

## DataLogger.c

```

1 // -----
2 //
3 // Copyright © 2011, Intel Corporation
4 //
5 // All rights reserved.
6 //
7 // Redistribution and use in source and binary forms, with or without modification, are permitted
8 // provided that the following conditions are met:
9 //
10 // * Redistributions of source code must retain the above copyright notice, this list of conditions
11 // and the following disclaimer.
12 // * Redistributions in binary form must reproduce the above copyright notice, this list of
13 // conditions and the following disclaimer in the documentation and/or other materials provided
14 // with the distribution.
15 // * Neither the name of Intel Corporation nor the names of its contributors may be used to endorse
16 // or promote products derived from this software without specific prior written permission.
17 //
18 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR
19 // IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND
20 // FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
21 // CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
22 // CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
23 // SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
24 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
25 // OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
26 // POSSIBILITY OF SUCH DAMAGE.
27 //
28 // -----
29 //
30 // Project      : Intel® Leibniz Challenge 2011
31 // File name    : DataLogger.c
32 // Author       : cwille
33 // Date        : 2011-01-06
34 // Revision    : 1.00
35 //
36 // Description  : Firmware for the ATmega128V to be used as data logger.
37 //                Recorded data will be stored in the internal EEPROM and later transferred to the
38 //                PC via Morse code.
39 //
40 // Remark:      : Coding scheme for Morse code taken from
41 //                http://en.wikipedia.org/wiki/Morse\_code
42 //
43 // -----
44 //
45 //
46 #include <avr/interrupt.h>
47 #include <avr/io.h>
48 #include <avr/pgmspace.h>
49 #include <stdbool.h>
50 #include <stdio.h>
51 #include "../inc/global.h"
52 #include "../inc/config.h"
53 #include "../inc/ADC.h"
54 #include "../inc/DataLogger.h"
55 #include "../inc/EEPROM.h"
56
57
58 // Set the controller fuses in the source code to avoid wrong settings.
59 FUSES =
60 {
61     .low      = 0x62,          // CKDIV8
62     .high     = 0x07,          // EESAVE
63     .extended = 0xFF,         // BOOT FLASH @ $1F80 w/ 128 words
64 };
65
66
67 // The three conditions of the output pins for the piezo buzzer.
68 enum ePIEZO
69 {
70     P_OFF,          // Both pins low
71     P_H,            // Pin A high, B low
72     P_L,            // Pin A low, B high
73 };
74
75

```

Intel® Leibniz Challenge 2011, Software „DataLogger“ für Aufgabe 2 und 4

```

76 // The different states of the Morse code output state machine.
77 enum eMSTATE
78 {
79     MS_IDLE,                // Wait for the next character to send.
80     MS_DITDAH,             // Issue a dit or dah.
81     MS_SYMBOL_GAP,        // Wait for a dit gap in the symbol to be finished.
82     MS_LETTER_GAP,        // Wait for a dah gap after a letter to be finished.
83     MS_WORD_GAP           // Wait for seven dits gap after a word to be finished.
84 };
85
86
87 // Morse alphabet
88 // Coded into two bytes per character:
89 // - Code: 0 = Dah, 1 = Dit, right aligned
90 // - Odd byte is code
91 // - Even byte is length of previous code byte
92 // - The character range goes from ASCII 32 (space) to 95 (underscore).
93 // - 0x00, 0 means, there is no Morse code equivalent. The word gap will be
94 //   send instead of this character.
95 // e.g. the ' (apostrophe) is .----. and is converted to 0x21, 6.
96 tUINT8 MorseAlphabet [128] PROGMEM =
97 {
98     0x00, 0, /* Space */
99     0x14, 6, /* ! */ 0x2D, 6, /* " */ 0x00, 0, /* # */ 0x76, 7, /* $ */ 0x00, 0, /* % */
100    0x17, 5, /* & */ 0x21, 6, /* ' */ 0x09, 5, /* ( */ 0x12, 6, /* ) */ 0x00, 0, /* * */
101    0x15, 5, /* + */ 0x0C, 6, /* , */ 0x1E, 6, /* - */ 0x2A, 6, /* . */ 0x0D, 5, /* / */
102    0x00, 5, /* 0 */ 0x10, 5, /* 1 */ 0x18, 5, /* 2 */ 0x1C, 5, /* 3 */ 0x1E, 5, /* 4 */
103    0x1F, 5, /* 5 */ 0x0F, 5, /* 6 */ 0x07, 5, /* 7 */ 0x03, 5, /* 8 */ 0x01, 5, /* 9 */
104    0x07, 6, /* : */ 0x15, 6, /* ; */
105    0x0A, 5, /* < Message Start CT */
106    0x0E, 5, /* = */
107    0x15, 5, /* > Message End AR */
108    0x33, 6, /* ? */ 0x25, 6, /* @ */ 0x02, 2, /* A */ 0x07, 4, /* B */ 0x05, 4, /* C */
109    0x03, 3, /* D */ 0x01, 1, /* E */ 0x0D, 4, /* F */ 0x01, 3, /* G */ 0x0F, 4, /* H */
110    0x03, 2, /* I */ 0x08, 4, /* J */ 0x02, 3, /* K */ 0x0B, 4, /* L */ 0x00, 2, /* M */
111    0x01, 2, /* N */ 0x00, 3, /* O */ 0x09, 4, /* P */ 0x02, 4, /* Q */ 0x05, 3, /* R */
112    0x07, 3, /* S */ 0x00, 1, /* T */ 0x06, 3, /* U */ 0x0E, 4, /* V */ 0x04, 3, /* W */
113    0x06, 4, /* X */ 0x04, 4, /* Y */ 0x03, 4, /* Z */ 0x00, 0, /* [ */ 0x00, 0, /* \ */
114    0x00, 0, /* ] */ 0x00, 0, /* ^ */ 0x32, 6, /* _ */
115 };
116
117
118 tVUINT8 MCode = 0;           // Morse code of the next letter to send
119 tVUINT8 MLength = 0;        // Length of the code.
120 tVUINT8 MStart = 0;         // Sending will start in next interrupt when != 0
121
122 tVUINT8 Button = 0;         // Debounced pushbutton/jumper information
123
124 tUINT16 M_dit;              // Dot
125 tUINT16 M_dah;              // Dash
126 tUINT16 M_symbol_gap;      // Gap between dots and dashes of a symbol, corrected
127 tUINT16 M_letter_gap;      // Gap between letters of a word (3 dit), corrected
128 tUINT16 M_word_gap;        // Gap between words (7 dit), corrected
129 tUINT16 M_space;           // Full 7 dit long gap for the second space and more.
130
131 tUINT8 Task;                // Either 2 or 4 for the ILC task to perform.
132 bool Single;                // True == single measurement only, else 12h.
133
134 tVBOOL ResetCounter;        // Flag: true == reset Counter in next timer ISR
135 tVUINT16 Counter;           // 16-bit counter, incremented every 10ms counts up to 655s
136
137
138 // -----
139
140
141 // Timer 0 overflow interrupt service routine.
142 // Called every 500µs to generate a 1000 Hz tone.
143 // To be precise: At exactly 1 MHz CPU clock, it will be called every 512µs and 977 Hz tone will be
144 // generated. That's an error of -2.4%. The internal generated CPU clock has a tolerance of
145 // +1.25.. -8.75%. That makes a range of 505.6.. 556.8µs or 989.. 898 Hz.
146
147
148 ISR (TIMER0_OVF_vect)
149 {
150     static enum ePIEZO s_piezo = P_OFF;           // Piezo buzzer state machine
151     static enum eMSTATE s_morse = MS_IDLE;        // Morse code output state machine
152     static tUINT16 counter = 0;                    // Countdown counter

