CT 2 - Praktikum

Ansteuerung eines TFT Displays über SPI – Teil 2
Alternativ: Mocked Display Konfiguration (ohne TFT HW)

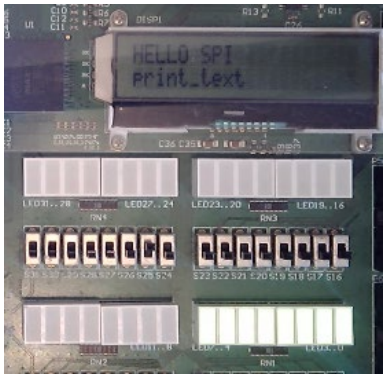
1 Einleitung

In diesem Praktikum realisieren Sie das Protokoll, um ein gegebenes TFT Display anzusteuern. Das Protokoll erlaubt es, Kommandos an das Display zu senden und Touch-Eingaben vom Display zu lesen. Um dies zu erreichen, werden die im Protokoll definierten Byte Sequenzen ausgetauscht.

Falls das **TFT Display** über SPI **verfügbar** ist, wird mit Hilfe der im letzten Praktikum implementierten SPI Funktion das Display angesteuert.

Falls das **TFT Display nicht verfügbar** ist, wird das *Mocked* Display verwendet, d.h. anstelle des physischen Displays wird in Software das Verhalten des Displays aus Protokoll-Sicht so gut wie nötig nachgebaut.

TFT-Display mit SPI Schnittstelle



Display Mock (ohne TFT Display an SPI)

Abbildung 1: TFT-Display und Mocked Display

Mocked Hardware wird oft verwendet, wo gewisse HW noch nicht verfügbar ist, wenn die SW entwickelt wird. Diese SW Modelle der HW sind nur so gut, dass man die SW testen kann. Wenn die HW dann verfügbar ist, kann man idealerweise die HW *Attrappe* (d.h. die *Mocked* HW) mit der realen HW ersetzen, und die SW kann unverändert weiterverwendet werden.

In unserem Fall testen wir, ob das Protokoll korrekt implementiert ist, unabhängig davon ob ein SPI TFT Display angeschlossen ist oder nur eine SW Attrappe davon.

2 Lernziele

- Sie können ein überliegendes Protokoll in Software realisieren, um auf dem Byte orientierten SPI die Übertragung von Zeichenfolgen mit Fehlererkennung und Empfangsbestätigung zu ermöglichen.
- Sie vertiefen Ihre Programmierkenntnisse in C. Insbesondere lernen Sie, wie ein Programm in C strukturiert werden kann.

3 Übertragungsprotokoll: Befehle an Display senden

Für die Übertragung von Befehlen und Daten vom CT Board zum Display wird auf dem SPI das in Abbildung 2 dargestellte Protokoll verwendet.

