Mikroprozessorpraktikum WS 12/13

Carlos Martín Nieto, Simon Hohberg, Tu Tran

February 14, 2013

9 Der Digital/Analog Wandler des MSP430F1612

9.1 Sinusgenerator

9.1.1 Entwickeln Sie das Grundgerüst eines Programms für einen Sinus Generator. Hier die Beschreibung der erforderlichen Programmfunktionalität:

- In ein Feld mit 100 Elementen soll eine volle Periode der Sinusfunktion abgelegt werden Die Sinusfunktion soll eine Amplitude von 1V und einen Gleichspannungsanteil von 1,5V besitzen
- Am Ausgang des DA-Wandlers liegt die Sinusfunktion innerhalb des Spannungsbereiches von 0...3V
- Die Ausgabe der einzelnen Werte an den DA-Wandler soll innerhalb einer Timer ISR erfolgen
- Ist eine Periode ausgegeben, wird der Vorgang kontinuierlich wiederholt
- Über die Zeitdauer der Ausgabe von 100 Werten, definiert sich die Periodendauer
- Wählen Sie eine Periodendauer die einer Frequenz von 100Hz entspricht

```
1  ...
2  void print_value(void);
3
4  #define BIT_SET(a,b) ((a) |= (b))
5  #define BIT_CLR(a,b) ((a) &= ~(b))
6  #define BIT_TOGGLE(a,b) ((a) ^= (b))
7
8  int sinus[100];
9
10 //====Hauptprogramm
       ==============================================
11
12 main(void)
13     {
```

```
14          int i;
15          double step;
16  //===Hier sollten Variablen deklariert werden
        ============================
17      //unsigned char i = 0;
18      //char text[60];
19      //int x,y;
20
21  //===Hier die notwendigen Initialisierungsschritte
        ====================
22  //=(1)== Port-Initialisierung
        ========================================
23      init_Port();                    // Initialisierung der Port
            Register
24
25  //(2)=== Clock-System-Initialisierung
        ================================
26      //== XT2() oder Dco() als Taktquelle einstellen
27      //== durch Ein- oder Auskommentieren
28      //== DCO ist bei LPM Einsatz bevorzugt muß zyklisch kalibriert
            werden
29      //== XT2 ist quarzstabil muß nicht zyklisch kalibriert werden
30      //
31      //XT2 ();                       // XT2 Taktquelle aktivieren mit
            7.3728MHz
32      DCO ();                     // Dco Taktquelle aktivieren mit 7.3728
            MHz
33      //                              beachte DELTA
34
35  //=(3)== Timer-Initialisierung=
        ======================================
36      init_Timer_A();             // Init Timer für Sekundeninterrupt
37                                  // !! noch leere Funktion
38
39  //=(4)== USART-Initialisierung
        ========================================
40      init_UART1();               // UART-RS232 mit 115.2kBit/s
            initialisieren
41                                  // !! noch leere Funktion
42
43  //=(5)== CC1100-Transceiver-Initialisierung
        ============================
44      init_UART0_SPI();           // CC1100 SPI UART initalisieren
45      init_CC1100_POWERDOWN();              // CC1100 init und in RX
            Mode setzen
46                                  // !!!Interrupte sind ab jetzt
                                        freigegeben!!
