Mikroprozessorpraktikum WS 12/13

Carlos Martín Nieto, Simon Hohberg, Tu Tran

February 23, 2013

11 Der 868MHz Transceiver CC1100

11.1 Der Transceiver CC1100 im Empfangsmode

11.1.1 Stellen Sie auf Basis der bereitgestellten Funktionen den Transceiver folgendermaßen ein:

- Transceiver im Empfangsmode
- ID und Frequenzkanal sind gleich der Nummer ihres Praktikumsarbeitsplatzes (im Bereich von 1..10, steht am Labortisch)

```
1
2  //=============================================================
3  //=== Startprojekt zum Mikroprozessorpraktikum ================
4  //=============================================================
5  #include "msp430x16x.h"      // Systemdefinitionen für den MSP430F1612
6  #include "init.h"            // Initialisierung des Mikrocontrollers
7  #include "system.h"          // Systemfunktionen MSB430H
8  #include "interrupts.h"      // ISR - Interrupt Service Routinen
9
10 #include "HW_CC1100.h"       // CC1100 868MHz Funk Transceiver
11 #include "HW_SHT11.h"        // SHT11 Feuchte- und Temperatursensor
12 #include "HW_MMA7260Q.h"     // MMA7260Q 3D-Beschleunigungssensor
13 #include "HW_LCD_TS.h"       // DOGM128 LCD Display Touchscreen
14 #include "HW_LCD.h"          // DOGM128 LCD Display
15
16 #include "stdio.h"           // Standard Libary
17 #include "string.h"          // Standard Libary
18 #include "math.h"            // Standard Libary
19
20 //#include "aufgabe_x.h"     // Header Datei für aufgabe_x.c
21                              // möglichst eigene xxx.h und xxx.c Dateien
22                              // erstellen und verwenden.
23 main(void);                  //Hauptprogramm
24 void print_value(void);
25
26
27 #define BIT_SET(a,b) ((a) |= (b))
28 #define BIT_CLR(a,b) ((a) &= ~(b))
29 #define BIT_TOGGLE(a,b) ((a) ^= (b))
30
31 //====Hauptprogramm============================================
32
33
34 main(void)
35     {
36
37 //===Hier sollten Variablen deklariert werden ================
38     //unsigned char i = 0;
39     //char text[60];
40     //int x,y;
41
42 //===Hier die notwendigen Initialisierungsschritte ===========
43 //=(1)== Port-Initialisierung ================================
44     init_Port();                 // Initialisierung der Port Register
```

```
45
46  //(2)=== Clock-System-Initialisierung ================================
47      //== XT2() oder Dco() als Taktquelle einstellen
48      //== durch Ein- oder Auskommentieren
49      //== DCO ist bei LPM Einsatz bevorzugt muß zyklisch kalibriert werden
50      //== XT2 ist quarzstabil muß nicht zyklisch kalibriert werden
51      //
52      //XT2 ();                       // XT2 Taktquelle aktivieren mit 7.3728MHz
53      DCO ();                     // Dco Taktquelle aktivieren mit 7.3728MHz
54      //                             beachte DELTA
55
56  //=(3)== Timer-Initialisierung= ======================================
57      init_Timer_A();             // Init Timer für Sekundeninterrupt
58                                  // !! noch leere Funktion
59
60  //=(4)== USART-Initialisierung ======================================
61      init_UART1();               // UART-RS232 mit 115.2kBit/s initialisieren
62                                  // !! noch leere Funktion
63
64  //=(5)== CC1100-Transceiver-Initialisierung ==========================
65      init_UART0_SPI();           // CC1100 SPI UART initalisieren
66      init_CC1100();              // CC1100 init und in RX Mode setzen
67                                  // !!!Interrupte sind ab jetzt freigegeben!!
68      //== Adresse und Funkkanal des Transceivers setzen
69      //== für die Arbeitsplaetze HWPx (x=1...10) sollten
70      //== ID=x und channnel=x gesetzt werden
