Einleitung – OLED I2C-Ansteuerung

Die I2C Schnittstelle benötigt nur die beiden Signale SDA (Daten) und SCL (Clock). Diese werden hier per "Bitbanging", d.h. durch direktes Setzen / Löschen von I/O Pins erzeugt. Da hierfür weder das Universal Serial Interface (USI), noch das bei ATmega µCs in Hardware implementierte Two Wire Interface (TWI) benutzt wird, ist das Verfahren auf beliebigen µCs nutzbar. Es können hierzu 2 beliebige I/O-Portpins verwendet werden. Die Implementierung basiert auf dem mit **Atmel Application Note AVR300** gelieferten Assembler Code. Jedoch werden die lesenden Funktionen zur Steuerung des Displays nicht benötigt, sondern nur:

- I2C_init Initialisiert lediglich die für I2C benutzten µC Portpins
- I2C_start Sendet die 8-Bit I2C Slave Adresse per I2C_write (seriell) an den Bus
- I2C_write Sendet ein beliebiges Byte an den I2C Bus
- I2C_stop Gibt den I2C Bus wieder frei (beendet Verbindung)

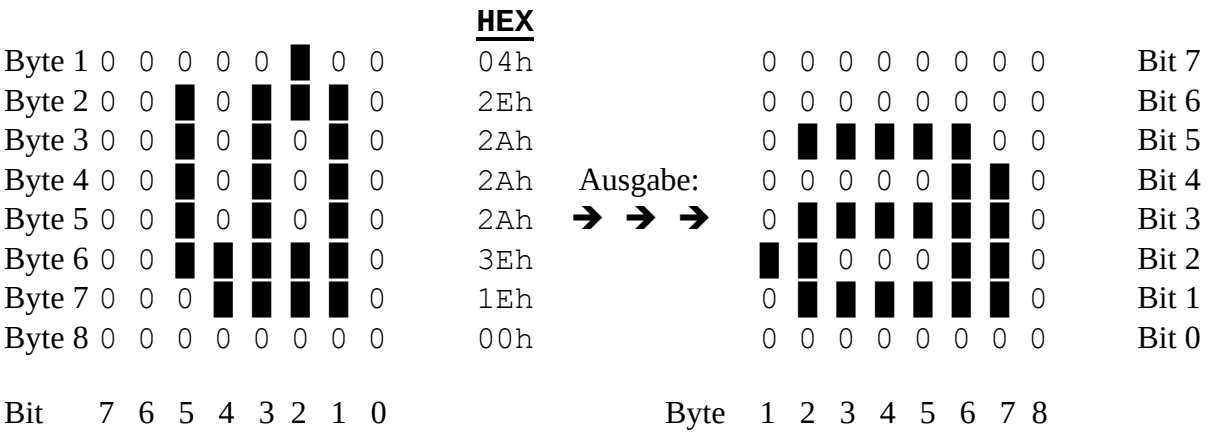
Beachte: Zwischen I2C_start und I2C_stop können beliebig viele Bytes gesendet werden.

OLED Display Ansteuerung

Das verwendete Display hat die Basisadresse 78h und verfügt über einen Command Modus und einen Data Modus (siehe Datenblatt). Der Command Modus wird durch senden von 00h, der Datenmodus durch senden von 40h eingeleitet. Ist einer der Modi aktiviert, können wieder beliebig viele Bytes übertragen werden. Die Initialisierung des Displays erfolgt im Command Modus durch Übertragung einer Byte Sequenz, die in einer Tabelle hinterlegt ist. Der Inhalt dieser Tabelle stammt aus https://www.mikrocontroller.net/topic/459397#new (siehe unten).

