Wdh. SPI-Schnittstelle

Beispiel:

```
#include <LPC.214X.h>;

int main()      {
        PINSEL1 = 0x88;                 //… 1000 1000 → SCK an P0.17, MOSI an P0.19
        IODIR0 = 0x10000;               // P0.16 → digitaler Ausgang (SSEL)
        IOSET0 = 0x10000;               //Ruhezustand (HIGH-Pegel) auf SSEL
        S0SPCCR = 0x08;                 // 15 MHz / 8 = 1,875 MHz an SCK
        S0SPCR = 0xE3C;                 //…1110 0011 1100 (Wortbreite anhand von Bit 8-11,
HIGH, steigend, Master, MSBF, keine Interrupts, 14-Bit Wortbreite)

        while(1)        {
                IOCLR = 0x10000;        // Sendebeginn → SSEL auf LOW
                S0SPDR = 0x2FAC;        // 14-Bit-Wort, das übertragen wird
                while (!(S0SPSR & 0x80)) {!}    // Warten auf Ende der Übertragung
                IOSET = 0x10000;
                for (x=0, x<10000, x++) {}      //Polling bis zum nächsten Ablauf
        }
}
```

Das TFT-Display der Laborplatine

- Größe: 136x176 Pixel → 2,1“ Diagonale = 53,34 mm
- Farbmodi: 8 Bit (256 Farben) und 16 Bit (65536 Farben)
- Datenanzahl: 8 Bit → 132*176 = 23 kByte, 16 Bit → 132*176 = 46 kByte
- Kommunikation LPC2148 → Display:
    - MOSI, SCK → via PINSELX einstellen
    - SSEL, Reset, Data/Command → digitale Ausgänge (P0.0 – P0.2)

SPI-Kommunikation mit dem Display

- Display empfängt serielle Daten zu je 8 Bit
- MSBF (Bit 6 im SxSPCR = 0)
- SCK (LOW-Pegel im Ruhezustand → CPOL = 0)
- Datenübertragung bei steigender Flanke (CPHA = 0)
- Takt < 13 MHz → SXSPCCR = 0x8; (1,875 MHz)
- SSEL im Ruhezustand auf HIGH-Pegel

Data/Command → Kommandos ans Display

- Data/Command vor Beginn und während der Übertragung HIGH-Pegel
- Kommandos bestehen aus 2 Byte (→ 16 Bit)
- Sobald SSEL nach der Übertragung beider Bytes Ruhezustand (HIGH-Pegel) aufweist, wird das Kommando ausgeführt

Pixeldaten ans Display senden

- Data/Command vor Beginn und während der Übertragung auf LOW-Pegel
- Pixeldaten werden direkt in den RAM-Speicher des Displays geschrieben
- Data/Command im Ruhezustand auf HIGH-Pegel

Initialisierung des Displays

- T1TC (32 Bit) wird hochgezählt → Zahlenbereich 0 – 65535, dann Beginn wieder von 0
- PINSEL0 = PINSEL0 & FFFFCCFF (Bits P0.4 und P0.6 konfigurieren)
- Routine waitms(…):

```
void waitms(const unsigned int msWait)          {
        unsigned int aktTime, diff;                     //Variablen definieren
        aktTime = T1TC;                                 // Wert des Counters übernehmen
        do      {
                if(T1TC >= aktTime)     {
                        diff = T1TC – aktTime;
                }
                else    {
                        diff = (0xFFFFFFFF - aktTime) + T1TC;
                }
        } while(diff < msWait)
}
```

- Beispiele:

1. T1TC = 500, msWait = 75 → aktTime = 500, Schleife bis T1TC → diff = 75

2. T1TC = 0xFFFFFFFC, msWait = 75
   diff = 0xFFFFFFFF – 0xFFFFFFFC = 0x3
   anschließend else-Zweig
   diff = 0x3 + T1TC (if-Ergebnis übernehmen und Zählerstand hinzufügen)

Send Command Sequence

- &InitData[0], 2 → übergibt 2 Elemente aus InitData an Adresse [0]
- *data → Pointer zur Angabe des Elements im Array InitData[]
- int Anzahl → Anzahl der zu sendenden Elemente/Worte
- Übergang von 16-Bit Werten zu 8-Bit Werten für das Senden
  - Bsp.: 0xFDFD → LSR#8 → 0xFD → SendeByte[]
  - Verschieben um 8 Bit führt zu Weitergabe der oberen 8 Bit
  - untere 8 Bit durch UND-Operation mit 0xFF und anschließendem Senden

SPISend8Bit(SendeByte)

- Daten ins Data-Register → Master-Modus, daher sofortige Übertragung
- Dauerschleife, bis SPIF im Statusregister gesetzt ist → Rücksprung

Farbwerte einstellen

- Knallrot (1111 1000 0000 0000) → 0xF800, im 8-Bit Modus 0xE0
- Knallgrün (0000 0111 1110 0000) → 0x07E0, im 8-Bit Modus 0x1C
- Knallblau (0000 0000 0001 1111) → 0x1F, im 8-Bit Modus 0x03
- Schwarz → 0x0
- Weiß → 0xFFFF, im 8-Bit Modus 0xFF

Kommando Sequenz zur Definition eines Ausgabefensters

1. Einleitungssequenz mit Kommando
   0xEF, 0x08 (bzw. 0xEF08, wird in 8 Bit Werte aufgeteilt)

2. Setzen der Ausgaberichtung
   Hochformat: 0x1800, gedreht (180°): 0x1803
   Querformat: 0x1805, gedreht (180°): 0x1806

3. Festlegen der x_1-Koordinate des ersten Punktes
   0x12 <x_1>

4. Festlegen der x_2-Koordinate des letzten Punktes
   0x15 <x_2>

5. Festlegen der y_1-Koordinate des ersten Punktes
   0x13 <y_1>

6. Festlegen der y_2-Koordinate des letzten Punktes
   0x16 <y_2>

Beispiel für 8-Bit Farbmodus

Kommandosequenz: 0xEF08 (init), 0x1800 (Hochformat), 0x1264 (x_1=0x64), 0x1566(x_2=0x66), 0x130A (y_1=0x64), 0x160C(y_2=0x66)

Pixel-Daten: 0x0303 0x03E0 0xE0E0 0x1C1C 0x1C (erste Zeile blau, zweite Zeile rot, dritte Zeile grün)

Umrechung:
x_1 = 0x64 → 6*16 + 4*1 = 84 + 16 = 100 y_1 = 0xA = 10
x_2 = 0x66 → 6*16 + 6*1 = 84 + 18 = 102 y_2 = 0xC = 12