Mikroprozessorpraktikum WS 12/13

Carlos Martín Nieto, Simon Hohberg, Tu Tran

February 23, 2013

10 Der DMA Controller des MSP430F1612

10.1 Schnelles Senden eines Buffers über die UART im DMA_Mode

10.1.1 Entwickeln Sie ein Programm, das folgende Funktionalitäten bietet:

- Definieren Sie eine Variable char uart_buffer[100];
- Initialisieren Sie die Variable mit einer Zeichenkette der Länge Null
- Bei Druck auf die Taste P1.0 sollen neue Sensorwerte wie in A 7.3.1 eingelesen werden
- Die Sensorwete sollen in die Variable per sprintf() eingetragen werden.
- Danach sollen diese Daten per DMA an die UART geschickt werden
- Initialisieren Sie die DMA in der Form, dass die Zeichenkette aus der Varible auf die Serielle Schnittstelle ausgegeben wird
- Die Anzahl der Bytes bestimmen Sie aus der Länge der Zeichenkette
- Schalten Sie beim Start des DMA Datentransfers die LED P4.0 ein und am Ende des Datentransfers (in der ISR) aus

```
1  //------------------------------- init.c ----------------------------------------
2
3  void init_UART1(void)
4      {
5      //  P3SEL .....;              // USART RX und TX dem Modul zuweisen
6      P3SEL = BIT6 | BIT7;
7      //  U1CTL .....;              // Reset
8      U1CTL = SWRST;
9      //  U1CTL .....;              // Format 8N1
10     U1CTL |= CHAR; // 8 Bits Daten, keine Paritaet und 1-Bit Stop ist default
11     //  U1TCTL.....;              // Taktquelle SMCLK
12     U1TCTL |= SSEL0 + SSEL1; // Taktquelle SMCLK
13     //  U1BR0 .....;              // Teiler Low-Teil, da 7372800/64 ca. 115200
14     U1BR0 = 64; //7372800/64 ca. 115200
15     U1BR1 = 0;
16     //  U1BR1 .....;              // Teiler High-Teil
17     //  U1MCTL ....;              // Modulationskontrolle
18     U1MCTL = 0;
19     //  ME2 .......;              // Enable USART1 TXD/RXD
20     ME2 = BIT5 + BIT4;
21     //  U1CTL .....;              // Reset
22     U1CTL &= ~SWRST;
23     //  IE2 .......;              // Enable Interrupt
24     //IE2 = BIT4;
25     }
26
27
28 //------------------------------- main.c ----------------------------------------
29
30 //=============================================================================
```

```
31  //=== Startprojekt zum Mikroprozessorpraktikum ============================
32  //==========================================================================
33  #include "msp430x16x.h"      // Systemdefinitionen für den MSP430F1612
34  #include "init.h"            // Initialisierung des Mikrocontrollers
35  #include "system.h"          // Systemfunktionen MSB430H
36  #include "interrupts.h"      // ISR - Interrupt Service Routinen
37
38  #include "HW_CC1100.h"       // CC1100 868MHz Funk Transceiver
39  #include "HW_SHT11.h"        // SHT11 Feuchte- und Temperatursensor
40  #include "HW_MMA7260Q.h"     // MMA7260Q 3D-Beschleunigungssensor
41  #include "HW_LCD_TS.h"       // DOGM128 LCD Display Touchscreen
42  #include "HW_LCD.h"          // DOGM128 LCD Display
43
44  #include "stdio.h"           // Standard Libary
45  #include "string.h"          // Standard Libary
46  #include "math.h"            // Standard Libary
47
48  //#include "aufgabe_x.h"     // Header Datei für aufgabe_x.c
