MPX_8DIGIT

02.02.2022 by R. Seelig

MPX_8DIGIT ist eine Library zum Multiplexen einer 8 stelligen 7-Segmentanzeige mit gemeinsamer Kathode sowie dem gemultiplexten Einlesen von 8 Tasten an einem 28 pol. ATmegaxx8 Microcontroller.

Da für das Multiplexen der Anzeige sowie der Tasten insgesamt 20 Anschlüsse benötigt werden und 2 Anschlüsse für eine eventuelle Kommunikation, bspw. über den UART, frei bleiben sollen, werden alle I/O-Pins inklusive der Pins für einen externen Quarz des Mikrocontroller benötigt!

Aus diesem Grund funktioniert diese Library nur mit einem auf internen 8 MHz Takt gefusten Controller.

Zu setzende Fuses sind für die Controller ATmega48, ATmega88, ATmega168 und ATmega328p identisch:

Lo-Fuse	Hi-Fuse	Ext-Fuse
0xE2	0xDF	0xFF

Funktionsweise

Die Ansteuerung der Anzeige (multiplexen) sowie das Einlesen der Tasten geschieht ausschließlich in der Interruptroutine des Timer 1. Diese Interruptroutine wird periodisch automatisch im Intervall aufgerufen.

Damit alle am Multiplexing beteiligten Anschlüsse sowie der Timerinterrupt initialisiert werden, muß einmalig im `main` – Programm

```
mpx8_init();
```

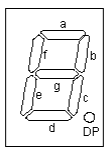
aufgerufen werden.

Die Anzeige

In der Library ist ein globales Array definiert,

```
uint8_t seg7_fb[8];
```

welches als Framebuffer fungiert. In diesem Buffer wird für jedes einzelne Segment eines Digits angegeben, welches leuchten soll und welches nicht. `seg7_fb[0]` addressiert Digit 0, `seg7_fb[7]` das Digit 7. Die Interruptroutine bildet den Inhalt des Framebuffers automatisch auf der Anzeige ab. Um etwas zur Anzeige zu bringen ist also lediglich das Beschreiben dieses Framebuffers erforderlich.

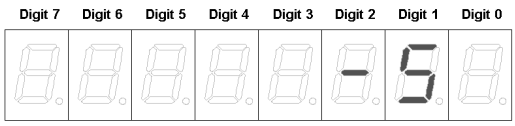


Bitnummer	7	6	5	4	3	2	1	0	
Segment	DP	g	f	e	d	c	b	a	
Beispiele									
"-"	0	1	0	0	0	0	0	0	=> 0x40
"5"	0	1	1	0	1	1	0	1	=> 0x6D

Ein einfaches Schreiben auf den Framebuffer:

```
seg7_fb[2]= 0x40;
seg7_fb[1]= 0x6d;
```

läßt auf der Anzeige -5 auf den Digits 2 und 1 erscheinen:



Bitmuster für alphanumerische Ziffern sind bereits im Array

```
uint8_t seg7_bmp[16];
```

definiert. Eine Zuweisung an den Framebuffer mittels

```
seg7_fb[1]= seg7_bmp[5];
```

würde ebenfalls eine 5 auf dem Digit 1 anzeigen.

Dezimalpunktsteuerung

Eine weitere globale Variable

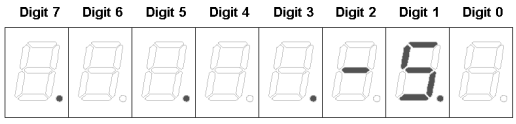
```
uint8_t seg7_dpmask;
```

bestimmt ob und an welcher Stelle ein Dezimalpunkt (DP) angezeigt werden soll. Bit 0 schaltet den Dezimalpunkt des Digits 0 an / aus, Bit 7 den des Digits 7.