71      ID = 8;                             // Adresse
72      channel = 8;                        // Funkkanal
73      setUid(ID);                         // Adresse im Transceiver setzen
74      switchFreq(channel);                // Funkkanal im Transceiver setzen
75      //== Soll der Transceiver genutzt werden müssen folgende zwei Zeilen
76      //== auskommentiert werden:
77      /*init_CC1100_IDLE();           // CC1100 in den IDLE Mode setzen
78      init_CC1100_POWERDOWN();*/// CC1100 in den PowerDown Mode setzen
79
80  //=(6)== LCD-Display-Initialisierung ================================
81      dogm_reset();           // Hardware Reset des LCD Controllers
82      dogm_init();            // Initialisierung der LCD Controller Register
83      lcd_clear(WHITE);       // Grafikspeicher auf dem MSP430 löschen
84      //lcd_string(BLACK, 15, 25, "MSP430-GESTARTET!");   // Textausgabe
85      lcd_paint();            // Grafikspeicher auf das LCD Display ausgeben
86
87
88  #define LED_ROT (0x01)    // 0 0 1 P4.0
89  #define LED_GELB (0x02)   // 0 1 0 P4.1
90  #define LED_GRUEN (0x04) // 1 0 0 P4.2
91  #define LED_ALL (LED_ROT | LED_GELB | LED_GRUEN)
92
93  #define LED_ON(led) (BIT_CLR(P4OUT, led))
94  #define LED_OFF(led) (BIT_SET(P4OUT, led))
95  #define LED_TOGGLE(led) (BIT_TOGGLE(P4OUT, led))
96
97  #define IS_LED_ON(led) (!(P4OUT & led))
98
99  #define TASTE_LINKS (0x1)
100 #define TASTE_RECHTS (0x2)
101
102 #define SLEEP_QUANTUM 10000
103 #define SLEEP(n) do {       /* sleep for n seconds */ \
104     long time = n * 100000; /* wait() sleeps 10*n microseconds */ \
105     while(time > SLEEP_QUANTUM) { \
106         wait(SLEEP_QUANTUM); \
107         time -= SLEEP_QUANTUM; \
108     } \
109     wait(time); \
110 } while(0)
111
112     LED_OFF(LED_ALL);
113
114     _bis_SR_register(GIE);
115
```

```
116 //===Hier die Endlosschleife quasi das Betriebssystem==================
117
118 while(1){
119 }   // Ende der Endlosschleife
120 }   // Ende Main
121 //====Ende des Hauptprogramms ===================================
122
123 #pragma vector = PORT2_VECTOR   // ISR für den CC1100 Transceiver
124 __interrupt void PORT2 (void) {
125     char res;                          // CRC Check
126     int i = 0;
127     char data[64];
128
129     if (P2IFG & 0x01)   // Check P2IFG Bit P2.0 - CC1100 Rx Packet
130         {
131         CLEAR(P2IFG, 0x01);
132
133         res = receivePacket();  // CRC Rückgabe
134         if (res)                // wenn Paket OK ...
135             {
136                 // kopiere daten
137                 for (i = 2; i < RxCC1100.length; i++)
138                 {
139                     data[i-2] = RxCC1100.data[i];
140                 }
141                 data[i-2] = 0x00;
142
143                 // receiver
144                 printPacket(0); // 0 - Paket auf LCD ausgeben
145                                 // 1 - Paket auf UART1 ausgeben
146
147                 lcd_clear(WHITE);   // Grafikspeicher auf dem MSP430 löschen
148                 lcd_string(BLACK, 2, 2, data);  // Textausgabe
149                 lcd_paint();        // Grafikspeicher auf das LCD Display ausgeben
150             }
151         else
152             {
153                 spiStrobe(CC1100_SIDLE);    // Switch to IDLE
154                 spiStrobe(CC1100_SFRX);     // Flush the RX FIFO
155             }
156         }
157     else
158         {
159         CLEAR(P2IFG, 0xFF);             // Clear all flags
160         }
161     spiStrobe(CC1100_SRX);              // Switch to RX Mode
162 }
```

Zunächst haben wir ID = 8 und Channel = 8 gesetzt. Danach haben wir außerdem ID = 2 und ID = 3 getestet, aber keine Pakete erhalten.