```

```

153 static tUINT8 code = 0; // Code to send
154 static tUINT8 length = 0; // Length of the code to send
155 static tUINT8 last_space = 0;
156
157 static tUINT8 button_new = 0;
158 static tUINT8 button_old = 0;
159 static tUINT8 button_count = 0;
160 static tUINT8 prescaler = 1;
161
162 // Reload the timer with -2
163 TCNT0 = 0xFE; // == (tUINT8) (tINT16)(-(F_CPU / 256 * 500e-6 + 0.5));
164
165 // State machine for Morse code
166 switch (s_morse)
167 {
168     case MS_IDLE: // Wait for the next character to send.
169         if (MStart > 0)
170         {
171             code = MCode; // Copy letter information from main program
172             length = MLength;
173             if (length != 0)
174             {
175                 // Length is greater zero -> this is a defined character
176                 code <<= (8 - length); // Left align the letter
177                 counter = (code & 0x80) ? M_dit : M_dah; // Load the correct pulse length
178                 length --; // One down
179                 code <<= 1;
180                 last_space = 0; // No space character this time
181                 s_morse = MS_DITDAH; // Next time, send the dit or dah
182             }
183             else
184             {
185                 // Length is zero -> this is a word gap
186                 counter = (last_space == 0) ? M_word_gap : M_space;
187                 s_morse = MS_WORD_GAP;
188                 last_space = 1; // Remember the word gap.
189             }
190         }
191         break;
192     case MS_DITDAH: // Issue a dit or dah.
193         switch (s_piezo)
194         {
195             case P_H:
196                 PORTB = (PORTB & ~PIEZO_msk) | PIEZO_1; // PIEZO_A = 0, PIEZO_B = 1
197                 s_piezo = P_L;
198                 break;
199             case P_L:
200             default:
201                 PORTB = (PORTB & ~PIEZO_msk) | PIEZO_0; // PIEZO_A = 1, PIEZO_B = 0
202                 s_piezo = P_H;
203                 break;
204         } // switch (s_piezo)
205         counter --;
206         if (counter == 0) // Are we done?
207         {
208             PORTB = PORTB & ~PIEZO_msk; // Turn off the output stage.
209             s_piezo = P_OFF;
210             counter = M_dit; // Yes! Pause for one dit.
211             s_morse = MS_SYMBOL_GAP;
212         }
213         break;
214     case MS_SYMBOL_GAP: // Wait for a dit gap to be finished.
215         counter --;
216         if (counter == 0)
217         {
218             if (length > 0)
219             {
220                 counter = (code & 0x80) ? M_dit : M_dah; // Load the correct pulse length
221                 code <<= 1;
222                 length --; // One down
223                 s_morse = MS_DITDAH; // Send the next dit or dah
224             }
225             else
226             {
227                 counter = M_letter_gap; // Letter is done, make a longer break
228                 s_morse = MS_LETTER_GAP;
229             }
230         }
231     }
232 }