49                               // möglichst eigene xxx.h und xxx.c Dateien
50                               // erstellen und verwenden.
51  main(void);                  //Hauptprogramm
52  void print_value(void);
53
54  #define BIT_SET(a,b) ((a) |= (b))
55  #define BIT_CLR(a,b) ((a) &= ~(b))
56  #define BIT_TOGGLE(a,b) ((a) ^= (b))
57
58  //====Hauptprogramm=========================================================
59
60  char uart_buffer[100] = { '\0' };
61
62  void print_buf(const char *str)
63  {
64      do {
65          U1TXBUF = *str;
66          while (!(U1TCTL & TXEPT));
67      } while(*str++);
68  }
69
70  main(void)
71      {
72  //===Hier sollten Variablen deklariert werden ==============================
73      //unsigned char i = 0;
74      //char text[60];
75      //int x,y;
76
77  //===Hier die notwendigen Initialisierungsschritte =========================
78  //=(1)== Port-Initialisierung ==============================================
79      init_Port();                 // Initialisierung der Port Register
80
81  //(2)=== Clock-System-Initialisierung ======================================
82      //== XT2() oder Dco() als Taktquelle einstellen
83      //== durch Ein- oder Auskommentieren
84      //== DCO ist bei LPM Einsatz bevorzugt muß zyklisch kalibriert werden
85      //== XT2 ist quarzstabil muß nicht zyklisch kalibriert werden
86      //
87      //XT2 ();                    // XT2 Taktquelle aktivieren mit 7.3728MHz
88      DCO ();                  // Dco Taktquelle aktivieren mit 7.3728MHz
89      //                          beachte DELTA
90
91  //=(3)== Timer-Initialisierung= ============================================
92      init_Timer_A();          // Init Timer für Sekundeninterrupt
93                               // !! noch leere Funktion
94
95  //=(4)== USART-Initialisierung =============================================
96      init_UART1();            // UART-RS232 mit 115.2kBit/s initialisieren
97                               // !! noch leere Funktion
98
99  //=(5)== CC1100-Transceiver-Initialisierung ================================
100     init_UART0_SPI();        // CC1100 SPI UART initalisieren
101     init_CC1100_POWERDOWN();          // CC1100 init und in RX Mode setzen
```

```
102                             // !!!Interrupte sind ab jetzt freigegeben!!
103     //== Adresse und Funkkanal des Transceivers setzen
104     //== für die Arbeitsplaetze HWPx (x=1...10) sollten
105     //== ID=x und channnel=x gesetzt werden
106     ID = 1;                         // Adresse
107     setUid(ID);                     // Adresse im Transceiver setzen
108     channel = 1;                    // Funkkanal
109     switchFreq(channel);            // Funkkanal im Transceiver setzen
110     //== Soll der Transceiver genutzt werden müssen folgende zwei Zeilen
111     //== auskommentiert werden:
112     init_CC1100_IDLE();     // CC1100 in den IDLE Mode setzen
113     init_CC1100_POWERDOWN();// CC1100 in den PowerDown Mode setzen
114
115 //=(6)== LCD-Display-Initialisierung ================================
116     dogm_reset();       // Hardware Reset des LCD Controllers
117     dogm_init();        // Initialisierung der LCD Controller Register