Erhaltene Paket für ID = 8, Channel = 8:

```
1  Quelladresse= 18
2  Zieladresse = 8
3  RSSI = 64
4  Paketlänge = 13
5
6  Datenpaket = Hallo HWP08
7
8  Quelladresse= 18
9  Zieladresse = 8
10 RSSI = 54
11 Paketlänge = 13
12
13 Datenpaket = Hallo HWP08
14
15 Quelladresse= 18
16 Zieladresse = 8
17 RSSI = 54
18 Paketlänge = 13
```

```
19
20 Datenpaket = Hallo HWP08
21
22 Quelladresse= 18
23 Zieladresse = 8
24 RSSI = 59
25 Paketlänge = 13
```

11.2 Der Transceiver CC1100 im Sendemode

In dieser Aufgabe sollen Sensordaten per Funk zwischen zwei Systemen übertragen werden. Entwickeln Sie eine Lösung für zwei MSB430H-Boards, eins als Sender und das andere als Empfänger, die folgende Funktionalität für den entsprechenden MSB430H bereitstellt:

11.2.1 Für den Sender:

- Transceiver in den Empfangsmode setzen
- Initialisierung Funkkanal entsprechend des Arbeitsplatzes
- Initialisierung auf ID = Arbeitsplatznummer + 10
- Per Funk an den Empfänger alle 5 Sekunden Sensordaten übertragen Sensordatenformat: <T><HT><RH><HT><ax><
- Vor dem Senden des Datenpaketes rote LED (P5.7) ein Wird der Erhalt des Datenpaketes bestätigt rote LED (P5.7) aus

Lösung siehe 1.2.2!

