493,0.002494,0.002495,0.002496,0.002496,0.002497,
 0.002498,0.002499,0.002500,0.002501,0.002502,0.002503,0.002505,0.002506,
 0.002507,0.002508,0.002510,0.002511,0.002512,0.002514,0.002515,0.002517,
 0.002519,0.002520,0.002522,0.002523,0.002525,0.002527,0.002529,0.002531,
 0.002533,0.002535,0.002537,0.002539,0.002541,0.002543,0.002545,0.002547,
 0.002550,0.002552,0.002554,0.002557,0.002559,0.002562,0.002564,0.002567,
 0.002570,0.002572,0.002575,0.002578,0.002581,0.002584,0.002587,0.002590,
 0.002593,0.002596,0.002599,0.002603,0.002606,0.002609,0.002613,0.002616,
 0.002620,0.002623,0.002627,0.002631,0.002634,0.002638,0.002642,0.002646,
 0.002650,0.002654,0.002658,0.002663,0.002667,0.002671,0.002676,0.002680,
 0.002685,0.002689,0.002694,0.002699,0.002704,0.002709,0.002714,0.002719,
 0.002724,0.002729,0.002735,0.002740,0.002746,0.002751,0.002757,0.002763,
 0.002768,0.002774,0.002780,0.002787,0.002793,0.002799,0.002805,0.002812,
 0.002819,0.002825,0.002832,0.002839,0.002846,0.002853,0.002860,0.002868,
 0.002875,0.002883,0.002891,0.002898,0.002906,0.002914,0.002923,0.002931,
 0.002940,0.002948,0.002957,0.002966,0.002975,0.002984,0.002993,0.003003,
 0.003013,0.003022,0.003032,0.003043,0.003053,0.003063,0.003074,0.003085,
 0.003096,0.003107,0.003119,0.003130,0.003142,0.003154,0.003167,0.003179,
 0.003192,0.003205,0.003218,0.003232,0.003246,0.003260,0.003274,0.003288,
 0.003303,0.003318,0.003334,0.003350,0.003366,0.003382,0.003399,0.003416,
 0.003433,0.003451,0.003469,0.003488,0.003507,0.003526,0.003546,0.003566,
 0.003587,0.003608,0.003630,0.003653,0.003675,0.003699,0.003723,0.003747,
 0.003772,0.003798,0.003825,0.003852,0.003880,0.003909,0.003938,0.003968,
 0.004000,0.004032,0.004065,0.004099,0.004134,0.004170,0.004208,0.004246,
 0.004286,0.004327,0.004370,0.004414,0.004460,0.004507,0.004556,0.004607,
 0.004660,0.004715,0.004773,0.004832,0.004895,0.004960,0.005028,0.005100,
 0.005175,0.005253,0.005336,0.005423,0.005515,0.005613,0.005716,0.005825,
 0.005942,0.006067,0.006200,0.006343,0.006497,0.006663,0.006844,0.007041,
 0.007258,0.007497,0.007762,0.008058,0.008393,0.008775,0.009217,0.009735,
 0.010355,0.011114,0.012074,0.013342,0.015129,0.017928,0.023341,0.056288,NULL};

// these character tables are used as content to be displayed on the LCD
```

```
unsigned char IICDCForwardTable[33]=
{'D','C',' ','F','O','R','W','A','R','D',' ','M','O','D','E',' ',
 ' ',' ','D','U','T','Y',':',' ',NULL,' ',' ',' ',' ',' ',' ',' ',NULL};

unsigned char IICDBackwardTable[33]=
{'D','C',' ','B','A','C','K','W','A','R','D',' ','M','O','D','E',
 ' ',' ','D','U','T','Y',':',' ',NULL,' ',' ',' ',' ',' ',' ',' ',NULL};

unsigned char IICACTable[33]=
{'A','C',' ','M','O','D','E',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',
 'F','R','E','Q','U','E','N','C','Y',':',' ',NULL,' ',' ',' ',' ',NULL};

unsigned char IICNONETable[33]=
{'N','o',' ','A','c','t','i','v','e',' ','S','i','g','n','a','l',
 '(','c',')',' ','E','J','H','M',' ','H','a','l','m','a','n','s',NULL};

// here ends the absolute adres value (no more stuff for the eeprom)
#pragma end_abs_address

//***************************************************************************
// end of Floattable.h
```