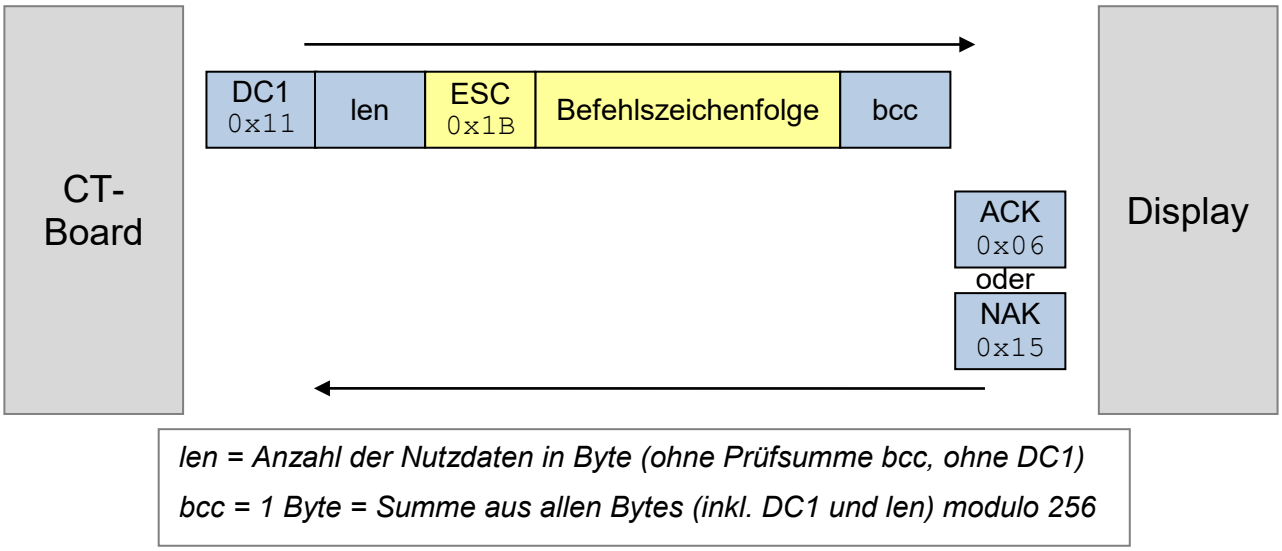


Abbildung 2: Protokoll für das Senden von Befehlen/Daten zum Display

Eingerahmt durch das Steuerzeichen **`DC1 (0x11)`**, die Anzahl der Daten **`len`** und die Prüfsumme **`bcc`** wird die jeweilige Befehlszeichenfolge übertragen. Jede Befehlszeichenfolge beginnt mit dem Steuerzeichen **`ESC (0x1B).`**

Das Display quittiert mit dem Zeichen **`ACK (0x06)`** den erfolgreichen Empfang oder mit dem Zeichen **`NAK (0x15)`** eine fehlerhafte Prüfsumme oder einen Empfangspufferüberlauf. In jedem Fall wird bei **`NAK`** das komplette Paket verworfen und muss nochmals gesendet werden. Ein **`ACK`** bestätigt lediglich die korrekte Übertragung. Ein Syntax-Check der Befehlszeichenfolge erfolgt nicht.

Beispiel: Der Befehl **`ClearDisplay`** besteht aus der Zeichenfolge „DL“ (ASCII **`0x44 0x4C`**). Das CT-Board sendet dafür die Zeichenfolge **`0x11 0x03 0x1B 0x44 0x4C 0xBF`**.

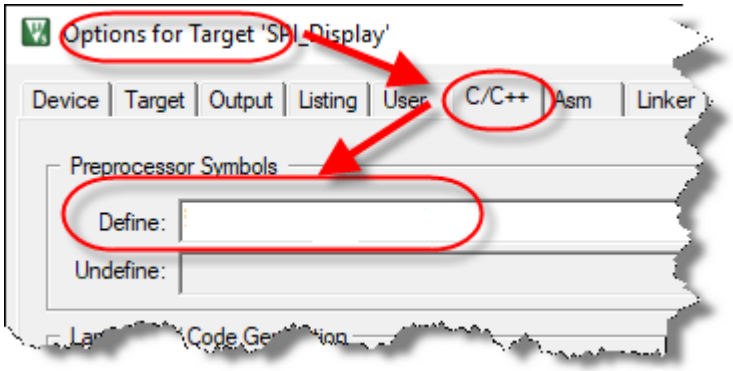
Hinweis: Das Zeichen **`ACK`** muss, anschliessend an den Sendevorgang, vom CT-Board eingelesen werden, damit der Befehl auf dem Display ausgeführt wird.