47      //== Adresse und Funkkanal des Transceivers setzen
48      //== für die Arbeitsplaetze HWPx (x=1...10) sollten
49      //== ID=x und channnel=x gesetzt werden
50      ID = 1;                             // Adresse
51      setUid(ID);                         // Adresse im Transceiver
            setzen
52      channel = 1;                        // Funkkanal
53      switchFreq(channel);                // Funkkanal im Transceiver
            setzen
54      //== Soll der Transceiver genutzt werden müssen folgende zwei
```

```
        Zeilen
55      //== auskommentiert werden:
56      init_CC1100_IDLE();       // CC1100 in den IDLE Mode setzen
57      init_CC1100_POWERDOWN();// CC1100 in den PowerDown Mode setzen
58
59  //=(6)== LCD-Display-Initialisierung
        ==================================
60      dogm_reset();          // Hardware Reset des LCD Controllers
61      dogm_init();           // Initialisierung der LCD Controller
            Register
62      lcd_clear(WHITE);      // Grafikspeicher auf dem MSP430 löschen
63      //lcd_string(BLACK, 15, 25, "MSP430-GESTARTET!");    //
            Textausgabe
64      lcd_paint();           // Grafikspeicher auf das LCD Display
            ausgeben
65
66
67  #define LED_ROT (0x01)    // 0 0 1 P4.0
68  #define LED_GELB (0x02)   // 0 1 0 P4.1
69  #define LED_GRUEN (0x04)  // 1 0 0 P4.2
70  #define LED_ALL (LED_ROT | LED_GELB | LED_GRUEN)
71
72  #define LED_ON(led) (BIT_CLR(P4OUT, led))
73  #define LED_OFF(led) (BIT_SET(P4OUT, led))
74  #define LED_TOGGLE(led) (BIT_TOGGLE(P4OUT, led))
75
76  #define IS_LED_ON(led) (!(P4OUT & led))
77
78  #define TASTE_LINKS (0x1)
79  #define TASTE_RECHTS (0x2)
80
81  #define SLEEP_QUANTUM 10000
82  #define SLEEP(n) do {         /* sleep for n seconds */ \
83      long time = n * 100000; /* wait() sleeps 10*n microseconds */ \
84      while(time > SLEEP_QUANTUM) { \
85          wait(SLEEP_QUANTUM); \
86          time -= SLEEP_QUANTUM; \
87      } \
88      wait(time); \
89  } while(0)
90
91  // Schrittlaenge
92  step = (2*3.14159)/100;
93
94  // berechne Ausgangswerte fuer sinus
95  for (i = 0; i < 100; i++) {
96      double x = sin(i*step);
97      x /= 2;       // Amplitude von sinus ist 2, wir moechten 1
98      x += 1.5;     // Gleichspannungsanteil, Verschiebung entlang der
            y-Achse
99
100     sinus[i] = x * ((double)4095/(double)3);      // 3V maximale
            Ausgangsspannung, 12 Bit Aufloesung => 4096 Werte
101 }
102
103 // interrupt wird bei 0 ausgeloest
104 TBR = 1;
```

```
105
106 BIT_SET(BCSCTL2, SELS); // waehle XT2CLK als SMCLK
107
108 // waehle SMCLK (7.4 MHz ohne divider)
109 BIT_SET(TBCTL, TBSSEL1);
110 BIT_CLR(TBCTL, TBSSEL0);
111
112 // wahle count up mode
113 BIT_SET(TBCTL, MC0);
114 BIT_CLR(TBCTL, MC1);
115
116 // setze anzahl fuer interrupt
117 TBCCR0 = 740; // 7.4 MHz: takt=7_400_000 / 10000 = count to 740
118
119 // loesche interrupt flag fuer timer
120 BIT_CLR(TBCCTL0, CCIFG);
121
122 LED_OFF(LED_ALL);
123
124 // erlaube interrupt
125 BIT_SET(TBCCTL0, CCIE);
126
127 // 1x Referenzspannung, schnell
128 DAC12_1CTL = DAC12IR + DAC12AMP0 + DAC12AMP1 + DAC12AMP2 +
        DAC12SREF_2 + DAC12CALON;
129
130 _bis_SR_register(GIE);
131
132 //===Hier die Endlosschleife quasi das Betriebssystem
        ==================
133 while(1){
134 }    // Ende der Endlosschleife
135 }    // Ende Main
136 //====Ende des Hauptprogramms
        ========================================
137
138 #pragma vector = TIMERB0_VECTOR
139 __interrupt void TIMERB0(void)
140 {
141     static int i = 0;
142
143     // setze Wert fuer DA-Wandler
144     DAC12_1DAT = sinus[i++];
145     if (i > 99)
146         i = 0;
147 }
```

9.2 XY-Schreiber

9.2.1 Benutzen Sie die DA-Wandlerausgänge DAC0 (P6.6 am X-Eingang des Oszi) und DAC1 (P6.7 am Y-Eingang des Oszi), um ein Lichtpunkt über die Bildschirmfläche des Oszilloscopes zu bewegen. Zur Steuerung der Bewegungsrichtung nutzen Sie die beiden Tasten (P1.0 und P1.1) oder zwei Achsen des Beschleunigungssensors (Aufgabe 8.2).