118     lcd_clear(WHITE);   // Grafikspeicher auf dem MSP430 löschen
119     //lcd_string(BLACK, 15, 25, "MSP430-GESTARTET!");   // Textausgabe
120     lcd_paint();        // Grafikspeicher auf das LCD Display ausgeben
121
122
123 #define LED_ROT (0x01)   // 0 0 1 P4.0
124 #define LED_GELB (0x02)  // 0 1 0 P4.1
125 #define LED_GRUEN (0x04) // 1 0 0 P4.2
126 #define LED_ALL (LED_ROT | LED_GELB | LED_GRUEN)
127
128 #define LED_ON(led) (BIT_CLR(P4OUT, led))
129 #define LED_OFF(led) (BIT_SET(P4OUT, led))
130 #define LED_TOGGLE(led) (BIT_TOGGLE(P4OUT, led))
131
132 #define IS_LED_ON(led) (!(P4OUT & led))
133
134 #define TASTE_LINKS (0x1)
135 #define TASTE_RECHTS (0x2)
136
137 #define SLEEP_QUANTUM 10000
138 #define SLEEP(n) do {       /* sleep for n seconds */ \
139     long time = n * 100000; /* wait() sleeps 10*n microseconds */ \
140     while(time > SLEEP_QUANTUM) { \
141         wait(SLEEP_QUANTUM); \
142         time -= SLEEP_QUANTUM; \
143     } \
144     wait(time); \
145 } while(0)
146
147 BIT_CLR(P1DIR, TASTE_LINKS);
148 BIT_CLR(P1IES, TASTE_LINKS); // LH
149 BIT_SET(P1IE, TASTE_LINKS);
150 BIT_CLR(P1IFG, TASTE_LINKS);
151
152
153 _bis_SR_register(GIE);
154
155 memset(uart_buffer, 0, sizeof(uart_buffer));
156
157 //===Hier die Endlosschleife quasi das Betriebssystem==================
158 while(1){
159     //SHT11_Read_Sensor();
160     //sprintf(uart_buffer, "%s %s\r\n", temp_char, humi_char);
161     //print_buf(uart_buffer);
162     //lcd_clear(WHITE); // Grafikspeicher auf dem MSP430 löschen
163     //lcd_string(BLACK, 15, 25, uart_buffer);   // Textausgabe
164     //lcd_paint();      // Grafikspeicher auf das LCD Display ausgeben
165     //wait(30000);
166
167 }   // Ende der Endlosschleife
168 }   // Ende Main
169 //====Ende des Hauptprogramms ========================================
170
171 #pragma vector = PORT1_VECTOR
172 __interrupt void PORT1(void)
```

```
{
    if (P1IFG & TASTE_LINKS) {
        BIT_CLR(P1IFG, TASTE_LINKS);
        LED_ON(LED_ROT);

        SHT11_Read_Sensor();
        sprintf(uart_buffer, "%s %s\r\n", temp_char, humi_char);

        // USART konfigurieren
        BIT_CLR(IFG2, UTXIFG1); // interrupt flag loeschen
        BIT_CLR(IE2, UTXIE1);   // interrupt ausschaulten

        // DMA konfigurieren
        DMA2SZ = strlen(uart_buffer);
        DMA2DA = (unsigned int)&U1TXBUF; // speichere die Addresse des TX Buffers
        DMA2SA = (unsigned int)uart_buffer; // und die von unserem String
        DMACTL0 = DMA2TSEL3 + DMA2TSEL1; // Trigger wenn TX IFG gestezt ist
        // kopiere 1 Byte vom String ins TX Buffer bei jedem Transfer und erhoehe src um eins
        DMA2CTL = DMAEN + DMASRCINCR0 + DMASRCINCR1 + DMADSTBYTE + DMASRCBYTE;

        // Bescheid sagen, wenn der Transfer fertig ist
        BIT_CLR(DMA2CTL, DMAIFG);
        DMA2CTL += DMAIE;

        BIT_SET(IFG2, UTXIFG1); // starte den ersten Transfer
    }
}

#pragma vector = DACDMA_VECTOR
__interrupt void DMA(void)
{
    // ISR aufgerufen, wenn der Transfer fertig ist
    if (DMA2CTL & DMAIFG) {
        BIT_CLR(DMA2CTL, DMAIFG);
        // Lampe ausmachen, wir sind fertig
        LED_OFF(LED_ROT);
    }
}
```