11.2.2 Für den Empfänger:

- Transceiver in den Empfangsmode setzen
- Initialisierung Funkkanal entsprechend des Arbeitsplatzes
- Initialisierung auf ID = Arbeitsplatznummer
- Empfangenes Datenpaket auf die serielle Schnittstelle ausgeben
- Ausgabeformat der Zeichenkette: <T><HT><RH><HT><ax><HT><ay><HT><az><HT><TA><CR><LF>
- Bei TA = 2 grüne LED (P4.2) an
- Bei TA = 1 grüne LED (P4.2) aus
- Bei jedem Datenpaket gelbe LED (P4.1) toggeln
- Bei Beschleunigungsdifferenzen größer 20% zum vorhergehenden Wert (in mindestens einer Achse) rote LED (P4.0) an, sonst aus
- Jedes empfangene Datenpaket per Funk bestätigen

```
1
2  //===========================================================
3  //=== Startprojekt zum Mikroprozessorpraktikum ==============
4  //===========================================================
5  #include "msp430x16x.h"      // Systemdefinitionen für den MSP430F1612
6  #include "init.h"            // Initialisierung des Mikrocontrollers
7  #include "system.h"          // Systemfunktionen MSB430H
8  #include "interrupts.h"      // ISR - Interrupt Service Routinen
9
10 #include "HW_CC1100.h"       // CC1100 868MHz Funk Transceiver
11 #include "HW_SHT11.h"        // SHT11 Feuchte- und Temperatursensor
12 #include "HW_MMA7260Q.h"     // MMA7260Q 3D-Beschleunigungssensor
13 #include "HW_LCD_TS.h"       // DOGM128 LCD Display Touchscreen
```

```
14 #include "HW_LCD.h"          // DOGM128 LCD Display
15
16 #include "stdio.h"          // Standard Libary
17 #include "string.h"         // Standard Libary
18 #include "math.h"           // Standard Libary
19
20 //#include "aufgabe_x.h"    // Header Datei für aufgabe_x.c
21                              // möglichst eigene xxx.h und xxx.c Dateien
22                              // erstellen und verwenden.
23 main(void);                  //Haupprogramm
24 void print_value(void);
25
26
27 mode_t mode;
28
29 unsigned char secs = 0;
30
31 #define BIT_SET(a,b) ((a) |= (b))
32 #define BIT_CLR(a,b) ((a) &= ~(b))
33 #define BIT_TOGGLE(a,b) ((a) ^= (b))
34
35 //====Hauptprogramm=============================================
36
37
38 main(void)
39     {
40
41     mode = RECEIVER;
42
43 //===Hier sollten Variablen deklariert werden ==========================
44     //unsigned char i = 0;
45     //char text[60];
46     //int x,y;
47
48 //===Hier die notwendigen Initialisierungsschritte ====================
49 //=(1)== Port-Initialisierung ========================================
50     init_Port();                 // Initialisierung der Port Register
51
52 //(2)=== Clock-System-Initialisierung ================================
53     //== XT2() oder Dco() als Taktquelle einstellen
54     //== durch Ein- oder Auskommentieren
55     //== DCO ist bei LPM Einsatz bevorzugt muß zyklisch kalibriert werden
56     //== XT2 ist quarzstabil muß nicht zyklisch kalibriert werden
57     //
58     //XT2 ();                     // XT2 Taktquelle aktivieren mit 7.3728MHz
59     DCO ();                  // Dco Taktquelle aktivieren mit 7.3728MHz
60     //                          beachte DELTA
61
62 //=(3)== Timer-Initialisierung= ======================================
63     init_Timer_A();          // Init Timer für Sekundeninterrupt
64                              // !! noch leere Funktion
65
66 //=(4)== USART-Initialisierung ========================================
67     init_UART1();            // UART-RS232 mit 115.2kBit/s initialisieren
68                              // !! noch leere Funktion
69
70 //=(5)== CC1100-Transceiver-Initialisierung ==========================
71     init_UART0_SPI();        // CC1100 SPI UART initalisieren
72     init_CC1100();           // CC1100 init und in RX Mode setzen
73                              // !!!Interrupte sind ab jetzt freigegeben!!
74     //== Adresse und Funkkanal des Transceivers setzen
75     //== für die Arbeitsplaetze HWPx (x=1...10) sollten
76     //== ID=x und channnel=x gesetzt werden
77     if (SENDER == mode) {
78         ID = 18;                         // Adresse
79         channel = 8;                     // Funkkanal
80     } else {
81         ID = 8;                          // Adresse
82         channel = 8;                     // Funkkanal