```
 1: //************************************************************
 2: //*                                                          *
 3: //* ---------------------------------------------------------- *
 4: //*  Filename: iic.h                                         *
 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: #ifndef mc9s12dp512_h
22: #include "mc9s12dp512.h"
23: #endif
24:
25: #ifndef Delay_h
26: #include "Delay.h"
27: #endif
28:
29: // function declarations
30:
31: // -------------------------------------------------------------------------
32: // Short description:
33: // this function initialises IIC ONCHIP hardware to 100khz bus speed,
34: // with no interrupt
35:
36: // Important: IIC hardware depends on the busspeed
37: //            this means it depends on PLL clock...
38: //            It is nearly impossible to find a correct speed for IIC
39: //            when PLL = 25 Mhz speed
40:
41: // Pre:  PLL should be set to a 24 Mhz busspeed
42: //       IIC harware registers must be defined (mc9s12dp512.h)
43:
44: // Post: IIC hardware registers are changed
45: //       ICC is set to 100 Khz and no interrupt
46: // -------------------------------------------------------------------------
47: void InitIIC(void);
48:
49: // -------------------------------------------------------------------------
50: // Short description:
51: // This function sends 0x50, the adress of the LCD, to establish a conection
52:
53: // Pre:  IIC should be initialised
54: // Post: The character 0x50 is send to the LCD module.
55: //       This is the default adresvalue stored in the LCD Module
56: // -------------------------------------------------------------------------
57: void SendIICadress(void);
58:
59: // -------------------------------------------------------------------------
60: // Short description:
```

```
61: // This function sends the codes through IIC bus to clear the LCD
62:
63: // Pre:  Adres has to be send to LCD
64: // Post: LCD is cleared, cursor = at first lcd Character
65: // ---------------------------------------------------------------------------
66: void ClearIIC(void);
67:
68: // ---------------------------------------------------------------------------
69: // Short description:
70: // this function sends a character to the LCD
71:
72: // pre:  IIC Adres has to be send
73: // post: unsigned character a is send so LCD
74: // ---------------------------------------------------------------------------
75: void SendIICbyte(unsigned char a);
76:
77: //***************************************************************************
78: // end of iic.h
79:
```

```
 1: //*************************************************************
 2: //*                                                           *
 3: //* ----------------------------------------------------------*
 4: //*  Filename: main.h                                         *
 5: //*  Compiler: Imagecraft ICC12(V6.16A) for freescale HCS12   *
 6: //*  Made by:  Eric Halmans                                   *
 7: //*  For:      Fontys Highschool Eindhoven, Mechatronica      *
 8: //*  Date:     August 2006                                    *
 9: //*  Version:  1.1 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: #ifndef _mc9s12dp512_h
33: #include "mc9s12dp512.h"
34: #endif
35:
36: #ifndef _stdlib_h
37: #include <stdlib.h>
38: #endif
39:
40: #ifndef _PLL_h
41: #include "PLL.h"
42: #endif
43:
44: #ifndef _Port_b_h
45: #include "Port_b.h"
46: #endif
47:
48: #ifndef _Delay_h
49: #include "Delay.h"
50: #endif
51:
52: #ifndef _SCI_h
53: #include "SCI.h"
54: #endif
55:
56: #ifndef _ASM_h
57: #include "ASM.h"
58: #endif
59:
60: #ifndef _pwm_h
```

```
 61: #include "pwm.h"
 62: #endif
 63:
 64: #ifndef _RTI_h
 65: #include "RTI.h"
 66: #endif
 67:
 68: #ifndef _iic_h
 69: #include "iic.h"
 70: #endif
 71:
 72: #ifndef _timer_h
 73: #include "timer.h"
 74: #endif
 75:
 76: #ifndef _Floattable_h
 77: #include "Floattable.h"
 78: #endif
 79:
 80: // Global variables
 81: // defines
 82:
 83: #ifndef NULL
 84: #define NULL 0
 85: #endif
 86:
 87: #ifndef MAXINT
 88: #define MAXINT 65500
 89: #endif
 90:
 91: // iic test vars
 92: int i,ii;
 93: int * pointii=&ii;
 94: unsigned char *pointtofloattoascii;
 95: float Duty1;
 96: float Duty2;
 97:
 98: // SCI characters to define frequency AC PWM
 99: char first;
100: char seccond;
101: char third;
102: char forth;
103:
104: // Floats to use in building of AC tables
105: float Frequency=0.0;
106:
107: // characters for SCI interrupt
108: char Temp=0x00;
109: char DCLastTemp0=0x10;
110: char DCLastTemp1=0x10;
111: char TempDC=0x00;
112: char Status=0x00;
113:
114: //Pointer to ICC Display table
115: unsigned char * PointIIC;
116:
117: // first PWM buffer fixed value of max 256 steps
118: // (gets filled if the other table is running)
119: unsigned int Table1[258];
120:
```