4 Programmrahmen

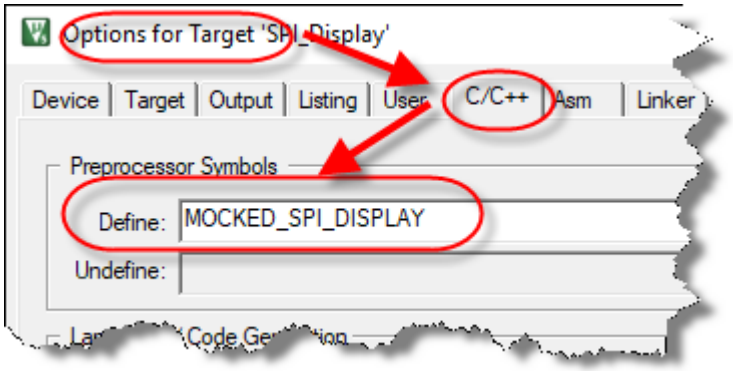
Der vorgegebene Programmrahmen ist in vier Schichten aufgeteilt. Diese Schichten und die zugehörigen Files sind im Folgenden (beginnend mit der obersten Schicht) kurz beschrieben.

- Machen Sie sich mit dem Aufbau des Programmrahmens vertraut.
- Die *header* Files enthalten Definitionen und Funktionsprototypen, welche durch das aufrufende Modul verwendet werden. Sie spezifizieren die Schnittstellen.
- Abhängig von der Verfügbarkeit des TFT Displays

  a) Wenn das TFT Display über SPI angeschlossen ist: Ergänzen Sie Ihre Files **`hal_spi.c`** und **`hal_spi.h`** aus dem letzten Praktikum. Sorgen Sie dafür, dass das Mocked Display nicht zum Zug kommt.



  b) Wenn (noch) kein TFT Display verfügbar ist, verwenden Sie das bereitgestellte **`hal_mocked.c`** und **`hal_mocked.h`** unverändert. Sorgen Sie dafür, dass das Mocked Display zum Zug kommt.



- Compilieren und linken Sie die Files.

4.1 Hauptprogramm

`main.c`

Sendet mit Hilfe der unterliegenden Schichten eine Folge von Befehlen an das Display um die Graphikschrift „HELLO SPI“ auf dem Display anzuzeigen und einen Touch-Button anzuzeigen.

4.2 Display Befehle

`cmd_lcd.h/cmd_lcd.c`

Displaybefehle bestehen aus Folgen von Zeichen (**`uint8_t`**). Ab Seite 13 im Datasheet sind die Zeichenfolgen für alle Befehle des Displays aufgelistet.

Die Funktionen dieser Files stellen die Zeichenfolgen der einzelnen Grundbefehle bereit. Sie schreiben die entsprechenden Zeichenfolgen mit Hilfe der Protokollfunktionen aus der nächsttieferen Schicht auf das Display.

4.3 Touchscreen Befehle

`cmd_touch.h/cmd_touch.c` Diese Files stellen die Funktionen zur Aktivierung und Verwendung der Touch-Funktionen des Displays bereit. Die Befehle werden mit den Protokollfunktionen aus der nächsttieferen Schicht an das Display übermittelt.

4.4 Übertragungsprotokoll

`lcd_io.h/lcd_io.c` Die Funktionen in diesen Files implementieren das in Kapitel 3 beschriebene Übertragungsprotokoll. Sie packen eine Zeichenfolge der oberen Schicht in den definierten Rahmen und senden das Ganze über SPI an das Display, bzw. entfernen den Rahmen eines auf dem SPI erhaltenen Paketes und übergeben die Nutzdaten an die obere Schicht.

4.5 SPI

`hal_spi.h/hal_spi.c` Funktionen für das Senden und Empfangen von einzelnen Bytes über SPI. Diese Schicht haben Sie bereits im letzten Praktikum implementiert.

`hal_sbuf.h/hal_sbuf.c` Behandelt das HW Signal, welches parallel zu SPI verwendet wird. Durch dieses Signal kann der SPI Slave dem SPI Master mitteilen, dass beim Slave Daten anliegen. Diese Daten sollen vom Master ausgelesen werden.