```
1  ....
2  main(void);                    //Haupprogramm
3  void print_value(void);
4
5  #define BIT_SET(a,b) ((a) |= (b))
6  #define BIT_CLR(a,b) ((a) &= ~(b))
7  #define BIT_TOGGLE(a,b) ((a) ^= (b))
8
9  static int x, y;
10 void set_pixel(int x, int y);
11
12 typedef enum {
13     NORTH,
14     EAST,
15     SOUTH,
16     WEST
17 } direction_t;
18
19 //====Hauptprogramm
       ==============================================
20
21 main(void)
22     {
23 //===Hier sollten Variablen deklariert werden
       ==========================
24     //unsigned char i = 0;
25     //char text[60];
26     //int x,y;
27
28 //===Hier die notwendigen Initialisierungsschritte
       ======================
29 //=(1)== Port-Initialisierung
       ========================================
30     init_Port();                   // Initialisierung der Port
           Register
31
32 //(2)=== Clock-System-Initialisierung
       ================================
33     //== XT2() oder Dco() als Taktquelle einstellen
34     //== durch Ein- oder Auskommentieren
35     //== DCO ist bei LPM Einsatz bevorzugt muß zyklisch kalibriert
           werden
36     //== XT2 ist quarzstabil muß nicht zyklisch kalibriert werden
37     //
38     //XT2 ();                    // XT2 Taktquelle aktivieren mit
           7.3728MHz
```

```
39      DCO();                      // Dco Taktquelle aktivieren mit 7.3728
            MHz
40      //                              beachte DELTA
41
42  //=(3)== Timer-Initialisierung=
        ======================================
43      init_Timer_A();             // Init Timer für Sekundeninterrupt
44                                  // !! noch leere Funktion
45
46  //=(4)== USART-Initialisierung
        ======================================
47      init_UART1();               // UART-RS232 mit 115.2kBit/s
            initialisieren
48                                  // !! noch leere Funktion
49
50  //=(5)== CC1100-Transceiver-Initialisierung
        ============================
51      init_UART0_SPI();           // CC1100 SPI UART initalisieren
52      init_CC1100_POWERDOWN();            // CC1100 init und in RX
            Mode setzen
53                                  // !!!Interrupte sind ab jetzt
                                        freigegeben!!
54      //== Adresse und Funkkanal des Transceivers setzen
55      //== für die Arbeitsplaetze HWPx (x=1...10) sollten
56      //== ID=x und channnel=x gesetzt werden
57      ID = 1;                             // Adresse
58      setUid(ID);                         // Adresse im Transceiver
            setzen
59      channel = 1;                        // Funkkanal
60      switchFreq(channel);                // Funkkanal im Transceiver
            setzen
61      //== Soll der Transceiver genutzt werden müssen folgende zwei
            Zeilen
62      //== auskommentiert werden:
63      init_CC1100_IDLE();         // CC1100 in den IDLE Mode setzen
64      init_CC1100_POWERDOWN();// CC1100 in den PowerDown Mode setzen
65
66  //=(6)== LCD-Display-Initialisierung
        ==================================
67      dogm_reset();           // Hardware Reset des LCD Controllers
68      dogm_init();            // Initialisierung der LCD Controller
            Register
69      lcd_clear(WHITE);       // Grafikspeicher auf dem MSP430 löschen
70      //lcd_string(BLACK, 15, 25, "MSP430-GESTARTET!");    //
            Textausgabe
71      lcd_paint();            // Grafikspeicher auf das LCD Display
            ausgeben
72
73
74  #define LED_ROT (0x01)    // 0 0 1 P4.0
75  #define LED_GELB (0x02)   // 0 1 0 P4.1
76  #define LED_GRUEN (0x04) // 1 0 0 P4.2
77  #define LED_ALL (LED_ROT | LED_GELB | LED_GRUEN)
78
79  #define LED_ON(led) (BIT_CLR(P4OUT, led))