```
121: // Step trough counters for PWM Table
122: int Count1;
123:
124: // Stepthrough counter for AC
125: int TableIndex;
126:
127: // Pointers used for stepping troug AC table
128: //unsigned int *PointTable1=NULL;
129: // unsigned int *FixedTable1=&Table1[0];
130:
131: // Compare value to compare with count... used for stepping trough active table
132: int  Value1=0;
133:
134: // used to finetune ac values... no 0xff or 0x00 values in AC PWM output
135: char Substractor;
136: char PreSubtract=0x00;
137:
138: // Pointers used for Building AC table
139: unsigned int *PointCalc=NULL;
140: float *FloatCalc=NULL;
141:
142: //  Main loop switch statement... defines states in main loop
143: enum MSM {BuildAC,SendCharIIC,PWMNone,TimerRead};
144: enum MSM MainStateMachine=PWMNone;
145:
146: // used as switch statement for the AC table1 walkthrough
147: enum RTN {ForwardPositive,BackwardPositive,ForwardNegative,
148:                BackwardNegative,SetTable1,NONE};
149: enum RTN Return=ForwardPositive;
150:
151: // used as switch statement for the incomming SCI1 characters
152: enum States{FirstChar,DCSecondChar,DCThirdChar,ACSecondChar,
153:                ACThirdChar,ACForthChar,ACFifthChar};
154: enum States StateMachine=FirstChar;
155:
156: // used for the DC direction
157: enum Turn{Left,Right,NoDC};
158: enum Turn Direction;
159:
160: // used as switch statement for wat to send to IIC
161: enum IIC {IICDCForward,IICDCBackward,IICAC,IICNONE};
162: enum IIC IICModus=IICNONE;
163:
164: // used to determine length of integer IIC value
165: int Max_Char;
166:
167:
168: // used to put send table on
169: int SendTable=0;
170: int TableEnd=0;
171: unsigned int Tempur=0;
172: char ToSendChar;
173:
174:
175: // interrupt routine declarations
176:
177:
178: // ---------------------------------------------------------------------------
179: // Short description:
180: // SCI1 Interrupt Routine
```

```
181:
182: // Pre:  SCI1 is waiting for an incomming character,
183: //       which will invoke this routine
184:
185: // Post: if incomming character is:
186: //        - 1st char: Determines AC or DCleft or DCRight or Stop story
187: //                    or it determines RTI speed set to either 15,625 hz or khz
188: //        - Dc 2nd char: sets DC Duty value for either PWM2 or PWM3
189: //        - AC 2nd char: Awaits for AC 3th char
190: //        - AC 3th char: Awaits for AC 4th char
191: //        - AC 4nd char: Awaits for AC 5th char
192: //        - AC 5nd char: Enables main routine to calculate a new AC table
193: // ------------------------------------------------------------------------
194: void RTI_Handler(void);
195:
196:
197: // ------------------------------------------------------------------------
198: // Short description:
199: // Real Time Interrupt Routine
200: // This Routine sets and runs through the AC frequencies.
201:
202: // Pre:  RTI is waiting for an interupt, wich can be set to either 15,625 hz
203: //       or 15,625 Khz this interrupt then will invoke this routine
204:
205: // Post: every interrupt is another step in running through the AC table1
206: // ------------------------------------------------------------------------
207: void SCI1_Handler(void);
208:
209:
210:
211: // ------------------------------------------------------------------------
212: // Short description:
213: // The routines down here all are called during RTI overflow.
214: // These routines deterrmine where the AC shaping is at a sertain moment in time
215:
216: //       If RTI_ForwardPositive is finished -> Go to RTI_BackwardPositive
217: //       if RTI_BackwardPositive is finished -> Go to RTI_ForwardNegative
218: //       if RTI_ForwardNegative is finished -> Go to RTI_BackwardNegative
219: //       if RTI_BackwardNegative is finished -> Go to RTI_ForwardPositive
220: //       and so on.
221: //       If we just enter a AC story, then RTI_SetTable1 comes
222: //       preleminary to the above story.
223: //       If we dont want an AC Signal anymore RTI_None is called
224: // ------------------------------------------------------------------------
225:
226: // disables RTI
227: void RTI_NONE(void);
228:
229:  // Start off point for running through the AC table
230: void RTI_SetTable1(void);
231:
232:   // first state of running throug the AC table:
233:   // PWM2 is going upwards to 100% duty
234: void RTI_ForwardPositive(void);
235:
236:  // seccond state of running throug the AC table:
237:  // PWM2 is going downward to 0% duty
238: void RTI_BackwardPositive(void);
239:
240:  // third state of running throug the AC table:
```

```
241:  // PWM3 is going upward to 100% duty
242: void RTI_ForwardNegative(void);
243:
244:  // forth state of running throug the AC table:
245:  // PWM3 is going downward to 0% duty
246: void RTI_BackwardNegative(void);
247:
248:
249: // this function is used to measure the amount of HCS12 processor time spend
250: // on a seartain statement or function
251: // only jused for debugging purposes
252: // void SendTimer(void);
253:
254: void StateFirstChar(void);
255:
256: void StateDCSecondChar(void);
257:
258: void StateACSecondChar(void);
259:
260: void StateACThirdChar(void);
261:
262: void StateACForthChar(void);
263:
264: void StateACFifthChar(void);
265:
266: void StateDefault(void);
267:
268:
269: void MainStateBuildAC(void);
270:
271: void MainStateSendCharIIC(void);
272:
273:
274:
275:
276:
277: // function declarations
278:
279:
280: // -------------------------------------------------------------------------
281: // Short description:
282: // This function translates 4 digit numbers, into one float
283:
284:
285: // Pre:  char x,y,z,s are given digets (0..9)
286:
287: // Post: if x=1,y=2,z=3,s=4 then return will be the float 123.4
288: // -------------------------------------------------------------------------
289: float CharsToFloat(char x, char y, char z, char s)
290: {
291:    int i=0;
292:    float ii=0;
293:
294:    i=100*(int)x+10*(int)y+(int)z;
295:    ii=(float)(i+((int)s/10));
296:    return ii;
297: }
298:
299:
300: // -------------------------------------------------------------------------
```