`hal_mocked.h/hal_mocked.c` Attrappe für die Display HW. Die Display HW ist für die SW nur über das SPI HAL und das SBUF HAL sichtbar – hier wird das Verhalten des TFT Displays soweit umgesetzt wie nötig, um das Protokoll gemäss Vorgaben zu testen.

5 Aufgabe: Befehle an Display senden

Im ersten Schritt möchten wir den Text **`HELLO SPI`**, je nach Variante, auf dem TFT Display oder auf dem LC-Display des CT Boards anzeigen. Zusätzlich soll, gegebenenfalls, auf dem TFT Display der Touch-Button anzeigt werden. Die Funktion dieses Touch-Buttons nimmt im *Mocked*-Fall der blaue T0 Taster auf dem CT Board wahr.

Die dazu notwendigen Befehle sind bereits in **`cmd_lcd.c`** und **`cmd_touch.h`** implementiert und werden in der Funktion **`main()`** aufgerufen. Im Programmrahmen fehlt jedoch das „Einpacken“ der Befehle ins Übertragungsprotokoll.

Implementieren Sie dazu im File **`lcd_io.c`** die Funktion

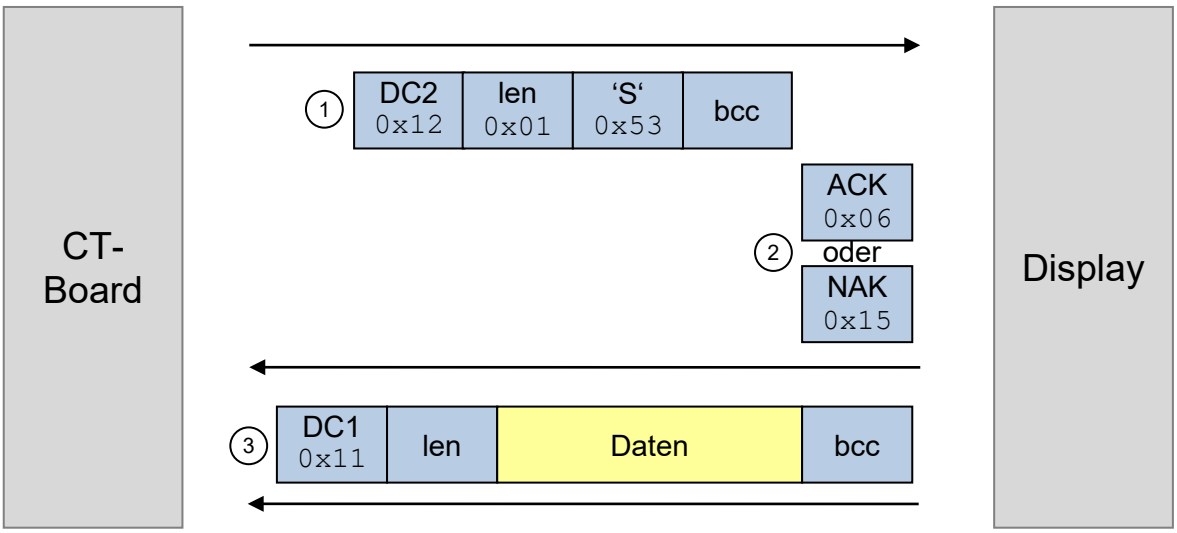
```
uint8_t write_cmd_to_display(const uint8_t *cmdBuffer,
                             uint8_t length);
```

- Lesen Sie in **`lcd_io.h`** nach, wie die Funktion beschrieben ist.
- Verwenden Sie zum Senden von einzelnen Bytes die im letzten Praktikum implementierte Funktion:
  `uint8_t hal_spi_read_write(uint8_t sendByte);`

  (Wenn das TFT Display angeschlossen ist, wird die Implementation Ihrer Funktion ausgeführt. Ansonsten wird die Implementation für die Mocked HW ausgeführt)
- Zum Empfangen verwenden Sie die gleiche Funktion. Um Daten vom Display zu empfangen, müssen Sie trotzdem ein Byte übertragen. Senden Sie den Wert 0x00:
  `uint8_t rec_byte = hal_spi_read_write(0x00);`
- Prüfen Sie Ihr Programm zusammen mit dem Display. Verwenden Sie dazu das Hauptprogramm **`main.c`**. Bei erfolgreicher Übertragung erscheint der Text **`HELLO SPI`** auf dem Display. Siehe Abbildung 1.

