TM1638-Anzeige-Taster-Modul

Das TM1638-Anzeige-Taster-Modul besitzt 8 Siebensegmentanzeigen, 8 LEDs und 8 Taster. Diese können von einem Mikrocontroller über den Baustein TM1638 angesprochen werden; die Kommunikation erfolgt hier über ein SPI-Protokoll.

Die Betriebsspannung ist 5,0 V. Signale unter 0,3 V gelten als Low, Signale über 0,7 V als High.

In diesem Beitrag behandeln wir zunächst nur die Ausgabemöglichkeiten für die Siebensegmentanzeigen; am Ende zeigen wir dann auch, wie man mit den LEDs umgeht und den Taster-Status abfragt.

Funktionsweise des TM1638-Anzeige-Moduls

Zur Ansteuerung der Siebensegmentanzeigen und LEDs benutzt der TM1638 einen Displayspeicher mit 16 Registern vom Typ Byte: Jedes dieser Register ist einer LED bzw. einer Siebensegmentanzeige zugeordnet (s. Abb. 1). Der Inhalt des Registers legt bei einer LED fest, ob sie leuchtet oder nicht, bei einer Siebensegmentanzeige, welche Segmente aktiviert werden. Das Register mit der Adresse 4 ist z. B. der Siebensegmentanzeige Nr. 3 (von links) zugeordnet.

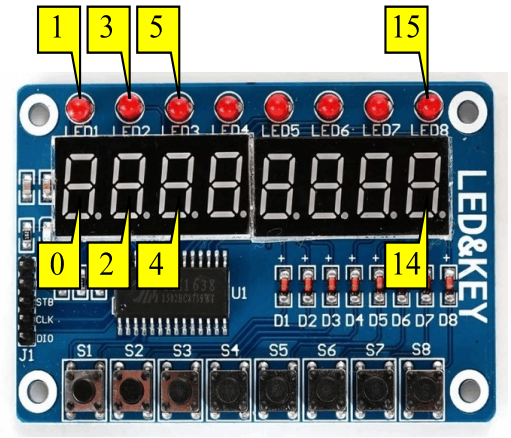

Abbildung 1

Soll in dieser Siebensegmentanzeige z. B. die Zahl 1 erscheinen, müssen dort die Segmente b und c aktiviert werden (vgl. Abb. 2). Dazu muss man das entsprechende Bitmuster &B00000110 in das Register mit der Adresse 4 speichern. Weiter unten werden wir genau beschreiben, welche Befehle und Daten der Mikrocontroller dazu an den TM1638-Baustein senden muss.

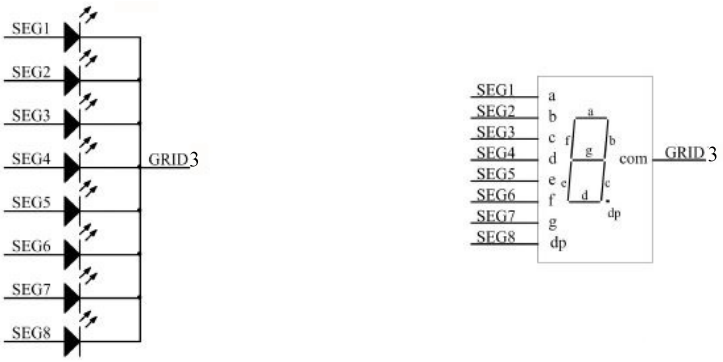

Abbildung 2

Nun ist nicht jede Siebensegmentanzeige direkt über 8 eigene Datenleitungen mit seinem zugehörigen Displayregister verbunden. Vielmehr besitzt der Displayspeicher für die Ansteuerung der Siebensegmentanzeigen einen gemeinsamen Datenbus (mit den Leitungen SEG1 bis SEG8); an diesen liegen immer die Signale von dem Register an, welches gerade adressiert ist. In unserem Beispiel ist es das Bitmuster [Bit0 : Bit7] aus dem Register mit der Adresse 4.