```

```

230     }
231     break;
232     case MS_LETTER_GAP:                               // Wait for a dah gap to be finished.
233     counter --;
234     if (counter == 0)
235     {
236         MStart = 0;                                  // Signal to main program: Ready for next
237         s_morse = MS_IDLE;                            // character!
238     }
239     break;
240     case MS_WORD_GAP:                                 // Wait for seven dits gap to be finished.
241     counter --;
242     if (counter == 0)
243     {
244         MStart = 0;                                  // Signal to main program: Ready for next
245         s_morse = MS_IDLE;                            // character!
246     }
247     break;
248     default:                                          // This code should never be reached
249     s_morse = MS_IDLE;                                // during normal execution. But just in
250     MStart = 0;                                       // case: Reset the state machine.
251     break;
252 } // switch (s_morse)
253
254 // Prescaler to run this part of software only every 10ms.
255 prescaler --;
256 if (prescaler == 0)
257 {
258     prescaler = 20;
259
260     // Debouncer for pushbutton/mode switch input
261     button_new = ~PIND & BUTTON_msk;                 // Only the button pins and positive logic.
262     if (button_new == button_old)
263     {
264         // Fresh sampled buttons are the same as before
265         if (button_count < 3)
266             // Wait one more time before declaring the sample stable/valid.
267             button_count ++;
268         else if (button_count == 3)
269             // Report the stable/valid sample after 40ms to the main program loop.
270             Button = button_new;
271     }
272     else
273     {
274         // Last sample and this one do not match.
275         button_old = button_new;
276         button_count = 0;
277     }
278
279     // Run the 10ms counter for the measurement module.
280     if (ResetCounter == true)
281     {
282         // Main program has requested a reset.
283         Counter = 0;
284         ResetCounter = false;
285     }
286     else
287         if (Counter < 65535)                          // No overflow permitted!
288             Counter ++;
289 }
290 }
291
292
293 // -----
294
295
296 // Replacement for the build in putchar () function.
297 // Sends one character via Morse code.
298 //
299 // Argument(s): ch          - character to send, range is 32..95
300 //
301 // Return:      0
302
303 static int morse_putchar (char ch, FILE *stream)
304 {
305     tUINT8 index;
306

```

```

307 index = ch;
308 // Make some useful indexcharacters of out of range data.
309 if (index == '\n') index = ';'; // Turn linefeed into semicolon
310 if (index < 32) index = 32; // All non print-able characters to spaces
311 if ((index > 95) && (index < 122)) index -= 32; // lowercase -> UPPERCASE
312 if (index > 121) index = 32; // Turn all characters beyond 'z' into spaces.
313
314 // Compute into an index for the big array
315 index -= 32;
316 index <<= 1;
317
318 // Store the Morse code into the volatile variables
319 MCode = pgm_read_byte (&(MorseAlphabet [index]));
320 MLength = pgm_read_byte (&(MorseAlphabet [index + 1]));
321
322 // Start the transmission and wait until its completion.
323 MStart = 1;
324 while (MStart == 1);
325 return 0;
326 }
327
328
329 // Set up a static file handler for the morse_putchar function.
330 static FILE morse_stdout = FDEV_SETUP_STREAM (morse_putchar, NULL, _FDEV_SETUP_WRITE);
331
332
333 // -----
334
335
336 // Read the jumper and set the program parameter accordingly.
337 // There is no need to debounce these inputs.
338 //
339 // Argument(s): -
340 //
341 // Return: -
342
343 void ReadJumper (void)
344 {
345 // Calculate morse speed.
346 if ((PIND & JUMPER_SPEED) == JUMPER_SPEED)
347 M_dit = DIT_FAST;
348 else
349 // For slow mode, take jumper settings into account.
350 M_dit = DIT_SLOW + 25 * (~PINC & JUMPER_SPEED_msk);
351
352 M_dah = DAH;
353 M_symbol_gap = SYMBOL_GAP;
354 M_letter_gap = LETTER_GAP;
355 M_word_gap = WORD_GAP;
356 M_space = SPACE_GAP;
357 // Determine the current task to use the correct logger functions.
358 Task = ((PIND & JUMPER_TASK) == JUMPER_TASK) ? 4 : 2;
359 Single = ((PIND & JUMPER_SINGLE) == JUMPER_SINGLE);
360 }
361
362
363 // Main program function
364 //
365 // Argument(s): -
366 //
367 // Return: -
368
369 int main (void)
370 {
371 ResetCounter = false;
372 Counter = 0;
373
374 // Configure port pins
375 DDRB = LED | PIEZO_A | PIEZO_B;
376 PORTB = LED | PIEZO_OFF;
377
378 DDRC = 0; // Inputs only with pullup resistors
379 PORTC = JUMPER_SPEED_msk;
380
381 DDRD = TENDENCY_HI | TENDENCY_LO;
382 PORTD = BUTTON_msk | JUMPER_msk;
383