```
301: // Short description:
302: // This function calculates the amount of steps, integer return value,
303: // based uppon a value (in the eeprom tables) and the disired frequency
304: // all is depending on the RTI frequency
305:
306: // Pre:  pointer to eeprom table value, frequency
307:
308: // Post: integer value saying how long to stay in this dutystep
309: // -------------------------------------------------------------------------
310: unsigned int CalcSteps(float *TableFloat, float Freq)
311: {
312:    float Temp1;
313:    int Temp2;
314:    //unsigned char Temp3;
315:
316:    Temp1 = (*TableFloat * 3906.25) / ( Freq );
317:    Temp2= (unsigned int)Temp1;
318:    //Temp3= (unsigned char)Temp2;
319:    return Temp2;
320: }
321:
322:
323:  // disables RTI
324: void RTI_NONE(void)
325:  {
326:   SEI();
327:   CRGINT = 0x00;
328:   CLI();
329:  }
330:
331:  // Start off point for running through the AC table
332: void RTI_SetTable1(void)
333:   {
334: //     PointTable1=&Table1[0];
335: //     PointTable1++;
336:      TableIndex=1;
337:      Return=ForwardPositive;
338:      Count1=0;
339:      PWME=0x04;
340:      PWMDTY2=0x00; //01
341:      CRGINT=0x80;
342:   }
343:
344: // preliminary commentry:
345: // Sometimes and it looks like random adjusted your Count1 code starts
346: // with a one and in other cases it starts with a zero.
347: // THis is realy strange and in theory they realy should have been the same.
348: // But my scope picture of the signal showed an strange inconsistancy:
349: // It was not at all semetricaly. It took me a long while to figure the one and
350: // It seems to have to do with the compiler and the way you walk through a table
351: // if you descent a table you have to start counting from zero,
352: // if you are ascending you have to initiate count at 1.
353: // Of course this is as puzzling to me as it might be to you.
354: // I only konw that my signal now by using ones and zero's is what you want it t
355: // Like what good tape can do for a real project, some triks work out perfect in
356: // Software. (Especialy if you compiler is not top of the art)
357:
358:  // first state of running throug the AC table:
359:  // PWM2 is going upwards to 100% duty
360: void RTI_ForwardPositive(void)
```

```
  {
     if (Count1<Value1)
      {
        Count1=Count1+1;
      }
     else if (Count1>=Value1)
      {
        TableIndex=TableIndex+1;
        if (!(Table1[TableIndex]==(MAXINT)))
           {
             Value1=Table1[TableIndex];
             // notice here count starts with a 1!
             Count1=0;
             PreSubtract=0xff;
             PreSubtract-=Substractor;
             if (PWMDTY2>=PreSubtract)PWMDTY2=0xff;
             else PWMDTY2=PWMDTY2+Substractor;
           }
          else if (Table1[TableIndex]==(MAXINT))
           {
             TableIndex=TableEnd;
             Value1=Table1[TableIndex];
             Count1=0;
             PWME=0x04;
             PWMDTY2=0xff;
             PWMDTY3=0x00;
             Return=BackwardPositive;
           }

      }
  }

 // seccond state of running throug the AC table:
 // PWM2 is going downward to 0% duty
void RTI_BackwardPositive(void)
    {
      if (Count1<Value1)
       {
         Count1=Count1+1;
       }
      else if (Count1>=Value1)
       {
         TableIndex=TableIndex-1;
         if (!(Table1[TableIndex]==(MAXINT-1)))
            {
              Value1=Table1[TableIndex];
              // notice here count starts with a 0!
              Count1=0;
              PreSubtract=(0x01+Substractor);
              if (PWMDTY2<=PreSubtract)PWMDTY2=0x00;
              else PWMDTY2=PWMDTY2-Substractor;
            }
         else if (Table1[TableIndex]==(MAXINT-1))
            {
              TableIndex=1;
              Value1=Table1[TableIndex];
              Count1=0;
              PWME=0x08;
              PWMDTY3=0x00;
              PWMDTY2=0x00;
```