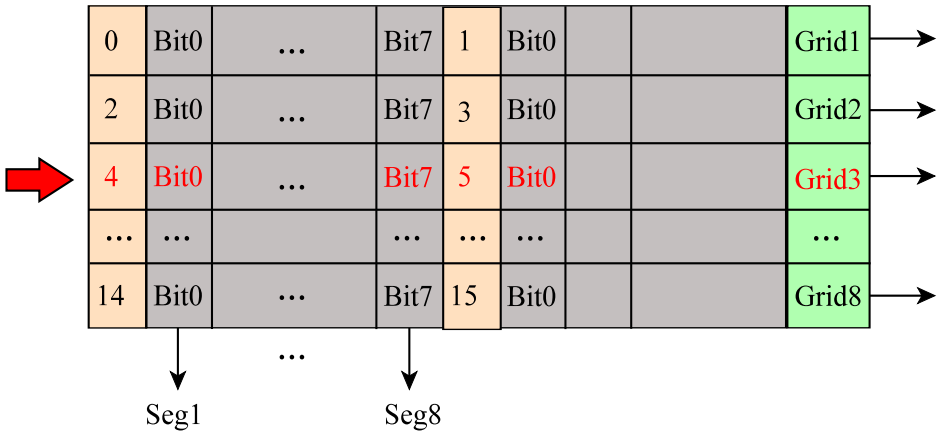

Abbildung 3: Der Adresszeiger weist hier auf das Doppelregister [4 : 5].

Abb. 3 macht deutlich, dass der Displayspeicher in so genannten Doppelregistern organisiert sind: Mit dem Register Nr. 4 wird gleichzeitig auch das Register Nr. 5 adressiert. Da das letztere einer LED zugeordnet ist, werden wir uns erst weiter unten darum kümmern.

In rascher Abfolge adressiert der TM1638 nun automatisch die einzelnen Register, um deren Inhalte an den Datenbus ausgegeben. Nun muss nur noch dafür gesorgt werden, dass jede Siebensegmentanzeige nur das für sie bestimmten Bitmuster anzeigt. Dies geschieht durch ein so genanntes ***Multiplexen***.

Für dieses Multiplexen ist die gemeinsame Kathode einer Siebensegmentanzeige nicht an Masse angeschlossen, sondern an einen der Anschlüsse Grid1 bis Grid8 des TM1638 (vgl. Abb. 2). Die Leuchtdioden einer Siebensegmentanzeige können deswegen nur dann zum Leuchten gebracht werden, wenn der zugehörige Grid-Anschluss auf Low liegt.

Standardmäßig liegen die Grid-Anschlüsse auf High. Der TM1638-Baustein sorgt nun dafür, dass immer nur die zu einem aktivierten Register zugehörige Gridleitung auf Low gelegt wird. Das bedeutet: Ist z. B. das Register 4 adressiert, dann liegt am Datenbus das passende Bitmuster für unsere Siebensegmentanzeige Nr. 3 an; und weil nur die Leitung Grid3 auf Low ist (vgl. Abb. 3), werden auch nur die entsprechenden Segmente der Anzeige Nr. 3 zum Leuchten gebracht (vgl. Abb. 2).

Nun adressiert der TM1638 immer wieder in rascher Folge die einzelnen Displayregister. Dies macht er autonom, d.h. ohne dass wir dazu weitere Maßnahmen ergreifen müssen. Dadurch werden die Bitmuster der einzelnen Register immer wieder auf den zugehörigen Siebensegmentanzeigen kurzzeitig zur Anzeige gebracht. Wegen der Trägheit des Auges sehen wir die Siebensegmentanzeigen allerdings fortwährend leuchten.

Abb. 4 zeigt unten das Signal von Grid1: Jede 5 ms ist es für 0,5 ms auf Low, die restliche Zeit auf High. Nur in dieser Low-Phase werden die LEDs von der linken Siebensegmentanzeige (Nr. 1) gemäß dem Bitmuster in Adresse 0 zum Leuchten gebracht. In der restlichen Zeit sind sie aus. In dieser Restzeit werden nun nacheinander die restlichen 7 Siebensegmentanzeigen auf die gleiche Art und Weise angesteuert: In Abb. 4 ist oben das Signal von Grid3 dargestellt, welches im dritten Zeitschlitz unsere Siebensegmentanzeige Nr. 3 aktiviert.