Bitnummer	7	6	5	4	3	2	1	0	
seg7_dpmask	1	0	1	0	1	0	1	0	=> 0xAA

Programmbeispiel:

```
seg7_fb[2]= 0x40;          // "-" Zeichen
seg7_fb[1]= 0x6d;          // "5" Zeichen
seg7_dpmask= 0xaa;         // 4 Dezimalpunkte setzen
```



Helligkeitssteuerung

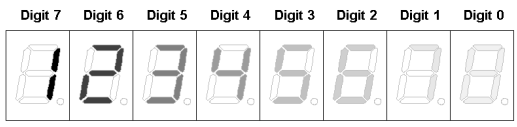
Jedes einzelne Digit (nicht ein einzelnes Segment eines Digit) kann in seiner Helligkeit gedimmt werden. Dieses ist insofern praktisch, wenn die 8-stellige Anzeige aus mehreren, eventuell aus unterschiedlichen Farben, bestehenden Anzeigen besteht. Zu diesem Zweck steht ein weiteres globales Array

```
uint8_t dig_pwm[8];
```

zur Verfügung. Das Dimmen wird innerhalb der Interruptroutine durch eine softwarerealisiert Pulseweitenmodulation (PWM) realisiert. Gültige Helligkeitsstufen sind im Bereich von 1 bis 16 (inklusive).

Programmbeispiel:

```
dig_pwm[7]= 1;
dig_pwm[6]= 3;
dig_pwm[5]= 5;
dig_pwm[4]= 7;
dig_pwm[3]= 9;
dig_pwm[2]= 11;
dig_pwm[1]= 13;
dig_pwm[0]= 16;
fb_setdezint32(12345678);
```



Die Zahl `12345678` wird mit abnehmender Helligkeit zur Anzeige gebracht.

Die Taster (keys)

Alle Taster werden in die globale Variable

```
uint8_t key_stat;
```

eingelesen. Diese Variable wird durch den Timerinterrupt ständig aktualisiert. Bit 0 der Variable beinhaltet den Status des Tasters 0, Bit 7 den Status des Tasters 7. Hiermit können somit auch mehrere gleichzeitig gedrückte Taster erfasst werden.

Für eine vereinfachte Abfrage der Tasten sind in mpx_8digit.h die Scancodes mittels

```
#define key0 bis #define key7
```

hinterlegt.

Programmbeispiel:

```
while(1)
{
  delay(30);
  switch (key_stat)
  {
    case key0 : seg7_fb[0]= seg7_bmp[1]; break;
    case key1 : seg7_fb[0]= seg7_bmp[2]; break;
    case key2 : seg7_fb[0]= seg7_bmp[3]; break;
    case key3 : seg7_fb[0]= seg7_bmp[4]; break;
    case key4 : seg7_fb[0]= seg7_bmp[5]; break;
    case key5 : seg7_fb[0]= seg7_bmp[6]; break;
    case key6 : seg7_fb[0]= seg7_bmp[7]; break;
    case key7 : seg7_fb[0]= seg7_bmp[8]; break;
    default   : seg7_fb[0]= seg7_bmp[0]; break;
  }
}
```

Mit obigem Programmausschnit wird auf der Anzeige die Taste angezeigt, die gedrückt ist (allerdings ist hiermit keine Auswertung gleichzeitig gedrückter Tasten möglich).

Benutzerfunktionen

mpx8_init

Prototyp: `void mpx8_init(void);`

initialisiert Timer1 Compare Interrupt Modus sowie alle am Multiplexing beteiligten I/O Pins

Programmbeispiel:

```
mpx8_init();
```

Framebuffer

Um den Umgang mit der Anzeige dem Benutzer zu erleichtern sind in der Library Framebuffer-Funktionen enthalten, die auf sehr einfache Weise numerische wie auch alphanumerische Zahlen zur Anzeige bringen.

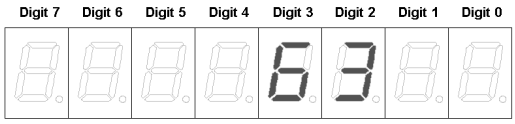
fb_setdez2

Prototyp: `void fb_setdez2(uint8_t val, uint8_t dpos);`

beschreibt den Framebuffer mit einem 2 stelligen Dezimalwert (0..99) ab Position dpos (POS 0 entspricht rechter Anzeigeposition)