```

```

384 // Timer 0 with 256 prescaler
385 TCCR0B = (1 << CS02);
386 TIMSK0 |= 1 << TOIE0;
387
388 // Enable global interrupts;
389 sei ();
390
391 // Redirect printf output to morse code generator
392 stdout = &morse_stdout;
393
394 // Read jumper settings to be able to send the correct greeting.
395 ReadJumper ();
396
397 // Initialize ADC register.
398 ADCinit ();
399
400 // Send a greeting message before doing anything else.
401 printf_P (PSTR (<welcome to ILC 2011, task %d>"), Task);
402
403 while (1)
404 {
405     // Re-read the jumper settings, just in case the user has changed his/her mind.
406     ReadJumper ();
407
408     switch (Button)
409     {
410     case BUTTON_INIT:
411         // Initialize EEPROM in task-two-mode
412         PORTB &= ~LED;
413         EEP_Init ();
414         // Wait until no button is down.
415         while (Button);
416         PORTB |= LED;
417         break;
418     case BUTTON_START:
419         // Start measurement cycle
420         PORTB &= ~LED;
421         ADctakeMeasurement ();
422         // Wait until no button is down.
423         while (Button);
424         PORTB |= LED;
425         break;
426     case BUTTON_PLAY:
427         // Play data from EEPROM via morse code
428         PORTB &= ~LED;
429         EEP_ReplayData ();
430         // Wait until no button is down.
431         while (Button);
432         PORTB |= LED;
433         break;
434     default:
435         break;
436     }
437 }
438 }

```

## DataLogger.h

```

1 // -----
2 //
3 // Copyright © 2010-2011, Intel Corporation
4 //
5 // All rights reserved.
6 //
7 // Redistribution and use in source and binary forms, with or without modification, are permitted
8 // provided that the following conditions are met:
9 //
10 // * Redistributions of source code must retain the above copyright notice, this list of conditions
11 // and the following disclaimer.
12 // * Redistributions in binary form must reproduce the above copyright notice, this list of
13 // conditions and the following disclaimer in the documentation and/or other materials provided
14 // with the distribution.
15 // * Neither the name of Intel Corporation nor the names of its contributors may be used to endorse
16 // or promote products derived from this software without specific prior written permission.
17 //
18 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR
19 // IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND
20 // FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
21 // CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
22 // CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
23 // SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
24 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
25 // OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
26 // POSSIBILITY OF SUCH DAMAGE.
27 //
28 // -----
29 //
30 // Project      : Intel® Leibniz Challenge 2011
31 // File name    : DataLogger.h
32 // Author       : cwille
33 // Date        : 2010-11-15
34 // Revision    : 1.00
35 //
36 // Description  : Header file for ADC module.
37 //
38 // -----
39 //
40 //
41 #ifndef __DATALOGGER_H__
42 #define __DATALOGGER_H__
43 //
44 //
45 extern tVBOOL   ResetCounter;    // Flag: true == reset Counter in next timer ISR
46 extern tVUINT16 Counter;        // 16-bit counter, incremented every 10ms counts up to 655s
47 //
48 extern tVUINT8  Button;         // Debounced pushbutton/jumper information
49 extern tUINT8   Task;           // Either 2 or 4 for the ILC task to perform.
50 extern bool     Single;         // True == single measurement only, else 12h.
51 //
52 //
53 #endif // __DATALOGGER_H__

```

## ADC.c

```

1 // -----
2 //
3 // Copyright © 2011, Intel Corporation
4 //
5 // All rights reserved.
6 //
7 // Redistribution and use in source and binary forms, with or without modification, are permitted
8 // provided that the following conditions are met:
9 //
10 // * Redistributions of source code must retain the above copyright notice, this list of conditions
11 // and the following disclaimer.
12 // * Redistributions in binary form must reproduce the above copyright notice, this list of
13 // conditions and the following disclaimer in the documentation and/or other materials provided
14 // with the distribution.
15 // * Neither the name of Intel Corporation nor the names of its contributors may be used to endorse
16 // or promote products derived from this software without specific prior written permission.
17 //
18 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR
19 // IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND
20 // FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
21 // CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
22 // CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
23 // SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
24 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
25 // OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
26 // POSSIBILITY OF SUCH DAMAGE.
27 //
28 // -----
29 //
30 // Project      : Intel® Leibniz Challenge 2011
31 // File name    : ADC.c
32 // Author       : cwille
33 // Date        : 2011-01-06
34 // Revision    : 1.00
35 //
36 // Description  : ADC module.
37 //
38 // Remark:     : In task-four-mode, samples will be taken every 10 seconds. Every three minutes, the
39 //               average will be computed and the result will be stored into the EEPROM.
40 //               After 12 hours, a full cycle will be completed.
41 //
42 // -----
43 //
44
45 #include <avr/interrupt.h>
46 #include <avr/io.h>
47 #include <avr/pgmspace.h>
48 #include <stdbool.h>
49 #include <stdio.h>
50 #include "../inc/global.h"
51 #include "../inc/config.h"
52 #include "../inc/DataLogger.h"
53 #include "../inc/EEPROM.h"
54 #include "../inc/ADC.h"
55
56
57 tUINT16 LastADCval;                                // Keeps the result of the last measurement.
58
59
60 // Show Tendency of measurement on two port pins.
61 //
62 // Argument(s): NewADCval      - Next 10 bit conversion result.
63 //
64 // Return:      -
65
66 void ADC_ShowTendency (tUINT16 NewADCval)
67 {
68     if (NewADCval == LastADCval)
69     {
70         // Equal
71         PORTD &= ~(TENDENCY_HI | TENDENCY_LO);
72     }
73     else if (NewADCval > LastADCval)
74     {
75         // Higher