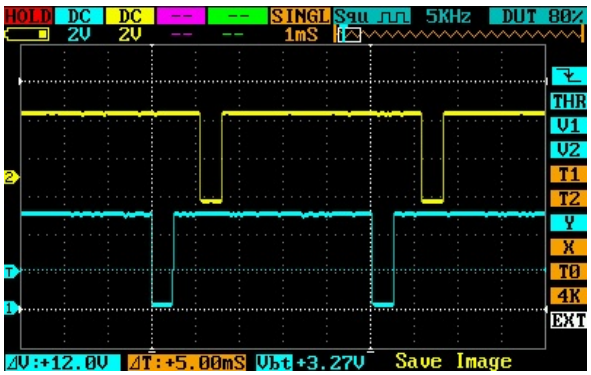

Abbildung 4: Grid1 (unten), Grid3 (oben)

Die Wiederholfrequenz ist $(5\ \text{ms})^{-1} = 200$ Hz. Das Auge nimmt das Flackern normalerweise nicht war. Wenn man das TM1638-Anzeige-Taster-Modul allerdings sehr schnell hin- und herbewegt, kann man allerdings auch mit dem bloßen Auge erkennen, dass die einzelnen Segmente nicht kontinuierlich leuchten.

Die Wiederholfrequenz ist immer konstant; dagegen kann man mit entsprechenden Befehlen die Pulsdauer verkleinern (s. u.). Durch diese Pulsweitenmodulation (PWM) kann die Helligkeit verringert werden.

Steuerung des TM1638-Bausteins

Kommen wir jetzt zu dazu, wie man den TM1638-Baustein mit einem Mikrocontroller ansteuert. Zunächst werden wir uns nur um die Siebensegmentanzeigen kümmern.

Da der TM1638-Baustein viele unterschiedliche Funktionen besitzt, reicht es nicht aus, ihm die Bitmuster für die Siebensegmentanzeigen zu übertragen. Neben diesen Daten muss man ihm vielmehr auch Steuerbefehle (commands) übermitteln. Sowohl Daten als auch Steuerbefehle werden mit dem SPI-Protokoll übertragen (s. z. B. http://www.forum.g-heinrichs.de). Die Taktsignale erhält der TM1638 (Slave) über den Eingang CLK vom Mikrocontroller (Master); die maximale Taktfrequenz ist 1 MHz. Damit der Baustein die einzelnen Bits empfangen kann, muss der STB-Eingang des Bausteins auf Low (Chip Selected) sein. Man beachte, dass der TM1638-Baustein bei der seriellen Übertragung der einzelnen Bits erwartet, dass das LSB (Least Significant Bit) zuerst gesendet wird. Will man die USI-Einheit des Attiny2313 ausnutzen, dann muss man PortB.7 (USCK) und PortB.6 (DO) des Attiny mit den Anschlüssen CLK bzw. DIO bei dem MT1638-Modul verbinden.

Es gibt drei verschiedene Steuerbefehle. Allen ist gemeinsam, dass sie durch eine negative Flanke am STB-Eingang eingeleitet werden müssen; das unterscheidet sie gerade von den Datenbytes.

Setting of Data Command (B7 = 0; B6 = 1; Restliche Bits: Parameter)
Setting of Address Command (B7 = 1; B6 = 1; Restliche Bits: Parameter)
Display Control Command (B7 = 1; B6 = 0; Restliche Bits: Parameter)

Man erkennt unschwer, dass die Steuerbefehle jeweils durch die beiden Bits B6 und B7 gekennzeichnet sind.

Nun geben wir die für die Ansteuerung der Siebensegmentanzeigen wichtigen Parameter der Steuerbefehle an.

