Mikroprozessorpraktikum WS 2011/12
Aufgabenkomplex: 8

Teilnehmer:

Marco Träger, Matr. 4130515
Alexander Steen, Matr. 4357549

Gruppe: Freitag, Arbeitsplatz: HWP 1

A 8.3 Touchscreen

8.3.1 Bei Berührung des TS soll eine ISR gestartet werden, die solange die Koordinaten der Berührung anzeigt und die LED P4.1 leuchten lässt, wie der TS berührt wird.

Wir führen eine Variable `flag` ein, die anzeigen soll, ob der Touchscreen gerade berührt wird oder nicht. Diese Variable wird in der ISR für den TS gesetzt. Die Konstanten für die gemessenen Maxima und Minima wurden durch Tests ermittelt und nachträglich in den Code eingefügt.

```
#define XMAX 3700
#define XMIN 1600
#define YMAX 3700
#define YMIN 3570

char str[100];
char flag = 0x00;
```

In den Initialisierungsfunktion werden nun, wie im Beispielcode auf der Veranstaltungsseite gezeigt, die LCD-Register vorbereitend gesetzt, der Bildschirm gelöscht und Interrupts eingeschaltet. Die genauen Zuweisungen werden inline erklärt.

```
void init831() {
  // der Website entnommen:
  // LCD init:
  TS_TIP_DIR_IN;
  TS_YP_DIR_IN; TS_YM_DIR_IN;
  TS_XP_DIR_IN; TS_XM_DIR_IN;
  // das Ausgangsregister vorbereitend setzen
  TS_TIP_1; // YP Y-Achse wird ueber einen PullUp Widerstand auf
      1 gezogen
  TS_XP_1; // XP X-Achse rechts auf 1
  TS_XM_0; // XM X-Achse links auf 0
  TS_YP_1; // YP Y-Achse oben auf 1
  TS_YM_0; // YM Y-Achse unten auf 0
  // Die Ausgaenge jetzt freigeben
  TS_TIP_DIR_OUT; // YP auf 1
  TS_XM_DIR_OUT;  // XM auf 0
  // der Website entnommen ENDE

  // LCD clearen
  lcd_clear(WHITE);
  lcd_paint();
  // Interrupt fuer LCD anschalten
  P1IE |= ((0x01) << 6); // Interrupts fuer P1.6
  P1IES |= ((0x01) << 6); // HL-Flanke fuer P1.6
}
```

Initial soll die ISR bei einer HI-LO-Flanke auslösen (Touchscreen wird berührt). Die ISR für einen Touchevent entscheidet nun, ob sie ausgelöst wurde weil der TS berührt wurde oder weil er losgelassen wurde. Wurde der TS berührt ist das siebte Bit in `P1IN` nicht gesetzt. Also muss nun die LED eingeschatet werden, `flag` auf "messen"gesetzt werden und die Interruptflanke für den Loslass-Event vorbereitet werden (LO-HI-Flanke). Wurde der TS losgelassen, wird `flag` zurückgesetzt und der Ausgangszustand wiederhergestellt:

```
#pragma vector = PORT1_VECTOR
__interrupt void coord (void) {

  if (P1IFG & 0x40) {
    // TS beruehrt
    if (!(P1IN & 0x40)) {
      // Finger auf dem TS
      flag = 0x01;
      // LED P4.1 an
      P4OUT &= ~0x02;
      // Umschalten auf LO-HI
      P1IES &= ~((0x01) << 6); // LH-Flanke fuer P1.6
    } else {
      flag = 0x00;
      // Finger vom TS weggenommen
      // LED P4.1 aus
      P4OUT |= 0x02;
      // Umschalten auf HI-LO
      P1IES |= ((0x01) << 6); // HL-Flanke fuer P1.6
    }
  }
  CLEAR(P1IFG, 0xFF);
}
```

Da nun von der ISR des Touchscreens über die Variable `flag` signalisiert wird, ob die Koordinaten gemessen und angezeigt werden sollen, können wir nun in der Hauptfunktion in der main-loop die Messung starten:

```
void aufgabe831() {
  uint32_t a,b;

  lcd_clear(WHITE);
  if (flag){
    //Messung starten und ausgeben
    a = xmessen();
    b = ymessen();
    a = (a-XMIN)*128/(XMAX-XMIN);
    b = (b-YMIN)*64/(YMAX-YMIN);
    sprintf(str, "TEST␣%i.%i", (unsigned int)a,(unsigned int)b);
    lcd_string(BLACK, 10, 10, str);
  }
  lcd_paint();
}
```

Dabei wird dann pro Hauptschleifen-Durchlauf die Messung gestartet (`xmessen()` gefolgt von `ymessen`), umgerechnet und auf dem Display ausgegeben.

Die Messfunktionen für x-Achse und y-Achse sind relativ ähnlich, deshalb wird nur der Code für die Funktion `xmessen` gezeigt. Die Referenzspannung und der Inputchannel werden in dem Register `ADC12MCTL0` gesetzt, letzteres auf `A4`, da die X-Achse des Touchscreens auf P6.4 anliegt. Für die Funktion der y-Achsenmessung, ist hier `A5` ausgewählt. Nun wird in `ADC12CTL1` die Startadresse der Messung (`CSTARTADD`) auf 0 gesetzt (wir wollen das Ergebnis der Messing in `ADC12MEM0` haben), `SHS_0` gesetzt (Messung wird durch `ADC12SC`-Bit gestartet). Außerdem wird hier der interne Samplingtimer genutzt (durch das `SHP`-Bit), und zwar im Single-Channel, single-Conversion-Mode (`CONSEQ_0`).

Durch das Setzen von `ENC + ADC12SC` im `ADC12CTL0`-Register wird das Sampling und die anschließende Konvertierung gestartet.

```
uint32_t xmessen() {
  TS_XP_DIR_OUT;  // XP auf 1 freigeben
  // AVCC AVSS Referenzspannung, Inputchannel A4
  ADC12MCTL0 = SREF_0 + INCH_4;
  // ADC12 On
  ADC12CTL0 |= ADC12ON;
  // Conversion Start Adress auf 0, selber sampling ausloesen,
     ADC12OSC
  ADC12CTL1 = CSTARTADD_0 + SHS_0 +SHP + ADC12SSEL_0 +CONSEQ_0;
  // messung starten (enable, start conversion)
  ADC12CTL0 |= ENC+ADC12SC;

  while((ADC12CTL1 & 0x01)) {} // warten bis fertig
  TS_XP_DIR_IN;
  return ADC12MEM0;
}
```

Das Busy-Bit (LSB in `ADC12CTL1`) wird zyklisch abgefragt um herauszufinden, wann die Konvertierung fertig ist. Am Ende wird der Inhalt von `ADC12MEM0` zurückgegeben, da dort das Ergebnis der Messung steht.

Nun wird während der gesamten Berührung des Touchscreens die Koordinate der Berührung auf dem Display angezeigt. Nach Loslassen des Touchscreens wird nichts mehr auf dem Display angezeigt.