80  #define LED_OFF(led) (BIT_SET(P4OUT, led))
81  #define LED_TOGGLE(led) (BIT_TOGGLE(P4OUT, led))
```

```
82
83 #define IS_LED_ON(led) (!(P4OUT & led))
84
85 #define TASTE_LINKS (0x1)
86 #define TASTE_RECHTS (0x2)
87
88 #define SLEEP_QUANTUM 10000
89 #define SLEEP(n) do {        /* sleep for n seconds */ \
90      long time = n * 100000; /* wait() sleeps 10*n microseconds */ \
91      while(time > SLEEP_QUANTUM) { \
92           wait(SLEEP_QUANTUM); \
93           time -= SLEEP_QUANTUM; \
94      } \
95      wait(time); \
96 } while(0)
97
98 BIT_CLR(P1IES, TASTE_LINKS | TASTE_RECHTS); // LH
99 BIT_SET(P1IE, TASTE_LINKS | TASTE_RECHTS);
100 BIT_CLR(P1IFG, TASTE_LINKS | TASTE_RECHTS);
101
102 // 1x Referenzspannung, schnell
103 DAC12_1CTL = DAC12IR + DAC12AMP0 + DAC12AMP1 + DAC12AMP2 +
      DAC12SREF_2 + DAC12CALON;
104 // 1x Referenzspannung, schnell
105 DAC12_0CTL = DAC12IR + DAC12AMP0 + DAC12AMP1 + DAC12AMP2 +
      DAC12SREF_2 + DAC12CALON;
106
107 // interrupt wird bei 0 ausgeloest
108 TBR = 1;
109
110 BIT_SET(BCSCTL2, SELS); // waehle XT2CLK als SMCLK
111
112 // waehle SMCLK (7.4 MHz ohne divider)
113 BIT_SET(TBCTL, TBSSEL1);
114 BIT_CLR(TBCTL, TBSSEL0);
115
116 // wahle count up mode
117 BIT_SET(TBCTL, MC0);
118 BIT_CLR(TBCTL, MC1);
119
120 // setze anzahl fuer interrupt
121 TBCCR0 = 7400; // 7.4 MHz: takt=7_400_000 / 1000 = count to 7400
122
123 // loesche interrupt flag fuer timer
124 BIT_CLR(TBCCTL0, CCIFG);
125
126 // erlaube interrupt
127 BIT_SET(TBCCTL0, CCIE);
128
129 _bis_SR_register(GIE);
130
131 //===Hier die Endlosschleife quasi das Betriebssystem
      ==========
132 while(1){
133 }    // Ende der Endlosschleife
134 }    // Ende Main
```

```
135 //====Ende des Hauptprogramms
        ======================================
136
137 static direction_t cur_dir = EAST;
138
139 #pragma vector = PORT1_VECTOR
140 __interrupt void PORT1(void)
141 {
142     if (P1IFG & TASTE_LINKS) {
143         BIT_CLR(P1IFG, TASTE_LINKS);
144         switch (cur_dir) {
145             case EAST:
146                 cur_dir = NORTH;
147                 break;
148             case NORTH:
149                 cur_dir = WEST;
150                 break;
151             case WEST:
152                 cur_dir = SOUTH;
153                 break;
154             case SOUTH:
155                 cur_dir = EAST;
156                 break;
157         }
158     }
159
160     if (P1IFG & TASTE_RECHTS) {
161         BIT_CLR(P1IFG, TASTE_RECHTS);
162         switch (cur_dir) {
163             case EAST:
164                 cur_dir = SOUTH;
165                 break;
166             case NORTH:
167                 cur_dir = EAST;
168                 break;
169             case WEST:
170                 cur_dir = NORTH;
171                 break;
172             case SOUTH:
173                 cur_dir = WEST;
174                 break;
175         }
176     }
177 }
178
179 #pragma vector = TIMERB0_VECTOR
180 __interrupt void TIMERB0(void)
181 {
182     switch (cur_dir) {
183         case EAST:
184             set_pixel(x++, y);
185             break;
186         case SOUTH:
187             set_pixel(x, y--);
188             break;
189         case WEST:
190             set_pixel(x--, y);
```

```
191                 break;
192             case NORTH:
193                 set_pixel(x, y++);
194                 break;
195         }
196
197         x = (x + 4096) % 4096;
198         y = (y + 4096) % 4096;
199 }
200
201 void set_pixel(int x, int y)
202 {
203         x = (x + 4096) % 4096;
204         y = (y + 4096) % 4096;
205
206         DAC12_0DAT = y;
207         DAC12_1DAT = x;
208 }
```

9.2.2 Realisieren Sie mit einem Programm auf dem MSB430H die auf dieser Webseite gezeigte Animation einer Lissajous-Figur.