Setting of Data Command

B7	B6	B5	B4	B3	B2	B1	B0	Beschreibung
0	1	0	0	0	0	0	0	Daten schreiben; autom. Inkrement
0	1	0	0	0	1	0	0	Daten schreiben; feste Adresse

Im ersten Fall (&B01000000) wird der TM1638 so eingestellt, dass er Daten empfängt und in den Displayspeicher schreibt; dabei wird der Adresszeiger nach jedem empfangenen Byte automatisch um 1 erhöht. Dieser Adresszeiger weist nicht auf die Doppelregister wie in Abb. 3, sondern auf die einzelnen Register. Dieser Modus eignet sich z. B. für das Löschen des gesamten Displayregisters.

Im zweiten Fall (&B0100 0100) wird der TM1638 so eingestellt, dass zu jedem Daten-Byte die zugehörige Adresse angegeben werden muss.

Die Angabe der Adresse geschieht mit dem **Setting of Address Command**

B7	B6	B5	B4	B3	B2	B1	B0	Beschreibung
1	1	0	0	x3	x2	x1	x0	setzt den Adresszeiger auf &Bx3x2x1x0

Beim Einschalten (power-up) besitzt der Adresszeiger den Wert 0.

Um das Display ein- oder auszuschalten, aber auch um die Helligkeit der Segmente zu steuern, wird das **Display Control Command** eingesetzt.

B7	B6	B5	B4	B3	B2	B1	B0	Beschreibung
1	0	0	0	1	0	0	0	setzt PWM-Pulsbreite auf 1/16
1	0	0	0	1	0	0	1	setzt PWM-Pulsbreite auf 2/16
1	0	0	0	1	0	1	0	setzt PWM-Pulsbreite auf 4/16
								weitere Pulsweiten s. Datasheet
1	0	0	0	0	0	0	0	Display aus
1	0	0	0	1	1	1	1	Display an

Beispiele

1. Ziffer “1” in der Siebensegmentanzeige Nr. 3 zur Anzeige bringen (*w* = write)

STB = 1
STB = 0 erzeugt negative Flanke
w &B01000100 Schreibmodus; feste Adresse
STB = 1
STB = 0 erzeugt negative Flanke
w &B11000110 Adresszeiger auf &B0110 (6)

w &B00000110 Datenbyte (Bitmuster entspricht der Ziffer 1)
STB = 1
STB = 0 erzeugt negative Flanke
w &B10001111 Display einschalten

2. Display ausschalten

STB = 1
STB = 0 erzeugt negative Flanke
w &B10000000 Display ausschalten

3. Alle Displayregister löschen (nach power-up empfohlen)

STB = 1
STB = 0 erzeugt negative Flanke
w &B01000100 Schreibmodus; autom. Inkrement
STB = 1
STB = 0 erzeugt negative Flanke
w &B11000000 Adresszeiger auf &B0000 (Startadresse 0)
Wiederhole 16 mal:
w &B00000000 Datenbyte 0 (kein Segment aktivieren)

Ansteuern der einzelnen LEDs

Soll z. B. die LED Nr. 3 (von links) eingeschaltet werden, muss man in das Displayregister mit der Adresse 5 das Bitmuster &B00000001 schreiben. Wenn dann das entsprechende Doppelregister (vgl. Abb. 5) aktiviert ist, liegt an der Leitung Seg9 ein High-Signal und (nur) an Grid3 ein Low-Signal; dadurch wird (nur) die LED Nr. 3 zum Leuchten gebracht.