```

```

76     PORTD &= ~TENDENCY_LO;
77     PORTD |= TENDENCY_HI;
78 }
79 else
80 {
81     // Lower
82     PORTD &= ~TENDENCY_HI;
83     PORTD |= TENDENCY_LO;
84 }
85 LastADCval = NewADCval;
86 }
87
88
89 // Central data logger routine to convert and store measurements.
90 //
91 // Argument(s): -
92 //
93 // Return:      -
94
95 void ADCtakeMeasurement (void)
96 {
97     tUINT16 i;
98     tUINT16 ADC3min;           // Average over 3 minutes.
99     tUINT32 ADCsum;           // Sum of all conversions for next average.
100    bool Full;                // True if EEPROM storage has been used up.
101
102    if (Single)                // Will be same for both tasks.
103    {
104        // Take one shot, play the result and exit.
105        ADCsum = 0;
106        for (i = 0; i < 32; i++)
107        {
108            ADCSRA |= (1 << ADSC);           // Start conversion.
109            while (ADCSRA & (1 << ADSC));    // Wait until conversion is complete.
110            ADCsum += ADCW;                   // Sum up the results.
111        }
112        ADCsum >>= 5;                         // Divide by 32
113        printf_P (PSTR (<S %d>"), ADCsum);
114        ADC_ShowTendency (ADCsum);
115    }
116    else
117    {
118        if (Task == 2)                // Twelve hour cycle for task 2 with one
119        {                               // measurement per hour.
120            // Record data.
121            Full = (EEP_ReturnIndex () > 11); // Twelve result will do for task 2.
122
123            if (Full)
124                // Send error code and exit function.
125                printf_P (PSTR ("HH"));
126
127            while (!Full)              // As long as there is some space in the EEPROM..
128            {
129                // Do 360 samples and store them in the EEPROM.
130                ADCsum = 0;
131                ADC3min = 0;
132                for (i = 0; i < 360; i++)
133                {
134                    ADCSRA |= (1 << ADSC);           // Start conversion.
135                    while (ADCSRA & (1 << ADSC));    // Wait until conversion is complete.
136                    ADC3min += ADCW;                 // Read newest conversion result and sum it up.
137                    printf_P (PSTR ("e"));           // Send a short "I'm still alive!"
138                    if (((i + 1) % 18) == 0)
139                    {
140                        // 3 minutes period is over.
141                        ADCsum += ADC3min;           // Add to 1 hour sum.
142                        ADC3min /= 18;               // Calculate 3 minutes average.
143                        ADC_ShowTendency (ADC3min); // Show tendency.
144                        ADC3min = 0;
145                    }
146                }
147                // Wait 10 seconds before the next conversion.
148                ResetCounter = true;                // Request a counter reset.
149                while (ResetCounter == true);       // Wait until the counter has been reset by the
150                                                    // timer interrupt routine.
151            }
152        }

```

Intel® Leibniz Challenge 2011, Software „DataLogger“ für Aufgabe 2 und 4

```

153         cli (); // Disable interrupts to ensure that checking of
154         if (Counter > 999) // the variable counter will not be interrupted.
155             break; // (Atomic code execution.)
156         sei (); // Re-enable interrupts after atomic check.
157     }
158     sei (); // Re-enable after last check.
159 }
160 ADCsum /= 360; // Compute the average over 360 measurements.
161 Full = EEP_StoreData (ADCsum) | // Store result in the controller's EEPROM.
162 (EEP_ReturnIndex () > 11);
163 }
164 }
165 else // Twelve hour cycle for task 4 with full
166 { // resolution.
167     // Record some more data.
168     Full = EEP_Full ();
169
170     if (Full)
171         // Send error code and exit function.
172         printf_P (PSTR ("HH"));
173
174     while (!Full) // As long as there is some space in the EEPROM..
175     {
176         // Do 18 samples and store them in the EEPROM.
177         ADCsum = 0;
178         for (i = 0; i < 18; i ++)
179         {
180             ADCSRA |= (1 << ADSC); // Start conversion.
181             while (ADCSRA & (1 << ADSC)); // Wait until conversion is complete.
182             ADCsum += ADCW; // Sum up the results.
183             printf_P (PSTR ("e")); // Send a short "I'm still alive!"
184
185
186             // Wait 10 seconds before the next conversion.
187             ResetCounter = true; // Request a counter reset.
188             while (ResetCounter == true); // Wait until the counter has been reset by the
189             // timer interrupt routine.
190
191             while (1)
192             {
193                 cli (); // Disable interrupts to ensure that checking of
194                 if (Counter > 999) // the variable counter will not be interrupted.
195                     break; // (Atomic code execution.)
196                 sei (); // Re-enable interrupts after atomic check.
197             }
198             sei (); // Re-enable after last check.
199             ADCsum /= 18; // Compute the average over 18 measurements.
200             Full = EEP_StoreData (ADCsum); // Store result in the controller's EEPROM.
201             ADC_ShowTendency (ADCsum); // Show tendency.
202         }
203     }
204 }
205 }
206
207
208 // Initialize ADC with internal 1.1V reference and select channel 0 for conversion.
209
210 void ADCinit (void)
211 {
212     tUINT16 Dummy;
213
214     // Disable digital stage for ADC pin.
215     DIDR0 = ADC_CH0;
216
217     // Select internal 1.1V reference, result right aligned and channel 0.
218     ADMUX = (1 << REFS1) | (1 << REFS0);
219
220     // F_CPU / 16 = 62,5 kHz sample rate.
221     ADCSRA = (1 << ADPS2);
222     ADCSRA |= (1 << ADEN);
223
224     // Trigger one dummy conversion to finally initialize the ADC
225     ADCSRA |= (1 << ADSC);
226     while (ADCSRA & (1 << ADSC));
227     Dummy = ADCW;
228 }