83     }
84     setUid(ID);                      // Adresse im Transceiver setzen
```

```
85      switchFreq(channel);              // Funkkanal im Transceiver setzen
86      //== Soll der Transceiver genutzt werden müssen folgende zwei Zeilen
87      //== auskommentiert werden:
88      /*init_CC1100_IDLE();         // CC1100 in den IDLE Mode setzen
89      init_CC1100_POWERDOWN();*/// CC1100 in den PowerDown Mode setzen
90
91  //=(6)== LCD-Display-Initialisierung ================================
92      dogm_reset();         // Hardware Reset des LCD Controllers
93      dogm_init();          // Initialisierung der LCD Controller Register
94      lcd_clear(WHITE);     // Grafikspeicher auf dem MSP430 löschen
95      //lcd_string(BLACK, 15, 25, "MSP430-GESTARTET!");    // Textausgabe
96      lcd_paint();          // Grafikspeicher auf das LCD Display ausgeben
97
98
99  #define LED_ROT (0x01)    // 0 0 1 P4.0
100 #define LED_GELB (0x02)   // 0 1 0 P4.1
101 #define LED_GRUEN (0x04) // 1 0 0 P4.2
102 #define LED_ALL (LED_ROT | LED_GELB | LED_GRUEN)
103
104 #define LED_ON(led) (BIT_CLR(P4OUT, led))
105 #define LED_OFF(led) (BIT_SET(P4OUT, led))
106 #define LED_TOGGLE(led) (BIT_TOGGLE(P4OUT, led))
107
108 #define IS_LED_ON(led) (!(P4OUT & led))
109
110 #define TASTE_LINKS (0x1)
111 #define TASTE_RECHTS (0x2)
112
113 #define SLEEP_QUANTUM 10000
114 #define SLEEP(n) do {         /* sleep for n seconds */ \
115     long time = n * 100000; /* wait() sleeps 10*n microseconds */ \
116     while(time > SLEEP_QUANTUM) { \
117         wait(SLEEP_QUANTUM); \
118         time -= SLEEP_QUANTUM; \
119     } \
120     wait(time); \
121 } while(0)
122
123
124     if (SENDER == mode) {
125         // interrupt wird bei 0 ausgeloest
126         TBR = 1;
127
128         // waehle ACLK
129         BIT_SET(TBCTL, TBSSEL0);
130         BIT_CLR(TBCTL, TBSSEL1);
131
132         // setze divider auf 8
133         BIT_SET(TBCTL, ID0 | ID1);
134
135         // wahle count up mode
136         BIT_SET(TBCTL, MC0);
137         BIT_CLR(TBCTL, MC1);
138
139         // setze taktanzahl fuer eine sekunde
140         TBCCR0 = 0x1000;
141
142         // loesche interrupt flag fuer timer
143         BIT_CLR(TBCCTL0, CCIFG);
144
145         // AD Wandler Konfiguration
146
147         // setze Pin 5.4, 5.5, 5.6 als output
148         BIT_SET(P5DIR, BIT4 + BIT5 + BIT6);
149         // Beschleunigungssensor g-Range 1.5g, Sensitivity 800mV/g
150         BIT_CLR(P5OUT, BIT4);
151         BIT_CLR(P5OUT, BIT5);
152         // sleep mode off
153         BIT_SET(P5OUT, BIT6);
154
155         // waehle P6.0, P6.1, P6.2 als input
```

```
156             P6SEL = BIT0 + BIT1 + BIT2;
157             // schalte wandler an, waehle "multiple sample and conversion", SHT = 0 -> 4 cycles
158             ADC12CTL0 = ADC12ON + MSC;
159             // ADC12 clock, sequence of channels
160             ADC12CTL1 = CONSEQ0 + SHP;
161             // waehle channel A0, A1, A2
162             ADC12MCTL0 = INCH_0;
163             ADC12MCTL1 = INCH_1;
164             ADC12MCTL2 = INCH_2 + EOS;
165
166             // loesche interrupt flag
167             BIT_CLR(ADC12IFG, BIT2);
168
169             // erlaube interrupt
170             BIT_SET(ADC12IE, BIT2);
171
172             // erlaube timer interrupt
173             BIT_SET(TBCCTL0, CCIE);
174
175             // erlaube conversion
176             ADC12CTL0 |= ENC;
177         }
178
179         LED_OFF(LED_ALL);
180
181         _bis_SR_register(GIE);
182
183 //===Hier die Endlosschleife quasi das Betriebssystem==================
184
185 while(1){
186 }    // Ende der Endlosschleife
187 }    // Ende Main
188 //====Ende des Hauptprogramms ========================================
189
190 #pragma vector = TIMERB0_VECTOR
191 __interrupt void TIMERB0(void)
192 {
193     secs++;
194     if (5 == secs) {
195         // starte conversion
196         ADC12CTL0 |= ADC12SC;
197     }
198 }
199
200 #pragma vector = ADC12_VECTOR
201 __interrupt void ADC12(void)
202 {
203     char buffer[64];
204     int tasten_status = 0;
205     int x, y, z;
206     if (ADC12IFG & BIT2)
207     {
208         // Beschleunigungsdaten