```
421:               Return=ForwardNegative;
422:             }
423:
424:         }
425:     }
426:
427:  // third state of running throug the AC table:
428:  // PWM3 is going upward to 100% duty
429: void RTI_ForwardNegative(void)
430:   {
431:      if (Count1<Value1)
432:       {
433:         Count1=Count1+1;
434:       }
435:      else if (Count1>=Value1)
436:       {
437:         TableIndex=TableIndex+1;
438:         if (!(Table1[TableIndex]==(MAXINT)))
439:            {
440:              Value1=Table1[TableIndex];
441:              // notice here count starts with a 1!
442:              Count1=0;
443:              PreSubtract=0xfe;
444:              PreSubtract-=Substractor;
445:              if (PWMDTY3>=PreSubtract)PWMDTY3=0xff;
446:              else PWMDTY3=PWMDTY3+Substractor;
447:            }
448:         else if (Table1[TableIndex]==(MAXINT))
449:            {
450:              TableIndex=TableEnd;
451:              Value1=Table1[TableIndex];
452:              Count1=0;
453:              PWME=0x08;
454:              PWMDTY3=0xff;
455:              PWMDTY2=0x00;
456:              Return=BackwardNegative;
457:            }
458:
459:       }
460:   }
461:
462:  // Forth state of running throug the AC table:
463:  // PWM3 is going downward to 0% duty
464: void RTI_BackwardNegative(void)
465:     {
466:
467:       if (Count1<Value1)
468:        {
469:          Count1=Count1+1;
470:        }
471:       else if (Count1>=Value1)
472:        {
473:          TableIndex=TableIndex-1;
474:          if (!(Table1[TableIndex]==(MAXINT-1)))
475:            {
476:              Value1=Table1[TableIndex];
477:              // notice here count starts with a 0!
478:              Count1=0;
479:              PreSubtract=0x01+Substractor;
480:              if (PWMDTY3<=PreSubtract)PWMDTY3=0x00;
```

```
481:              else PWMDTY3=PWMDTY3-Substractor;
482:             }
483:           else if (Table1[TableIndex]==(MAXINT-1))
484:            {
485:              TableIndex=1;
486:              Value1=Table1[TableIndex];
487:              Count1=0;
488:              PWME=0x04;
489:              PWMDTY3=0x00;
490:              PWMDTY2=0x00; //01
491:              Return=ForwardPositive;
492:             }
493:         }
494:     }
495:
496:
497: // only used for debugging
498: /*
499: void SendTimer(void)
500: // Timer measurement:
501: {
502:  //Timer_ON();
503:  //Timer_OFF();
504:  //TimerCount=Get_Time();
505:  //StopRTI();
506:  //SendChar(Timer_Read_Low(TimerCount));
507:  //SendChar(Timer_Read_High(TimerCount));
508: }
509: */
510:
511: //
512:
513: void StateFirstChar(void)
514: {
515:  // stopmode
516:  if (Temp==0x00)
517:   {
518:     PWMDTY2=0x00;
519:     PWMDTY3=0x00;
520:     PWME=0x00;
521:     StateMachine=FirstChar;
522:     SEI();
523:     CRGINT = 0x00;
524:     Return=NONE;
525:     CLI();
526:     if (IICModus!=IICNONE)
527:        {
528:          IICModus=IICNONE;
529:          MainStateMachine=SendCharIIC;
530:         }
531:     ClearReceive();
532:    }
533: // set RTI to 15.625 hz
534:   else if (Temp==0x05)
535:    {
536:      PWME=0x00;
537:      PWMDTY2=0x00;
538:      PWMDTY3=0x00;
539:      StateMachine=FirstChar;
540:      SEI();
```

```
541:       CRGINT = 0x00;
542:       RTICTL = 0x7f;
543:       Return=NONE;
544:       CLI();
545:       ClearReceive();
546:     }
547: // set RTI to 15.625 Khz
548:   else if (Temp==0x10)
549:     {
550:       PWME=0x00;
551:       PWMDTY2=0x00;
552:       PWMDTY3=0x00;
553:       StateMachine=FirstChar;
554:       SEI();
555:       CRGINT = 0x00;
556:       RTICTL = 0x10;
557:       Return=NONE;
558:       CLI();
559:       ClearReceive();
560:     }
561: // disables the sending of characters wich form the choosen AC Table
562:   else if (Temp==0x20)
563:     {
564:       PWME=0x00;
565:       PWMDTY2=0x00;
566:       PWMDTY3=0x00;
567:       StateMachine=FirstChar;
568:       SendTable=0;
569:       ClearReceive();
570:     }
571: // enables the sending of characters wich form the choosen AC Table
572:   else if (Temp==0x25)
573:     {
574:       PWME=0x00;
575:       PWMDTY2=0x00;
576:       PWMDTY3=0x00;
577:       StateMachine=FirstChar;
578:       SendTable=1;
579:       ClearReceive();
580:     }
581: // enter AC character sequence
582:   else if (Temp==0x55)
583:     {
584:       PWME=0x00;
585:       PWMDTY2=0x00;
586:       PWMDTY3=0x00;
587:       StateMachine=ACSecondChar;
588:       ClearReceive();
589:     }
590: // enter DC Right sequence
591:     else if (Temp==0x52)
592:      {
593:        SEI();
594:        CRGINT = 0x00;
595:        CLI();
596:        Return=NONE;
597:        PWMDTY2=0x00;
598:        PWMDTY3=0x00;
599:        PWME=0x00;
600:        Direction=Right;
```