6 Übertragungsprotokoll: Daten aus Display auslesen

Im nächsten Schritt wollen wir die Touch Screen Funktionalität des Displays verwenden. Dazu müssen wir Daten aus dem Display auslesen können. Dazu wird auf dem SPI das in Abbildung 3 dargestellte Protokoll verwendet.



len = Anzahl der Daten in Byte (ohne Prüfsumme bcc, ohne DC1)

bcc = 1 Byte = Summe aus allen Bytes (inkl. DC1 und len) modulo 256

Abbildung 3: Protokoll für das Auslesen von Daten aus dem Display

Sobald das Display die Daten zum Senden aufbereitet hat, zieht es die **$\overline{\texttt{SBUF}}$** Leitung nach GND. Danach können die Daten eingelesen werden:

1. Die Befehlsfolge **`DC2, 0x01, 'S', bcc`** fordert die Entleerung des Sendepuffer des Displays an.
2. Die Befehlsfolge wird vom Display, wie üblich, mit einem **`ACK`** quittiert.
3. Danach beginnt das Display damit, alle gesammelten Daten, wie z.B. den Druck eines Touch-buttons, zu senden.

Die erhaltenen Daten haben dabei das folgende Format:

`ESC 'A' 0x01 0x01` Touch-button wurde gedrückt und ist jetzt *DOWN*

`ESC 'A' 0x01 0x02` Touch-button wurde gedrückt und ist jetzt *UP*

7 Aufgabe: Daten aus Display auslesen

Das abgegebene Projekt enthält bereits die Files **`cmd_touch.c/cmd_touch.h,`** welche die Befehle für die Touch-Funktionen des Displays enthalten. Im Programmrahmen fehlt jedoch das „Entpacken“ der Befehle aus dem Übertragungsprotokoll.

Implementieren Sie im File **`lcd_io.c`** die Funktion

```
uint8_t read_display_buffer(uint8_t *readBuffer)
```

und nutzen Sie diese Funktion, um in der while-Schleife des Main-Programms den Zustand des Touch-Buttons einzulesen und auf den LEDs L7 bis L0 auszugeben.

- Lesen Sie im File **`lcd_io.h`** nach, wie die Funktion beschrieben ist.
- Implementieren und verwenden Sie die Funktion **`static void send_read_display_buffer_request()`** um die Daten vom Display anzufordern, d.h. um die Befehlsfolge **`DC2, 0x01, 'S', bcc`** zu senden.
- Mit der Funktion **`uint8_t hal_sbuf_get_state (void)`** kann geprüft werden, ob im Display Daten zum Senden bereitliegen. Falls keine Daten bereitliegen, soll **`read_display_buffer()`** den Wert 0 zurückgeben ohne ein Auslesen zu starten.
- Verwenden Sie zum Empfangen der einzelnen Bytes wieder die Funktion

```
// senden sie den Wert 0 (send_byte = 0)
uint8_t hal_spi_read_write(uint8_t send_byte);
```

8 Bewertung

Die lauffähigen Programme müssen präsentiert werden. Die einzelnen Studierenden müssen die Lösungen und den Quellcode verstanden haben und erklären können.

Bewertungskriterien	**Gewichtung**
Befehle können wie in Aufgabe 5 gefordert an das Display übertragen werden.	1/2
Touch-Events können, wie in Aufgabe 7 gefordert, ausgelesen und an LEDs L7 bis L0 angezeigt werden.	1/2