```

## ADC.h

```

1 // -----
2 //
3 // Copyright © 2011, Intel Corporation
4 //
5 // All rights reserved.
6 //
7 // Redistribution and use in source and binary forms, with or without modification, are permitted
8 // provided that the following conditions are met:
9 //
10 // * Redistributions of source code must retain the above copyright notice, this list of conditions
11 // and the following disclaimer.
12 // * Redistributions in binary form must reproduce the above copyright notice, this list of
13 // conditions and the following disclaimer in the documentation and/or other materials provided
14 // with the distribution.
15 // * Neither the name of Intel Corporation nor the names of its contributors may be used to endorse
16 // or promote products derived from this software without specific prior written permission.
17 //
18 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR
19 // IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND
20 // FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
21 // CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
22 // CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
23 // SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
24 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
25 // OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
26 // POSSIBILITY OF SUCH DAMAGE.
27 //
28 // -----
29 //
30 // Project      : Intel® Leibniz Challenge 2011
31 // File name    : ADC.h
32 // Author       : cwille
33 // Date        : 2011-01-06
34 // Revision     : 1.00
35 //
36 // Description  : Header file for ADC module.
37 //
38 // -----
39 //
40 //
41 #ifndef __ADC_H__
42 #define __ADC_H__
43 //
44 //
45 // Run measurement cycle according to the jumper settings.
46 void ADCtakeMeasurement (void);
47 //
48 // Initialize ADC after power on.
49 void ADCinit ();
50 //
51 //
52 #endif // __ADC_H__

```

## EEPROM.c

```

1 // -----
2 //
3 // Copyright © 2011, Intel Corporation
4 //
5 // All rights reserved.
6 //
7 // Redistribution and use in source and binary forms, with or without modification, are permitted
8 // provided that the following conditions are met:
9 //
10 // * Redistributions of source code must retain the above copyright notice, this list of conditions
11 // and the following disclaimer.
12 // * Redistributions in binary form must reproduce the above copyright notice, this list of
13 // conditions and the following disclaimer in the documentation and/or other materials provided
14 // with the distribution.
15 // * Neither the name of Intel Corporation nor the names of its contributors may be used to endorse
16 // or promote products derived from this software without specific prior written permission.
17 //
18 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR
19 // IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND
20 // FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
21 // CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
22 // CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
23 // SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
24 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
25 // OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
26 // POSSIBILITY OF SUCH DAMAGE.
27 //
28 // -----
29 //
30 // Project      : Intel® Leibniz Challenge 2011
31 // File name    : EEPROM.c
32 // Author       : cwille
33 // Date        : 2011-01-06
34 // Revision     : 1.00
35 //
36 // Description  : Access rotuines for internal EEPROM.
37 //
38 // -----
39
40
41 #include <avr/eeprom.h>
42 #include <avr/pgmspace.h>
43 #include <stdbool.h>
44 #include <stdio.h>
45 #include "../inc/global.h"
46 #include "../inc/DataLogger.h"
47 #include "../inc/EEPROM.h"
48
49
50 // EEPROM memory map:
51 // -----
52 //
53 // addr | purpose
54 // -----+-----
55 // 0 | Not used
56 // 1 | Number of already stored measurements (0..240)
57 // 2 | Task ID (2 or 4)
58 // 4 | 16 bit ADC result 0
59 // 6 |      "-"          1
60 // ... |
61 // 484 |      "-"          239
62
63
64 // Return the actual index for the next measurement.
65 //
66 // Argument(s): -
67 //
68 // Return:      Byte within 0..240 range.
69
70 tUINT8 EEP_ReturnIndex (void)
71 {
72     tUINT8 index;
73
74     index = eeprom_read_byte ((tUINT8 *) 1);
75     if (index > MAX_DATA_WORD)