10.1.2 Berechnen Sie auf Basis der Datenübertragungsrate die Zeit für die Übertragung der Zeichenkette. Überprüfen Sie messtechnisch mit dem Counter HM8021 die ermittelte Zeit. Nutzen Sie für die Messung den Port P1 Bit2 als Signalquelle. Beim Start der Übertragung setzen Sie das Bit auf "1" und beim Ende der Übertragung auf "0". Der Vorgang muß innerhalb einer Sekunde zyklisch ablaufen damit das Messgerät richtig funktioniert (also Tastendruck per Software emulieren).

Es werden 20 Byte gesendet:

- Berechnung: $\frac{20\,Byte \cdot 8}{115200\,Symbole(=Bit)/s} = \frac{160\,Bit \cdot s}{115{,}2 \cdot 10^3\,Bit} = 1{,}3889 \cdot 10^{-3}\,s = \underline{1{,}3889\,ms}$
- 1. Messung: $1{,}73\,ms$
- 2. Messung: $1{,}77\,ms$

10.2 Ausgabe von Analogwerten über den DA-Umsetzer im DMA_Mode

10.2.1 Verändern Sie die Lösung aus der Aufgabe A 9.1.1 in der Form, dass die Ausgabe der Sinuswerte auf den D/A-Wandler durch die DMA Einheit vorgenommen wird.

```

//===========================================================
//=== Startprojekt zum Mikroprozessorpraktikum ==============
//===========================================================
#include "msp430x16x.h"      // Systemdefinitionen für den MSP430F1612
#include "init.h"            // Initialisierung des Mikrocontrollers
#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 #define BIT_SET(a,b) ((a) |= (b))
27 #define BIT_CLR(a,b) ((a) &= ~(b))
28 #define BIT_TOGGLE(a,b) ((a) ^= (b))
29
30 //====Hauptprogramm=============================================
31
32
33 #define SINUS_VALUES 100
34
35 int sinus[SINUS_VALUES];
36
37 main(void)
38     {
39         float step;
40         int i;
41 //===Hier sollten Variablen deklariert werden ==========================
42     //unsigned char i = 0;
43     //char text[60];
44     //int x,y;
45
46 //===Hier die notwendigen Initialisierungsschritte =====================
47 //=(1)== Port-Initialisierung ========================================
48     init_Port();                 // Initialisierung der Port Register
49
50 //(2)=== Clock-System-Initialisierung ================================
51     //== XT2() oder Dco() als Taktquelle einstellen
52     //== durch Ein- oder Auskommentieren
53     //== DCO ist bei LPM Einsatz bevorzugt muß zyklisch kalibriert werden
54     //== XT2 ist quarzstabil muß nicht zyklisch kalibriert werden
55     //
56     //XT2 ();                    // XT2 Taktquelle aktivieren mit 7.3728MHz
57     DCO ();                  // Dco Taktquelle aktivieren mit 7.3728MHz
58     //                          beachte DELTA
59
60 //=(3)== Timer-Initialisierung= ======================================
61     init_Timer_A();          // Init Timer für Sekundeninterrupt
62                              // !! noch leere Funktion
63
64 //=(4)== USART-Initialisierung ========================================
65     init_UART1();            // UART-RS232 mit 115.2kBit/s initialisieren
66                              // !! noch leere Funktion
67
68 //=(5)== CC1100-Transceiver-Initialisierung ==========================
69     init_UART0_SPI();        // CC1100 SPI UART initalisieren
70     init_CC1100_POWERDOWN();             // CC1100 init und in RX Mode setzen
71                              // !!!Interrupte sind ab jetzt freigegeben!!
72     //== Adresse und Funkkanal des Transceivers setzen
73     //== für die Arbeitsplaetze HWPx (x=1...10) sollten
74     //== ID=x und channnel=x gesetzt werden
75     ID = 1;                            // Adresse
76     setUid(ID);                        // Adresse im Transceiver setzen
77     channel = 1;                       // Funkkanal
78     switchFreq(channel);               // Funkkanal im Transceiver setzen
```

```
79      //== Soll der Transceiver genutzt werden müssen folgende zwei Zeilen
80      //== auskommentiert werden:
81      init_CC1100_IDLE();       // CC1100 in den IDLE Mode setzen
82      init_CC1100_POWERDOWN();// CC1100 in den PowerDown Mode setzen
83
84  //=(6)== LCD-Display-Initialisierung ================================
85      dogm_reset();         // Hardware Reset des LCD Controllers
86      dogm_init();          // Initialisierung der LCD Controller Register
87      lcd_clear(WHITE);     // Grafikspeicher auf dem MSP430 löschen
88      //lcd_string(BLACK, 15, 25, "MSP430-GESTARTET!");    // Textausgabe
89      lcd_paint();          // Grafikspeicher auf das LCD Display ausgeben
90
91
92  #define LED_ROT (0x01)    // 0 0 1 P4.0
93  #define LED_GELB (0x02)   // 0 1 0 P4.1
94  #define LED_GRUEN (0x04) // 1 0 0 P4.2
95  #define LED_ALL (LED_ROT | LED_GELB | LED_GRUEN)
96
97  #define LED_ON(led) (BIT_CLR(P4OUT, led))
98  #define LED_OFF(led) (BIT_SET(P4OUT, led))
99  #define LED_TOGGLE(led) (BIT_TOGGLE(P4OUT, led))
100
101 #define IS_LED_ON(led) (!(P4OUT & led))
102
103 #define TASTE_LINKS (0x1)
104 #define TASTE_RECHTS (0x2)
105
106 #define SLEEP_QUANTUM 10000
107 #define SLEEP(n) do {         /* sleep for n seconds */ \
108     long time = n * 100000; /* wait() sleeps 10*n microseconds */ \
109     while(time > SLEEP_QUANTUM) { \
110         wait(SLEEP_QUANTUM); \
111         time -= SLEEP_QUANTUM; \
112     } \
113     wait(time); \
114 } while(0)
115
116     // Schrittlaenge
117     step = (2*3.14159)/SINUS_VALUES;
118
119     // berechne Ausgangswerte fuer sinus
120     for (i = 0; i < SINUS_VALUES; i++) {
121         double x = sin(i*step);
122         x /= 2.0;    // Amplitude von sinus ist 2, wir moechten 1
123         x += 1.0;    // Gleichspannungsanteil, Verschiebung entlang der y-Achse
124
125         sinus[i] = x * ((double)4095/(double)3);     // 3V maximale Ausgangsspannung, 12 Bit
                Aufloesung => 4096 Werte
126     }
127
128
129     BIT_SET(BCSCTL2, SELS); // waehle XT2CLK als SMCLK
130
131     // waehle SMCLK (7.4 MHz ohne divider)
132     BIT_SET(TBCTL, TBSSEL1);
133     BIT_CLR(TBCTL, TBSSEL0);
134
135     // wahle count up mode
136     BIT_SET(TBCTL, MC0);
137     BIT_CLR(TBCTL, MC1);
138
139     // setze anzahl fuer interrupt
140     TBCCR0 = 740; // 7.4 MHz: takt=7_400_000 / 10000 = count to 740
141
142     // loesche interrupt flag fuer timer
143     BIT_CLR(TBCCTL0, CCIFG);
144
145     // 1x Referenzspannung, schnell
146     DAC12_1CTL = DAC12IR + DAC12AMP0 + DAC12AMP1 + DAC12AMP2 + DAC12SREF_2 + DAC12CALON;
147
148     // DMA konfigurieren
```

```
149     DMA2SZ = SINUS_VALUES;
150     DMA2DA = (unsigned int)&DAC12_1DAT; // speichere die Addresse des DAU Buffers
151     DMA2SA = (unsigned int)sinus; // und die unserer Werte
152     DMACTL0 = DMA2TSEL3; // Trigger kommt von TBCCR0
153     // kopiere 1 Byte vom String ins TX Buffer bei jedem Transfer und erhoehe src um eins
154     DMA2CTL = DMAEN + DMASRCINCR0 + DMASRCINCR1;
155
156     // Bescheid sagen, wenn der Transfer fertig ist
157     BIT_CLR(DMA2CTL, DMAIFG);
158     DMA2CTL += DMAIE;
159
160     _bis_SR_register(GIE);
161
162     // interrupt wird bei 0 ausgeloest
163     TBR = 1;
164 //===Hier die Endlosschleife quasi das Betriebssystem==================
165 while(1){
166 }   // Ende der Endlosschleife
167 }   // Ende Main
168 //====Ende des Hauptprogramms ========================================
169
170
171 #pragma vector = DACDMA_VECTOR
172 __interrupt void DMA(void)
173 {
174     // ISR aufgerufen, wenn der Transfer fertig ist
175     if (DMA2CTL & DMAIFG) {
176         BIT_CLR(DMA2CTL, DMAIFG);
177         DMA2SZ = SINUS_VALUES;
178         DMA2DA = (unsigned int)&DAC12_1DAT; // speichere die Addresse des DAU Buffers
179         DMA2SA = (unsigned int)sinus; // und die unserer Werte
180         DMA2CTL += DMAEN;
181     }
182 }
```