Programmbeispiel:

```
fb_setdez2(63, 2);
```



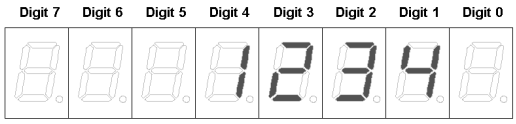
fb_setdezint

Prototyp: `void fb_setdezint(uint16_t val, uint8_t dpos);`

beschreibt den Framebuffer mit einem 4 stelligen Dezimalwert (16 Bit) ab Position dpos (POS 0 entspricht rechter Anzeigeposition)

Programmbeispiel:

```
fb_setdezint(1234, 1);
```



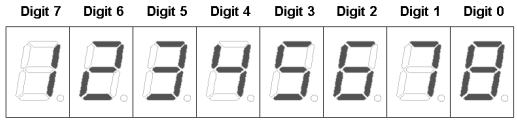
fb_setdezint32

Prototyp: `void fb_setdezint32(uint32_t val);`

beschreibt den Framebuffer mit einem 8 stelligen Dezimalwert (32 Bit)

Programmbeispiel:

```
fb_setdezint32((uint32_t)12345678);
```



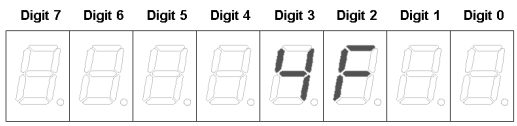
fb_sethexbyte

Prototyp: `void fb_sethexbyte(uint8_t val, uint8_t dpos);`

beschreibt den Framebuffer mit einem 2 stelligen Hexadezimalwert (8 Bit) ab Position dpos (POS 0 entspricht rechter Anzeigeposition)

Programmbeispiel:

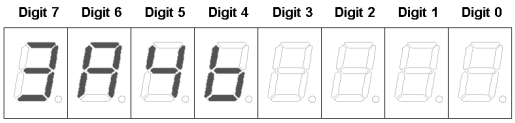
```
fb_sethexbyte(0x4f, 2);
```



fb_sethexint

Prototyp: `void fb_sethexint(uint8_t val, uint8_t dpos);`

beschreibt den Framebuffer mit einem 4 stelligen Hexadezimalwert (16 Bit) ab Position dpos (POS 0 entspricht rechter Anzeigeposition)



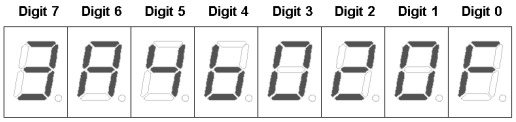
Programmbeispiel:

```
fb_sethexint(0x3a4b, 4);
```

fb_sethexint32

Prototyp: `void fb_sethexint32(uint32_t val);`

beschreibt den Framebuffer mit einem 8 stelligen Hexadezimalwert (32 Bit)



Programmbeispiel:

```
fb_setdezint32((uint32_t)0x3a4b020f);
```

Hardwareanbindung

Im Header `mpx_8digit.h` wird die Zuordnung der Controllerpins zu den Anschlüssen der 7-Segmentanzeige sowie den Anschlüssen zu den Tastern angegeben. Hier ist eine sehr einfache Anpassung an einen Hardwareaufbau möglich.

`kseg0` bis `kseg7` bezeichnen die Multiplexleitungen für die Anzeige, wobei `kseg0` und `kseg1` gleichzeitig auch die Multiplexleitungen für die 2 x 4 Tastenblöcke sind. Die `kseg` Anschlüsse werden mit der gemeinsamen Kathode eines Digits verbunden. Im Falle von `kseg0` und `kseg1` werden diese zusätzlich mit den gemeinsamen Anschlüssen von Tastenblock 0 und Tastenblock 1 angeschlossen (siehe Schaltplan).