```
1   ...
2   main(void)
3       {
4           float step;
5           int i;
6   //===Hier sollten Variablen deklariert werden
        ==========================
7       //unsigned char i = 0;
8       //char text[60];
9       //int x,y;
10
11  //===Hier die notwendigen Initialisierungsschritte
        ======================
12  //=(1)== Port-Initialisierung
        ========================================
13      init_Port();                    // Initialisierung der Port
            Register
14
15  //(2)=== Clock-System-Initialisierung
        ================================
16      //== XT2() oder Dco() als Taktquelle einstellen
17      //== durch Ein- oder Auskommentieren
18      //== DCO ist bei LPM Einsatz bevorzugt muß zyklisch kalibriert
            werden
19      //== XT2 ist quarzstabil muß nicht zyklisch kalibriert werden
20      //
21      //XT2 ();                       // XT2 Taktquelle aktivieren mit
            7.3728MHz
22      DCO ();                     // Dco Taktquelle aktivieren mit 7.3728
            MHz
23      //                              beachte DELTA
24
```

```
25  //=(3)== Timer-Initialisierung=
        ========================================
26      init_Timer_A();             // Init Timer für Sekundeninterrupt
27                                  // !! noch leere Funktion
28
29  //=(4)== USART-Initialisierung
        ========================================
30      init_UART1();               // UART-RS232 mit 115.2kBit/s
            initialisieren
31                                  // !! noch leere Funktion
32
33  //=(5)== CC1100-Transceiver-Initialisierung
        ============================
34      init_UART0_SPI();           // CC1100 SPI UART initalisieren
35      init_CC1100_POWERDOWN();              // CC1100 init und in RX
            Mode setzen
36                                  // !!!Interrupte sind ab jetzt
                                        freigegeben!!
37      //== Adresse und Funkkanal des Transceivers setzen
38      //== für die Arbeitsplaetze HWPx (x=1...10) sollten
39      //== ID=x und channnel=x gesetzt werden
40      ID = 1;                             // Adresse
41      setUid(ID);                         // Adresse im Transceiver
            setzen
42      channel = 1;                        // Funkkanal
43      switchFreq(channel);                // Funkkanal im Transceiver
            setzen
44      //== Soll der Transceiver genutzt werden müssen folgende zwei
            Zeilen
45      //== auskommentiert werden:
46      init_CC1100_IDLE();         // CC1100 in den IDLE Mode setzen
47      init_CC1100_POWERDOWN();// CC1100 in den PowerDown Mode setzen
48
49  //=(6)== LCD-Display-Initialisierung
        ====================================
50      dogm_reset();           // Hardware Reset des LCD Controllers
51      dogm_init();            // Initialisierung der LCD Controller
            Register
52      lcd_clear(WHITE);       // Grafikspeicher auf dem MSP430 löschen