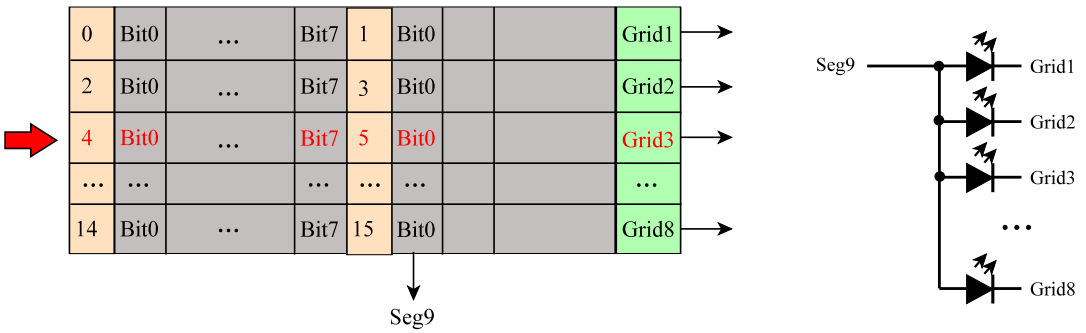

Abbildung 5

Wenn nun das Bit0 in diesem Displayregister auf 0 gesetzt wird, liegt an der Leitung Seg9 ein Low-Signal an und die LED Nr. 3 geht aus.

Die Ansteuerung der LEDs erfolgt mit denselben Befehlen wie bei den Siebensegmentanzeigen. Natürlich sind hier düe die Datenbytes nur die Werte 0 und 1 sinnvoll.

Abfragen des Taster-Status

Die 8 Taster unseres TM1638-Anzeige-Taster-Moduls sind wie in Abb. 6 dargestellt auf der einen Seite über Dioden mit den Leitungen Seg1 ... Seg8 des Datenbusses und auf der anderen mit dem Anschluss K3 des TM1638 verbunden.

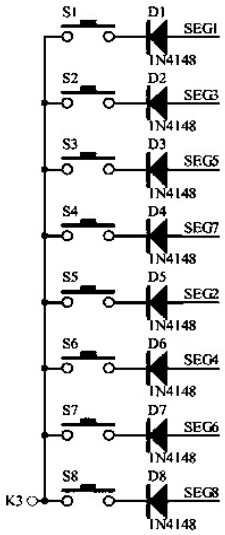

Abbildung 6

Bei dem Anschluss K3 handelt es sich um einen Eingang. Unbeschaltet liegt er auf Low; wenn einer der Taster S1 ... S8 gedrückt ist, übernimmt er das Potenzial der zugehörigen Seg-Leitung (vgl. Abb. 6).

Um damit einen gedrückten Taster zu identifizieren, geht der TM1638 nun folgendermaßen vor: Abb. 4 zeigt, dass ein Anzeigezyklus genau 5,0 ms dauert. Dabei stehen für jede Siebensegmentanzeige und ihrer zugeordneten LED jeweils ein Zeitschlitz von 0,5 ms zur Verfügung. Insgesamt werden also bei jedem Zyklus für die Anzeige 8 · 0,5 ms = 4,0 ms benötigt. In der verbleibenden Zeit sind alle Grid-Leitungen auf High; damit sind in dieser Phase alle Siebensegmentanzeigen und LEDs ausgeschaltet.

Genau diese Phase wird nun benutzt, um die Taster zu “scannen”. Dazu legt der TM1638-Baustein an die Leitungen Seg1 ... Seg8 nacheinander für kurze Zeit ein High-Signal. Abb. 7 zeigt solche Signale an den Leitungen Seg1 und Seg2. Man erkennt, dass die Signale sehr kurz sind, nämlich 70 us. Im ersten Zeitintervall kann damit der Zustand von Taster S1 erfasst werden, im zweiten Intervall der Zustand von Taster S2 u.s.w. Der Scanvorgang dauert insgesamt 560 us und ist somit abgeschlossen, wenn der nächste Anzeigezyklus beginnt.