```
601:       StateMachine=DCSecondChar;
602:       ClearReceive();
603:     }
604: // enter DC Left sequence
605:    else if (Temp==0x4c)
606:     {
607:       SEI();
608:       CRGINT = 0x00;
609:       CLI();
610:       Return=NONE;
611:       PWME=0x00;
612:       Direction= Left;
613:       StateMachine=DCSecondChar;
614:       CRGINT = 0x00;
615:       ClearReceive();
616:     }
617: // in other cases go back to FirstChar state to expect a first char
618:    else
619:     {
620:       StateMachine=FirstChar;
621:       PWME=0x00;
622:       CRGINT = 0x00;
623:       ClearReceive();
624:     }
625: }
626:
627:
628:
629:
630: void StateDCSecondChar(void)
631: {
632:   TempDC=SCI1DRL;
633:   if (Direction==Left)
634:    {
635:      PWMDTY2=TempDC;
636:      PWMDTY3=0xff;
637:      DCLastTemp0=TempDC;
638:      PWME=0x04;
639:      MainStateMachine=SendCharIIC;
640:      IICModus=IICDCBackward;
641:      StateMachine=FirstChar;
642:      ClearReceive();
643:    }
644:   else if (Direction==Right)
645:    {
646:      PWMDTY3=TempDC;
647:      PWMDTY2=0xff;
648:      DCLastTemp1=TempDC;
649:      PWME=0x08;
650:      MainStateMachine=SendCharIIC;
651:      IICModus=IICDCForward;
652:      StateMachine=FirstChar;
653:      ClearReceive();
654:    }
655:    // in other cases go to FirstChar state to expect a first char
656:    else
657:     {
658:       StateMachine=FirstChar;
659:       PWMDTY3=0xff;
660:       PWMDTY2=0xff;
```

```
661:       PWME=0x00;
662:       SEI();
663:       CRGINT = 0x00;
664:       CLI();
665:       ClearReceive();
666:     }
667: }
668:
669: void StateACSecondChar(void)
670: {
671:   if (Temp<=0x09)
672:    {
673:      first=Temp;
674:      StateMachine=ACThirdChar;
675:      ClearReceive();
676:    }
677:    else
678:    {
679:      StateMachine=FirstChar;
680:      ClearReceive();
681:    }
682: }
683:
684:
685: void StateACThirdChar(void)
686: {
687:   if (Temp<=0x09)
688:    {
689:      seccond=Temp;
690:      StateMachine=ACForthChar;
691:      ClearReceive();
692:    }
693:   else
694:    {
695:      StateMachine=FirstChar;
696:      ClearReceive();
697:    }
698: }
699:
700:
701: void StateACForthChar(void)
702: {
703:   if (Temp<=0x09)
704:    {
705:      StateMachine=ACFifthChar;
706:      third=Temp;
707:      ClearReceive();
708:    }
709:   else
710:    {
711:      StateMachine=FirstChar;
712:      ClearReceive();
713:    }
714: }
715:
716: void StateACFifthChar(void)
717: {
718:   if (Temp<=0x09)
719:    {
720:      forth=Temp;
```

```
721:      MainStateMachine=BuildAC;
722:      StateMachine=FirstChar;
723:      ClearReceive();
724:    }
725:   else
726:    {
727:      StateMachine=FirstChar;
728:      ClearReceive();
729:    }
730: }
731:
732:
733:
734: void StateDefault(void)
735: {
736:  StateMachine=FirstChar;
737:  SEI();
738:  CRGINT = 0x00;
739:  CLI();
740:  ClearReceive();
741: }
742:
743: ////////////////////////////////////
744:
745: void MainStateBuildAC(void)
746: {
747:         SEI();
748:         PWME=0x00;
749:         Frequency=CharsToFloat(first,seccond,third,forth);
750:         PointCalc=&Table1[0];
751:         while (PointCalc!=&Table1[257])
752:          {
753:            *PointCalc=MAXINT;
754:            PointCalc++;
755:          }
756:         *PointCalc=-2;
757:         PointCalc=&Table1[0];
758:         if ((Frequency>=1.0)  && (Frequency<5.0))
759:          {
760:            FloatCalc = &array256bit[0];
761:            TableEnd=256;
762:            Substractor=0x01;
763:          }
764:         else if ((Frequency>=5.0) && (Frequency<10.0))
765:          {
766:            FloatCalc = &array128bit[0];
767:            TableEnd=128;
768:            Substractor=0x02;
769:          }
770:         else if ((Frequency>=10.0) && (Frequency<20.0))
771:          {
772:            FloatCalc = &array64bit[0];
773:            TableEnd=64;
774:            Substractor=0x04;
775:          }
776:         else if ((Frequency>=20.0) && (Frequency<40.0))
777:          {
778:            FloatCalc = &array32bit[0];
779:            TableEnd=32;
780:            Substractor=0x08;
```