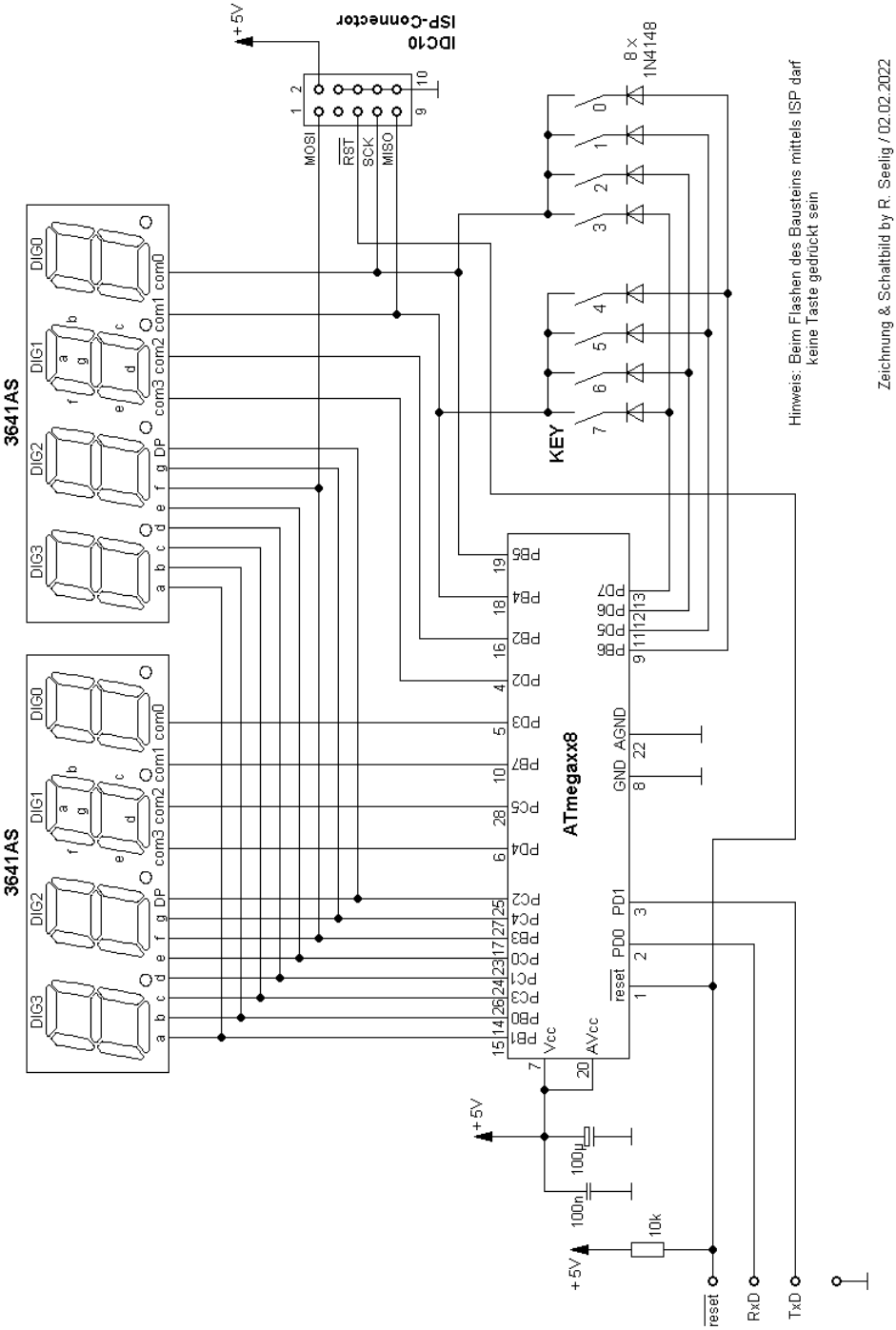
Beispiel:

```
#define kseg0            B5
```

Die gemeinsame Kathode des Digits 0 sowie die gemeinsamen Anschlüsse des Tastenblocks 0 werden an den Controllerpin PB5 (Pinnummer 19) angeschlossen.

Achtung: Für den Define-Wert darf nur B5 und nicht PB5 angegeben werden !

Schaltplan


+5V
ISP-Connector
IDC10
MOSI
RST
SCK
MISO
8 x
1N4148
KEY
3641AS
3641AS
DIG0
DIG1
DIG2
DIG3
ATmegaxx8
Vcc
AVcc
GND
AGND
reset
PD0
PD1
100n
100µ
10k
+5V
reset
RxD
TxD
Hinweis: Beim Flashen des Bausteins mittels ISP darf keine Taste gedrückt sein
Zeichnung & Schaltbild by R. Seelig / 02.02.2022