209         x = ADC12MEM0;
210         y = ADC12MEM1;
211         z = ADC12MEM2;
212
213         BIT_CLR(ADC12IFG, BIT2 + BIT1 + BIT0);
214
215         // tasten status
216         if (P1IN & TASTE_RECHTS) {
217             tasten_status += 1;
218         }
219         if (P1IN & TASTE_LINKS) {
220             tasten_status += 2;
221         }
222
223         // sensor lesen
224         SHT11_Read_Sensor();
225
```

```
226         sprintf(buffer, "%s\t%s\t%d\t%d\t%d\t%d\r\n", temp_char, humi_char, x, y, z, tasten_status
                );
227
228         // rote LED an
229         LEDON;
230
231         // sende string terminierung mit
232         sendPacket(8, 18, buffer, strlen(buffer)+1);
233
234         secs = 0;
235     }
236 }
237
238 #pragma vector = PORT2_VECTOR   // ISR für den CC1100 Transceiver
239 __interrupt void PORT2 (void) {
240     char res;                           // CRC Check
241     int tasten_status = 0;
242     int x, y, z;
243     static int last_x = 0, last_y = 0, last_z = 0;
244     char temp[16];
245     char humi[16];
246     char data[64];
247     char ack[] = "ACK";
248     int i = 0;
249     float x_dif;
250     float y_dif;
251     float z_dif;
252
253     if (P2IFG & 0x01)   // Check P2IFG Bit P2.0 - CC1100 Rx Packet
254         {
255         CLEAR(P2IFG, 0x01);
256         if (RECEIVER == mode) {
257             LED_TOGGLE(LED_GELB);
258         }
259
260         res = receivePacket();  // CRC Rückgabe
261         if (res)                // wenn Paket OK ...
262             {
263                 // kopiere daten
264                 for (i = 2; i < RxCC1100.length; i++)
265                 {
266                     data[i-2] = RxCC1100.data[i];
267                 }
268                 data[i-2] = 0x00;
269
270                 if (RECEIVER == mode) {
271                     // receiver
272                     printPacket(1); // 0 - Paket auf LCD ausgeben
273                                     // 1 - Paket auf UART1 ausgeben
274
275                     sendPacket(18, 8, ack, strlen(ack)+1);
276
277                     // parse paket
278                     sscanf(data, "T=_%s\t%s\t%d\t%d\t%d\t%d\r\n", temp, humi, &x, &y, &z, &
                            tasten_status);
279
280                     lcd_clear(WHITE);   // Grafikspeicher auf dem MSP430 löschen
281                     lcd_string(BLACK, 2, 2, data);  // Textausgabe
282                     lcd_paint();        // Grafikspeicher auf das LCD Display ausgeben
283
284                     // ueberpruefe tasten status
285                     if (2 == tasten_status) {
286                         LED_ON(LED_GRUEN);
287                     } else if (1 == tasten_status) {
288                         LED_OFF(LED_GRUEN);
289                     }
290
291                     x_dif = fabs(((float)x/(float)last_x) - 1);
292                     y_dif = fabs(((float)y/(float)last_y) - 1);
293                     z_dif = fabs(((float)z/(float)last_z) - 1);
294                     if ((x_dif > 0.2) || (y_dif > 0.2) || (z_dif > 0.2)) {
```

```
295                         LED_ON(LED_ROT);
296                     } else {
297                         LED_OFF(LED_ROT);
298                     }
299 
300                     last_x = x;
301                     last_y = y;
302                     last_z = z;
303 
304                 } else {
305                     // sender
306                     // ueberpruefe laenge (src byte + dst byte + laenge von ack + \0)
307                     if (strlen(ack)+3 == RxCC1100.length) {
308                         if (!memcmp(data, ack, strlen(ack))) {
309                             // rote LED an
310                             LEDOFF;
311                         }
312                     }
313                 }
314             }
315         else
316             {
317                 spiStrobe(CC1100_SIDLE);     // Switch to IDLE
318                 spiStrobe(CC1100_SFRX);      // Flush the RX FIFO
319             }
320         }
321     else
322         {
323         CLEAR(P2IFG, 0xFF);              // Clear all flags
324         }
325     spiStrobe(CC1100_SRX);               // Switch to RX Mode
326 }
```