Zeichensatz und Bildaufbau

Es wird ein externer 8x8 Zeichnsatz verwendet, bei dem die Zeichen wie folgt abgelegt sind: Beispiel "a" = 97d in der Tabelle "font_reduced.asm":



Das Muster wird also gespiegelt und 1x um 90 Grad nach rechts gedreht um zur tatsächlichen Darstellung zu kommen. Zeilen und Spalten werden getauscht bei umgekehrter Reihenfolge. Zum Entwurf neuer Zeichen bietet sich die Nutzung einer Schablone aus Transparentpapier an. **So wie das Display initialisiert ist, erfolgt diese Spiegelung & Rotation im SSD1306 Controller; das Zeichen erscheint also "aufrecht".** Infolge der Display Auflösung von 128 x 64 Punkten, passen so 16 Zeichen in eine Zeile, bei 8 Zeilen insgesamt. Eine GotoXY Routine ermöglicht das Setzen der Ausgabekoordinate (Cursor) bezogen auf die Zeichenhöhe- und Breite. Oben links ist der Nullpunkt (X=0, Y=0). X adressiert die Spalte von 0 – 15, Y die Zeile von 0 – 7. Die Ausgabe eines Zeichens in Standardgröße erfordert also die Ausgabe von 8 Datenbytes aus der Zeichentabelle an das OLED. Etwas komplizierter wird es für Zeichen mit doppelter Höhe und Breite wie im folgenden Abschnitt beschrieben.

Zeichenausgabe doppelt hoch und doppelt breit

Verfahren:

1.) Die unteren 4 Bit jedes Bytes des Zeichens werden durch Dopplung auf 1 Byte gedehnt. Damit wird die Zeichenhöhe verdoppelt: Bit 0 (alt) ➔ Bits 0&1 (neu) etc.
2.) Die 8 so gedehnten Nibbles werden 2x nacheinander ausgegeben, womit die Zeichenbreite verdoppelt wird.
3.) Der Cursor wird per GotoXY um 1 Zeile tiefer an den Zeichenanfang gesetzt
4.) Die oberen 4 Bit jedes Bytes des Zeichens werden durch Dopplung auf 1 Byte gedehnt.
5.) Die 8 gedehnten Nibbles werden wieder 2x nacheinander ausgegeben für doppelte Breite
6.) Der Cursor wird per GotoXY in die obere Zeile hinter das doppelt breite Zeichen gesetzt. Die Ausgabe des nächsten Zeichens kann erfolgen.

Jedes Byte hat also 4 Ausgaben zur Folge. Das Verfahren bedingt, dass der Cursor zu Beginn definiert ist, und dass die Cursorposition stets aktualisiert wird. Zur Dopplung (Dehnung) der Bytes wurde die **Routine "Stretch"** vorgesehen, für die es 2 Varianten gibt: Die kürzere nutzt das Transfer Flag des Mikrocontrollers und besteht aus 4 Sequenzen der Form:

```
bst i2cdata, 0        ; Bits über Transfer Flag umkopieren
bld temp1, 0          ; Bit 0 von i2cdata => Bits 0 & 1 von temp1
bld temp1, 1
…
mov i2cdata, temp1    ; gedehntes Nibble zur Ausgabe
```

Bei der 2. Variante werden die 4 unteren Bits des auszugebenden Bytes nacheinander nach rechts in das Carry Flag geschoben. Ist Carry gesetzt, werden 2 Bits des Zielbytes gesetzt. Da das Zielbyte mit 0 initialisiert ist, müssen nur die gesetzten Bits verarbeitet werden.

```
clr temp1             ; Hilfsregister für gedehntes Byte initialisieren
lsr i2cdata           ; Bit0 => Carry
brcc stretch1         ; Sprung, wenn Bit = 0, es werden nur Einsen gesetzt
ori temp1, 0b00000011; 2 Bit (gedehnt) setzen
stretch1:
lsr i2cdata           ; Bit1 => Carry
…
Mov i2cdata, temp1    ; gedehntes Nibble zur Ausgabe
```

OLED Initialisierungssequenz

Wenn man sich auf die Einstellungen beschränkt, die nicht schon beim Reset als Standard definiert sind, kommt man mit 5 Byte zur Initialisierung aus, sofern man den (gespiegelten und gedrehten) Zeichensatz wie oben beschrieben nutzt.

Bytefolge	**Bedeutung**
0x8D 0x14	Ladungspumpe aktivieren
0xAF	Display einschalten
0xA1	Column 127 wird auf Segment 0 gemappt
0xC0	Normal Scan Direction

Diese 5 Byte werden aus einer Tabelle im Command Modus per I2C an das OLED gesendet.