```
781:          }
782:         else if ((Frequency>=40.0)&& (Frequency<80.0))
783:          {
784:            FloatCalc = &array16bit[0];
785:            TableEnd=16;
786:            Substractor=0x10;
787:          }
788:         else if ((Frequency>=80.0)&& (Frequency<160.0))
789:          {
790:            FloatCalc = &array8bit[0];
791:            TableEnd=8;
792:            Substractor=0x20;
793:          }
794:         else if ((Frequency>=160.0)&& (Frequency<=320.0))
795:          {
796:            FloatCalc = &array4bit[0];
797:            TableEnd=4;
798:            Substractor=0x40;
799:          }
800:        PointCalc=&Table1[0];
801:        *PointCalc=MAXINT-1;
802:        PointCalc++;
803:        while (*FloatCalc!=NULL)
804:          {
805:            Tempur=CalcSteps(FloatCalc,Frequency);
806:            *PointCalc=Tempur;
807:            if(SendTable==1)
808:             {
809:               ToSendChar= Timer_Read_High(*PointCalc);
810:               while ((SCI1SR1 & TDRE) == 0){};
811:               SCI1DRL = ToSendChar;
812:               TimingShort();
813:
814:               ToSendChar = Timer_Read_Low(*PointCalc);
815:               while ((SCI1SR1 & TDRE) == 0){};
816:               SCI1DRL = ToSendChar;
817:               TimingShort();
818:             }
819:            FloatCalc++;
820:            PointCalc++;
821:          }
822:        *PointCalc=MAXINT;
823:        PWME=0x00;
824:        Return=SetTable1;
825:        MainStateMachine=SendCharIIC;
826:        IICModus=IICAC;
827:        CRGINT = 0x80;
828:        CLI();
829:       }
830:
831:
832:
833:
834: void MainStateSendCharIIC(void)
835:       {
836:         if (IICModus==IICDCForward)
837:          {
838:            PointIIC=&IICDCForwardTable[0];
839:            Duty1=(((float)DCLastTemp1/256.0)*100);
840:            if (Duty1<10.0)Max_Char=3;
```

```
841:            else if (Duty1<100.0)Max_Char=4;
842:            else Max_Char=5;
843:            pointtofloattoascii=ftoa(Duty1,pointii);
844:          }
845:         else if (IICModus==IICDCBackward)
846:          {
847:            PointIIC=&IICDBackwardTable[0];
848:            Duty2=(((float)DCLastTemp0/256.0)*100);
849:            if (Duty2<10.0)Max_Char=3;
850:            else if (Duty2<100.0)Max_Char=4;
851:            else Max_Char=5;
852:            pointtofloattoascii=ftoa(Duty2,pointii);
853:          }
854:         else if (IICModus==IICAC)
855:          {
856:            PointIIC=&IICACTable[0];
857:            pointtofloattoascii=ftoa(Frequency,pointii);
858:            if (Frequency<10.0)Max_Char=3;
859:            else if (Frequency<100.0)Max_Char=4;
860:            else Max_Char=5;
861:          }
862:         else if (IICModus==IICNONE)
863:          {
864:            PointIIC=&IICNONETable[0];
865:          }
866:         SendIICadress();
867:         ClearIIC();
868:         while (*PointIIC!=NULL)
869:          {
870:            SendIICbyte(*PointIIC);
871:            PointIIC++;
872:          }
873:         if(IICModus!=IICNONE)
874:          {
875:            for (i=1;i<=Max_Char;i++)
876:             {
877:               SendIICbyte(*pointtofloattoascii);
878:               pointtofloattoascii++;
879:             }
880:          }
881:         MainStateMachine=PWMNone;
882:       }
883:
884:
885:
886: //*****************************************************************************
887: // end of main.h
888:
```

```
1: //************************************************************
2: //*                                                          *
3: //* ---------------------------------------------------------- *
4: //*  Filename: pll.h                                         *
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: #ifndef mc9s12dp512_h
22: #include "mc9s12dp512.h"
23: #endif
24:
25:
26: // function declarations
27:
28: // ------------------------------------------------------------------------------
29: // Short description:
30: // this sets the PLL Phase Lock loop to 48 Mhz
31: // This is part of the CRG module
32:
33: // pre:  PLL harware registers must be defined (mc9s12dp512.h)
34: // post: Busclock is set to 48 Mhz
35: // ------------------------------------------------------------------------------
36: void InitPLL48(void);
37:
38: // ------------------------------------------------------------------------------
39: // Short description:
40: // this sets the PLL Phase Lock loop to 50 Mhz
41:
42: // pre:  PLL harware registers must be defined (mc9s12dp512.h)
43: // post: Busclock is set to 50 Mhz (Max Bus Speed)
44: // ------------------------------------------------------------------------------
45: // void InitPLL50(void);
46:
47: //*****************************************************************************
48: // end of PLL.h
49:
```