10.3 Einlesen von Analogwerten über den AD-Umsetzer im DMA-Mode

10.3.1 Entwickeln Sie ein Programm, das folgende Funktionalitäten bietet, wobei sich der Mikrocontroller in dem höchstmöglichsten Low Power Mode befindet:

- Durch einen Timer getaktet, sollen vom AD-Umsetzer (P6.7) 100 Messwerte in ein Array eingelesen werden (siehe A 8.1). Dabei soll der AD-Umsetzer über einen Zeitraum von 1 Sekunden 100 Meßwerte in äquidistanten Zeitintervallen von jeweils 0,01 Sekunde erfassen. Die Taktung der Wandlung des AD-Umsetzers soll ohne Interrupt über TB1 (siehe A 6.5) als Taktquelle für den AD-Umsetzer erfolgen.
- Das Auslesen aus dem AD-Umsetzer und der Datentransport in das Array sollen per DMA erfolgen. Der DMA wird dabei durch das ADC12IFGx-Bit getriggert.
- Nach erfolgter Übertragung von 100 Messwerten durch die DMA, soll in der DMA-ISR der Mittelwert aus den 100 Messwerten berechnet werden
- Der Mittelwert soll als Zeichenkette mittels DMA auf die UART ausgegeben werden
- Der berechnete Mittelwert soll per DMA getriggert durch das UTXIFG1-Bit über die UART1 ausgegeben werden.
- Der ganze Prozess soll endlos laufen
- Schalten Sie während der Zeit der Mittelwertbildung und der Übertragung über die UART die LED (P4.0) ein.

```
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 char buffer[16];
27
28 #define BIT_SET(a,b) ((a) |= (b))
29 #define BIT_CLR(a,b) ((a) &= ~(b))
30 #define BIT_TOGGLE(a,b) ((a) ^= (b))
31
32 //====Hauptprogramm=============================================
33
34
35 int values[100];
36
37 main(void)
38     {
39 //===Hier sollten Variablen deklariert werden ==========================
40     //unsigned char i = 0;
41     //char text[60];
42     //int x,y;
43
44 //===Hier die notwendigen Initialisierungsschritte =====================
45 //=(1)== Port-Initialisierung ==========================================
46     init_Port();                 // Initialisierung der Port Register
47
48 //(2)=== Clock-System-Initialisierung ==================================
49     //== XT2() oder Dco() als Taktquelle einstellen
50     //== durch Ein- oder Auskommentieren
51     //== DCO ist bei LPM Einsatz bevorzugt muß zyklisch kalibriert werden
52     //== XT2 ist quarzstabil muß nicht zyklisch kalibriert werden
53     //
54     //XT2 ();                    // XT2 Taktquelle aktivieren mit 7.3728MHz
55     DCO ();                  // Dco Taktquelle aktivieren mit 7.3728MHz
56     //                          beachte DELTA
57
58 //=(3)== Timer-Initialisierung= ========================================
59     init_Timer_A();          // Init Timer für Sekundeninterrupt
60                              // !! noch leere Funktion
61
62 //=(4)== USART-Initialisierung =========================================
63     init_UART1();            // UART-RS232 mit 115.2kBit/s initialisieren
64                              // !! noch leere Funktion
65
66 //=(5)== CC1100-Transceiver-Initialisierung ============================
67     init_UART0_SPI();        // CC1100 SPI UART initalisieren