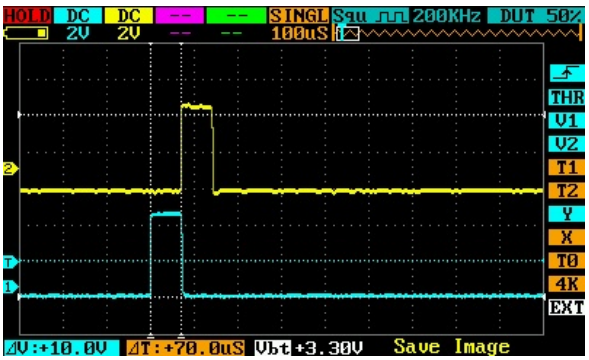

Abbildung 7

Das Ergebnis eines solchen Scan-Vorgangs hält der TM1638 in einem Scan-Speicher fest. Dieses besteht aus den 4 Bytes SR1 ... SR4:

B0	**B1**	**B2**	**B3**	**B4**	**B5**	**B6**	**B7**	
S1				S5				**SR1**
S2				S6				**SR2**
S3				S7				**SR3**
S4				S8				**SR4**

Der TM1638-Baustein kann noch insgesamt 16 weitere Taster scannen; diese werden dann an die Eingänge K1 bzw. K2 angeschlossen. Die Ergebnisse dieser Scanvorgänge findet man in den Bits B1 und B2 sowie B5 und B6. Auch hier steht eine 1 jeweils für einen gedrückten Taster.

Die hier dargestellte Scanmethode liefert nur dann korrekte Ergebnisse, wenn nicht zwei oder mehr Tasten gleichzeitig gedrückt sind.

Steuern des Scanvorgangs

Zum Lesen der 4 Scan-Register senden wir dem TM1638 zunächst den Lesebefehl &B01000010. Wie bei den bereits vorgestellten Setting of Data Commands muss auch er durch eine negative Flanke bei STB eingeleitet werden (*r&s* = read and store):

STB = 1
STB = 0
w &B01000010 Lesebefehl
warte 1 us
ggf. Datenrichtungsregister beim Mikrocontroller anpassen (s. u.)
r&s ->SR(1)
r&s ->SR(2)
r&s ->SR(3)
r&s ->SR(4)
STB = 1
ggf. Datenrichtungsregister beim Mikrocontroller zurücksetzen (s. u.)

Die Inhalte der vier Scan-Register befinden sich nun in einer Array-Variablen SR und können nun je nach Bedarf ausgewertet werden.

Zuletzt noch einige Bemerkung zum Datenrichtungsregister: Benutzt man zur Datenübertragung nur einen einzigen Portanschluss des Mikrocontrollers, so muss vor dem Ausführen der SPI-

Lesesequenz dieser Anschluss als Eingang und am Ende wieder als Ausgang konfiguriert werden. Bei der Hardware-USI des Attiny 2313 gibt es für die Ausgabe und die Eingabe getrennte Anschlüsse: DO (= PortB.6) bzw. DI (PortB.5). Zusätzlich zum Anschluss DO verbinden wir hier auch DI mit dem Anschluss DIO des TM1638-Moduls. Damit das Eingangssignal bei DI nicht durch DO gestört wird, muss für die Dauer der *r&s*-Sequenzen PortB.6 als Eingang konfiguriert werden (DDRB.6 = 0). PortB.5 ist natürlich dauerhaft als Eingang eingestellt.

SPI-Funktion (BASCOM) mit USI

Die USI des Attiny ist für eine serielle Datenübertragung konzipiert, die mit dem MSB (Most Significant Bit) beginnt. Um die USI-Befehle dennoch benutzen zu können, kehren wir die Bitreihenfolge vor und nach der Übertragung einfach um.

```
Function Spi(s As Byte) As Byte
  For K = 0 To 7                                        'Umkehrung
    K1 = 7 - K
    S1.k = S.k1
  Next K
  Usidr = S1
  Usisr = &B01000000                                    'USIOIF und Zähler löschen
  Usicr = &B00011010                                    'USIWM0, USICS1,USICLK auf 1
  Waitus 1
  Do
    Usicr.usitc = 1                                     'Takten durch Setzen von usitc
    Waitus 1                                            'später weg!
  Loop Until Usisr.usioif = 1                           '16 mal toggeln
  S1 = Usidr
  For K = 0 To 7                                        'Umkehrung
    K1 = 7 - K
    S1.k = Usidr.k1
  Next K
  Spi = S1
End Function
```