```
 1: //************************************************************
 2: //*                                                          *
 3: //* ---------------------------------------------------------- *
 4: //*  Filename: Port_b.h                                       *
 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: #ifndef mc9s12dp512_h
22: #include "mc9s12dp512.h"
23: #endif
24:
25: // function declarations
26:
27: // ------------------------------------------------------------------------
28: // Short description:
29: // this sets Port B as an output and gives it the value: 11111111
30:
31: // pre:  PortB harware registers must be defined (mc9s12dp512.h)
32: // post: PortB is set as output and is given the value 0xFF
33: //       On the HCS12 TBoard, PortB is connected to a Led Array
34: // ------------------------------------------------------------------------
35: void InitPortB(void);
36:
37: //************************************************************************
38: // end of Port_b.c
39:
```

```
1: //************************************************************
2: //*                                                          *
3: //* ---------------------------------------------------------- *
4: //*  Filename: pwm.h                                         *
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: #ifndef mc9s12dp512_h
22: #include "mc9s12dp512.h"
23: #endif
24:
25:
26: // function declarations
27:
28: // ------------------------------------------------------------------------------
29: // Short description:
30: // this function initialises all the PWM registers  to a freqency of 31,25 Khz
31: // and 256 steps
32:
33: // pre:  PWM harware registers must be defined (mc9s12dp512.h)
34: // post: PWM registers are set:
35: //         - polarity is positive (0x00)
36: //         - Clock is Clock a for PWM0..PWM3 and Clock is clock b for PWM4..PWM7
37: //         - Clock prescaler is set to 128 for clock a and b
38: //         - PWM are all left aligned
39: //         - 8 x 8 bit clock (No 16 bit clock)
40: //         - PWM Periods are set to 256 steps for all clocks
41: // ------------------------------------------------------------------------------
42: void InitPWM(void);
43:
44: //*******************************************************************************
45: // end of pwm.h
46:
```

```
 1: //************************************************************
 2: //*                                                          *
 3: //* ---------------------------------------------------------- *
 4: //*  Filename: RTI.h                                         *
 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: #ifndef mc9s12dp512_h
22: #include "mc9s12dp512.h"
23: #endif
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: // ------------------------------------------------------------------------------
37: void InitRTI(void);
38:
39:
40: void StopRTI(void);
41:
42: //*******************************************************************************
43: // end of RTI.h
```

```
 1: //************************************************************
 2: //*                                                          *
 3: //* ---------------------------------------------------------- *
 4: //*  Filename: sci.h                                         *
 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: #ifndef mc9s12dp512_h
22: #include "mc9s12dp512.h"
23: #endif
24:
25: #include "asm.h"
26:
27: // defines bitfunction
28: // defines TDRE as 01000000
29:
30: #ifndef bit
31: #define bit(x) (1<<(x))
32: #endif
33:
34: #ifndef TDRE
35: #define TDRE  bit(7)
36: #endif
37:
38: // function declarations
39:
40: // ------------------------------------------------------------------------
41: // Short description:
42: // this initialises de SCI1 Port to interupt on receive
43:
44: // pre:  SCI harware registers must be defined (mc9s12dp512.h)
45: // post: RTI registers are set:
46: //       - Baudrate is set  to 19200 bit/s
47: //       - Receiver interupt, receiver enable and transmitter enable are set
48: //       - Interrupt flag is cleared (OPTIONAL)
49: // ------------------------------------------------------------------------
50: void InitSCI1(void);
51:
52: // ------------------------------------------------------------------------
53: // Short description:
54: // this function sends a character through SCI1
55:
56: // pre:  SCI harware registers must be defined (mc9s12dp512.h)
57: //       SCI registers must have been set
58: //       TDRE (bit7) must have been declared
59: // post: sends character X to SCI1
60: // ------------------------------------------------------------------------
```

```
61: void SendChar(char X);
62:
63: // int read out
64: char Read_High(int x);
65:
66: char Read_Low(int x);
67:
68: void ClearReceive(void);
69:
70:
71:
72: //***************************************************************************
73: // end of SCI.h
```

```
1: //************************************************************
2: //*                                                          *
3: //* ---------------------------------------------------------- *
4: //*  Filename: timer.h                                          *
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: #ifndef mc9s12dp512_h
20: #include "mc9s12dp512.h"
21: #endif
22:
23: void Timer_ON (void);
24:
25: void Timer_OFF (void);
26:
27: char Timer_Read_Low(int);
28:
29: int Get_Time(void);
30:
31: char Timer_Read_High(int);
32:
33:
```