```

```

76     // Treat unprogrammed EEPROM as a full memory!
77     index = MAX_DATA_WORD;
78
79     return index;
80 }
81
82
83 // Returns true if the storage capacity of the EEPROM is exhausted.
84 //
85 // Argument(s): -
86 //
87 // Return:      True if number >= MAX_DATA_WORD.
88
89 bool EEP_Full (void)
90 {
91     return ((eeprom_read_byte ((tUINT8 *) 1) >= MAX_DATA_WORD));
92 }
93
94
95 // Stores the ADC value into the next free EEPROM location and advances the data pointer.
96 //
97 // Argument(s): ADCvalue      - 16 bit data to store.
98 //
99 // Return:      True if memory is full, else false.
100
101 bool EEP_StoreData (tUINT16 ADCvalue)
102 {
103     tUINT8 index;
104
105     index = eeprom_read_byte ((tUINT8 *) 1);
106     if (index >= MAX_DATA_WORD)
107         // Memory already full, store nothing and exit!
108         return false;
109
110     eeprom_write_word ((tUINT16 *) (index * 2 + 4), ADCvalue);
111     index ++;
112     eeprom_write_byte ((tUINT8 *) 1, index);
113
114     if (index >= MAX_DATA_WORD)
115         // Memory now full!
116         return true;
117     else
118         // Still memory available for new data.
119         return false;
120 }
121
122
123 // Replays the previously recorded data from EEPROM. Sends an error message if EEPROM seems
124 // uninitialized.
125 //
126 // Argument(s): -
127 //
128 // Return:      -
129
130 void EEP_ReplayData (void)
131 {
132     tUINT8 index;
133     tUINT8 task;
134     tUINT8 i;
135
136     index = eeprom_read_byte ((tUINT8 *) 1);
137     task = eeprom_read_byte ((tUINT8 *) 2);
138
139     if (index > MAX_DATA_WORD)
140     {
141         // No data stored!
142         printf_P (PSTR (<No data!>));
143         return;
144     }
145
146     printf_P (PSTR (<ILC 2011, task %d, %d entries>"), task, index);
147     for (i = 0; i < index; i ++)
148     {
149         if ((i % 12) == 0)
150             // Start a line with max. twelve result with '<000'
151             printf_P (PSTR (<%03d"), i);
152

```

Intel® Leibniz Challenge 2011, Software „DataLogger“ für Aufgabe 2 und 4

```
153     // Send a result.
154     printf_P (PSTR ("%04d"), eeprom_read_word ((tUINT16 *) (i * 2 + 4)));
155
156     if (((i + 1) % 12) == 0)
157         // End a line with '>'
158         printf_P (PSTR (">"));
159 }
160
161 if ((i % 12) != 0)
162     // End the last line with less than twelve results with '>'.
163     printf_P (PSTR (">"));
164 }
165
166 // Initializes EEPROM for new recording session.
167 //
168 // Argument(s): -
169 //
170 // Return: -
171
172 void EEP_Init (void)
173 {
174     eeprom_write_byte ((tUINT8 *) 1, 0); // Rewind index to 0.
175     eeprom_write_byte ((tUINT8 *) 2, Task); // Remember current task setting.
176 }
177
```

## EEPROM.h

```

1 // -----
2 //
3 // Copyright © 2011, Intel Corporation
4 //
5 // All rights reserved.
6 //
7 // Redistribution and use in source and binary forms, with or without modification, are permitted
8 // provided that the following conditions are met:
9 //
10 // * Redistributions of source code must retain the above copyright notice, this list of conditions
11 // and the following disclaimer.
12 // * Redistributions in binary form must reproduce the above copyright notice, this list of
13 // conditions and the following disclaimer in the documentation and/or other materials provided
14 // with the distribution.
15 // * Neither the name of Intel Corporation nor the names of its contributors may be used to endorse
16 // or promote products derived from this software without specific prior written permission.
17 //
18 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR
19 // IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND
20 // FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
21 // CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
22 // CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
23 // SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
24 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
25 // OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
26 // POSSIBILITY OF SUCH DAMAGE.
27 //
28 // -----
29 //
30 // Project      : Intel® Leibniz Challenge 2011
31 // File name    : EEPROM.h
32 // Author       : cwille
33 // Date        : 2011-01-06
34 // Revision     : 1.00
35 //
36 // Description  : Access rotuines for internal EEPROM.
37 //
38 // -----
39 //
40 //
41 #ifndef __EEPROM_H__
42 #define __EEPROM_H__
43 //
44 //
45 #define MAX_DATA_WORD          240          // Maximum storage capacity: 240 measurements.
46 //
47 //
48 tUINT8 EEP_ReturnIndex (void);           // Return the actual index for the next measurement.
49 //
50 bool EEP_Full (void);                   // Return true if at least one more meas. can be stored.
51 //
52 bool EEP_StoreData (tUINT16 ADCvalue);   // Stores the given value as next item.
53 //
54 void EEP_ReplayData (void);             // Replay the recorded data from EEPROM.
55 //
56 void EEP_Init (void);                   // Initializes the EEPROM for a new measurement cycle.
57 //
58 //
59 #endif // __EEPROM_H__