53      //lcd_string(BLACK, 15, 25, "MSP430-GESTARTET!");    //
            Textausgabe
54      lcd_paint();            // Grafikspeicher auf das LCD Display
            ausgeben
55
56
57  #define LED_ROT (0x01)    // 0 0 1 P4.0
58  #define LED_GELB (0x02)   // 0 1 0 P4.1
59  #define LED_GRUEN (0x04) // 1 0 0 P4.2
60  #define LED_ALL (LED_ROT | LED_GELB | LED_GRUEN)
61
62  #define LED_ON(led) (BIT_CLR(P4OUT, led))
63  #define LED_OFF(led) (BIT_SET(P4OUT, led))
64  #define LED_TOGGLE(led) (BIT_TOGGLE(P4OUT, led))
65
66  #define IS_LED_ON(led) (!(P4OUT & led))
67
68  #define TASTE_LINKS (0x1)
```

```
69  #define TASTE_RECHTS (0x2)
70
71  #define SLEEP_QUANTUM 10000
72  #define SLEEP(n) do {        /* sleep for n seconds */ \
73       long time = n * 100000; /* wait() sleeps 10*n microseconds */ \
74       while(time > SLEEP_QUANTUM) { \
75            wait(SLEEP_QUANTUM); \
76            time -= SLEEP_QUANTUM; \
77       } \
78       wait(time); \
79  } while(0)
80
81  // Schrittlaenge
82  step = (2*3.14159)/SINUS_VALUES;
83
84  // berechne Ausgangswerte fuer sinus
85  for (i = 0; i < SINUS_VALUES; i++) {
86       double x = sin(i*step);
87       x /= 2.0;    // Amplitude von sinus ist 2, wir moechten 1
88       x += 1.0;    // Gleichspannungsanteil, Verschiebung entlang der
              y-Achse
89
90       sinus[i] = x * ((double)4095/(double)3);     // 3V maximale
              Ausgangsspannung, 12 Bit Aufloesung => 4096 Werte
91  }
92  // 1x Referenzspannung, schnell
93  DAC12_1CTL = DAC12IR + DAC12AMP0 + DAC12AMP1 + DAC12AMP2 +
       DAC12SREF_2 + DAC12CALON;
94  // 1x Referenzspannung, schnell
95  DAC12_0CTL = DAC12IR + DAC12AMP0 + DAC12AMP1 + DAC12AMP2 +
       DAC12SREF_2 + DAC12CALON;
96
97  // interrupt wird bei 0 ausgeloest
98  TBR = 1;
99
100 BIT_SET(BCSCTL2, SELS); // waehle XT2CLK als SMCLK
101
102 // waehle SMCLK (7.4 MHz ohne divider)
103 BIT_SET(TBCTL, TBSSEL1);
104 BIT_CLR(TBCTL, TBSSEL0);
105
106 // wahle count up mode
107 BIT_SET(TBCTL, MC0);
108 BIT_CLR(TBCTL, MC1);
109
110 // setze anzahl fuer interrupt
111 TBCCR0 = 1; // trieggere staendig
112
113 // loesche interrupt flag fuer timer
114 BIT_CLR(TBCCTL0, CCIFG);
115
116 // erlaube interrupt
117 //BIT_SET(TBCCTL0, CCIE);
118
119 _bis_SR_register(GIE);
120
121
```

```
122 //===Hier die Endlosschleife quasi das Betriebssystem
        ==================
123 while(1){
124     float x, y;
125     static int i = 0, offset = 0;
126 
127     x = sinus[(i*2 + offset) % SINUS_VALUES];
128     y = sinus[(i*3) % SINUS_VALUES];
129 
130     set_pixel(x, y);
131 
132     i++;
133 
134     if (i >= SINUS_VALUES) {
135         i = 0;
136         offset = (offset + 2) % SINUS_VALUES;
137     }
138 
139 }   // Ende der Endlosschleife
140 }   // Ende Main
141 //====Ende des Hauptprogramms
        ====================================
142 
143 void set_pixel(int x, int y)
144 {
145     //x = (x + 4096) % 4096;
146     //y = (y + 4096) % 4096;
147 
148     DAC12_0DAT = y;
149     DAC12_1DAT = x;
150 }
```