68     init_CC1100_POWERDOWN();             // CC1100 init und in RX Mode setzen
69                              // !!!Interrupte sind ab jetzt freigegeben!!
70     //== Adresse und Funkkanal des Transceivers setzen
71     //== für die Arbeitsplaetze HWPx (x=1...10) sollten
72     //== ID=x und channnel=x gesetzt werden
73     ID = 1;                            // Adresse
74     setUid(ID);                        // Adresse im Transceiver setzen
75     channel = 1;                       // Funkkanal
76     switchFreq(channel);               // Funkkanal im Transceiver setzen
```

```
77      //== Soll der Transceiver genutzt werden müssen folgende zwei Zeilen
78      //== auskommentiert werden:
79      init_CC1100_IDLE();       // CC1100 in den IDLE Mode setzen
80      init_CC1100_POWERDOWN();// CC1100 in den PowerDown Mode setzen
81
82  //=(6)== LCD-Display-Initialisierung ================================
83      dogm_reset();         // Hardware Reset des LCD Controllers
84      dogm_init();          // Initialisierung der LCD Controller Register
85      lcd_clear(WHITE);     // Grafikspeicher auf dem MSP430 löschen
86      //lcd_string(BLACK, 15, 25, "MSP430-GESTARTET!");    // Textausgabe
87      lcd_paint();          // Grafikspeicher auf das LCD Display ausgeben
88
89
90  #define LED_ROT (0x01)    // 0 0 1 P4.0
91  #define LED_GELB (0x02)   // 0 1 0 P4.1
92  #define LED_GRUEN (0x04) // 1 0 0 P4.2
93  #define LED_ALL (LED_ROT | LED_GELB | LED_GRUEN)
94
95  #define LED_ON(led) (BIT_CLR(P4OUT, led))
96  #define LED_OFF(led) (BIT_SET(P4OUT, led))
97  #define LED_TOGGLE(led) (BIT_TOGGLE(P4OUT, led))
98
99  #define IS_LED_ON(led) (!(P4OUT & led))
100
101 #define TASTE_LINKS (0x1)
102 #define TASTE_RECHTS (0x2)
103
104 #define SLEEP_QUANTUM 10000
105 #define SLEEP(n) do {         /* sleep for n seconds */ \
106     long time = n * 100000; /* wait() sleeps 10*n microseconds */ \
107     while(time > SLEEP_QUANTUM) { \
108         wait(SLEEP_QUANTUM); \
109         time -= SLEEP_QUANTUM; \
110     } \
111     wait(time); \
112 } while(0)
113
114     memset(values, 0, 100*sizeof(int));
115
116     // waehle ACLK (32.7 kHz ohne divider)
117     BIT_CLR(TBCTL, TBSSEL0);
118
119     // wahle count up mode
120     BIT_SET(TBCTL, MC0);
121     BIT_CLR(TBCTL, MC1);
122
123     // setze anzahl fuer 100Hz Trigger
124     TBCCR0 = 327; // 32.7kHz: takt=32_768 / 100 = count to 327
125     //TBR = 1;
126
127     // loesche interrupt flag fuer timer
128     BIT_CLR(TBCCTL0, CCIFG);
129
130     P6SEL |= BIT7; // P6.7 als input
131     ADC12CTL0 = ADC12ON + MSC;
132     // Trigger Timer_B.OUT0, SMCLK
133     ADC12CTL1 = CONSEQ1 + SHP;
134     ADC12MCTL0 = INCH_7;
135     ADC12IFG = 0;
136
137     // erlaube konversion
138     ADC12CTL0 |= ENC;
139
140     // DMA konfigurieren
141     DMA2SZ = 100;
142     DMA2SA = (unsigned int)&ADC12MEM0;  // speichere die Addresse des ADC Register
143     DMA2DA = (unsigned int)values;      // und die unserer Werte
144     DMACTL0 = DMA2TSEL2 + DMA2TSEL1;    // Trigger kommt vom ADC