```

## global.h

```

1 // -----
2 //
3 // Copyright © 2011, Intel Corporation
4 //
5 // All rights reserved.
6 //
7 // Redistribution and use in source and binary forms, with or without modification, are permitted
8 // provided that the following conditions are met:
9 //
10 // * Redistributions of source code must retain the above copyright notice, this list of conditions
11 // and the following disclaimer.
12 // * Redistributions in binary form must reproduce the above copyright notice, this list of
13 // conditions and the following disclaimer in the documentation and/or other materials provided
14 // with the distribution.
15 // * Neither the name of Intel Corporation nor the names of its contributors may be used to endorse
16 // or promote products derived from this software without specific prior written permission.
17 //
18 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR
19 // IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND
20 // FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
21 // CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
22 // CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
23 // SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
24 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
25 // OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
26 // POSSIBILITY OF SUCH DAMAGE.
27 //
28 // -----
29 //
30 // Project      : Intel® Leibniz Challenge 2011
31 // File name    : global.h
32 // Author       : cwille
33 // Date         : 2011-01-06
34 // Revision     : 1.00
35 //
36 // Description  : Central definition of basic data types.
37 //
38 // -----
39
40
41 #ifndef __GLOBAL_H__
42 #define __GLOBAL_H__
43
44
45 typedef signed char      tINT8;
46 typedef signed int      tINT16;
47 typedef signed long     tINT32;
48 typedef unsigned char   tUINT8;
49 typedef unsigned int    tUINT16;
50 typedef unsigned long   tUINT32;
51 typedef volatile signed char tvINT8;
52 typedef volatile signed int tvINT16;
53 typedef volatile signed long tvINT32;
54 typedef volatile unsigned char tvUINT8;
55 typedef volatile unsigned int tvUINT16;
56 typedef volatile unsigned long tvUINT32;
57 typedef volatile bool    tvBOOL;
58
59
60 #endif // __GLOBAL_H__

```

## config.h

```

1 // -----
2 //
3 // Copyright © 2011, Intel Corporation
4 //
5 // All rights reserved.
6 //
7 // Redistribution and use in source and binary forms, with or without modification, are permitted
8 // provided that the following conditions are met:
9 //
10 // * Redistributions of source code must retain the above copyright notice, this list of conditions
11 // and the following disclaimer.
12 // * Redistributions in binary form must reproduce the above copyright notice, this list of
13 // conditions and the following disclaimer in the documentation and/or other materials provided
14 // with the distribution.
15 // * Neither the name of Intel Corporation nor the names of its contributors may be used to endorse
16 // or promote products derived from this software without specific prior written permission.
17 //
18 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR
19 // IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND
20 // FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
21 // CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
22 // CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
23 // SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
24 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
25 // OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
26 // POSSIBILITY OF SUCH DAMAGE.
27 //
28 // -----
29 //
30 // Project      : Intel® Leibniz Challenge 2011
31 // File name    : config.h
32 // Author       : cwille
33 // Date        : 2011-01-10
34 // Revision    : 1.00
35 //
36 // Description  : Central configuration file for the data logger project.
37 //
38 // -----
39
40
41 #ifndef __CONFIG_H__
42 #define __CONFIG_H__
43
44
45 #define F_CPU      1000000          // 1MHz internal RC-oscillator
46
47
48 // Morse code pulse length definitions related to a 500µs timer interrupt service routine.
49 #define DIT_SLOW   400              // Dot, human readable -> 200ms/dit
50 #define DIT_FAST   100             // Dot, machine readable -> 50ms/dit
51 #define DAH        (M_dit * 3)    // Dash
52 #define SYMBOL_GAP M_dit          // Gap between dots and dashes of a symbol
53 #define LETTER_GAP (M_dah - M_dit) // Gap between letters of a word (3 dit)
54 #define WORD_GAP   (M_dit * 4)    // Gap between words (7 dit)
55 #define SPACE_GAP  (M_dit * 7)    // Full gap, to be used after the word gap.
56
57
58 // Port usage
59
60 // Port B
61 #define LED        (1 << 0)        // Connect anode of LED to port, cathode to GND.
62 #define PIEZO_A    (1 << 1)        // Black cable of the piezo
63 #define PIEZO_B    (1 << 2)        // Red cable of the piezo
64 #define PIEZO_msk  (PIEZO_A | PIEZO_B) // Mask for both bits
65
66 #define PIEZO_0    PIEZO_A         // Output condition 0
67 #define PIEZO_1    PIEZO_B         // Output condition 1
68 #define PIEZO_OFF  0              // Both output low = piezo off
69
70 // Port C
71 #define ADC_CH0    (1 << 0)        // ADC channel 0 analog input 0..1V
72 #define JUMPER_SPEED0 (1 << 3)     // Three slow speed select jumper
73 #define JUMPER_SPEED1 (1 << 4)     // (See circuit diagram for speed table.)
74 #define JUMPER_SPEED2 (1 << 5)
75 #define JUMPER_SPEED_msk (JUMPER_SPEED0 | JUMPER_SPEED1 | JUMPER_SPEED2)

```

```
76
77 // Port D
78 #define BUTTON_INIT (1 << 0) // Initializes (AKA reset) measurement & EEPROM
79 #define BUTTON_START (1 << 1) // Triggers measurement cycle
80 #define BUTTON_PLAY (1 << 2) // Plays the recorded data.
81 #define BUTTON_msk (BUTTON_INIT | BUTTON_START | BUTTON_PLAY) // _msk == mask
82 #define JUMPER_SPEED (1 << 3) // Low or high speed replay
83 #define JUMPER_SINGLE (1 << 4) // High == single or low == 12h measurement cycle
84 #define JUMPER_TASK (1 << 5) // Selects task 2 (low) or task 4 (high) mode
85 #define JUMPER_msk (JUMPER_SPEED | JUMPER_SINGLE | JUMPER_TASK)
86 #define TENDENCY_LO (1 << 6) // 1: Current measurement lower than last one; 0: Equal
87 #define TENDENCY_HI (1 << 7) // 1: Current meas. higher than last one; 0: Equal
88
89
90 #endif // __CONFIG_H__
```