145     // kopiere 1 Word aus dem ADC Register und erhoehe dst um eins
146     DMA2CTL = DMADT2 + DMAEN + DMADSTINCR0 + DMADSTINCR1;
147
```

```
148     // Bescheid sagen, wenn der Transfer fertig ist
149     BIT_CLR(DMA2CTL, DMAIFG);
150     DMA2CTL += DMAIE;
151 
152     ADC12CTL0 |= ADC12SC;
153     _bis_SR_register(GIE);
154 
155 //===Hier die Endlosschleife quasi das Betriebssystem==================
156 
157 while(1){
158     LPM3;
159 }   // Ende der Endlosschleife
160 }   // Ende Main
161 //====Ende des Hauptprogramms ========================================
162 
163 #pragma vector = DACDMA_VECTOR
164 __interrupt void DMA(void)
165 {
166     // fuer synchronisierung des transfers
167     static unsigned char transfer_done = 1;
168 
169     // starte keinen neuen transfer bis aktueller transfer fertig
170     if ((DMA2CTL & DMAIFG) && transfer_done) {
171         // wandlung fertig
172         double val = 0.0;
173         int i;
174 
175         transfer_done = 0;
176 
177         BIT_CLR(DMA2CTL, DMAIFG);
178 
179         LED_ON(LED_ROT);
180 
181         for (i = 0; i < 100; i++) {
182             double this = values[i];
183 
184             this /= 4096.0f;     // Wertbreite
185             this *= 3.3f;        // Referenzspannung
186 
187             val += this/100.0f;
188         }
189 
190         sprintf(buffer, "U=%.2f V\r\n", val);
191 
192         // konfigurieren DMA fuer uart transfer
193 
194         // USART konfigurieren
195         BIT_CLR(IFG2, UTXIFG1); // interrupt flag loeschen
196         BIT_CLR(IE2, UTXIE1);   // interrupt ausschaulten
197 
198         // DMA konfigurieren
199         DMA1SZ = strlen(buffer);
200         DMA1DA = (unsigned int)&U1TXBUF; // speichere die Addresse des TX Buffers
201         DMA1SA = (unsigned int)buffer; // und die von unserem String
202         BIT_SET(DMACTL0, DMA1TSEL3 + DMA1TSEL1); // Trigger wenn TX IFG gestezt ist
203         // kopiere 1 Byte vom String ins TX Buffer bei jedem Transfer und erhoehe src um eins
204         DMA1CTL = DMAEN + DMASRCINCR0 + DMASRCINCR1 + DMADSTBYTE + DMASRCBYTE;
205 
206         // Bescheid sagen, wenn der Transfer fertig ist
207         BIT_CLR(DMA1CTL, DMAIFG);
208         DMA1CTL += DMAIE;
209 
210         BIT_SET(IFG2, UTXIFG1); // starte den ersten Transfer
211 
212     } else if (DMA1CTL & DMAIFG) {
213         // uart transfer fertig
214         transfer_done = 1;
215         BIT_CLR(DMA1CTL, DMAIFG);
216         LED_OFF(LED_ROT);
217     }
218 }
```

10.3.2 Bestimmen Sie mit dem Counter HM8021 messtechnisch die Zeit für die Berechnung des Mittelwertes und die anschließende Datenübertragung über die UART. Nutzen Sie dazu den Port P1 Bit2 als Signalquelle für die Messung. Der Vorgang muß innerhalb einer Sekunde zyklisch ablaufen damit das Messgerät richtig funktioniert (also Tastendruck per Software emulieren).

Zusätzlich zum An- und Ausschalten der roten LED fügen wir ein BIT_SET(P1OUT, BIT2) und BIT_CLR(P1OUT, BIT2) ein.

Messung (Zeit für Berechnung und Übertragung): $22,2\,ms$