Praktikum EGI SS2007

Versuch 10 Versuchsprotokoll

Moritz Venn

29. Juni 2007

Inhaltsverzeichnis

1 Allgemein **2**

2 10.2 **2**

2.1 10.2.1 1 2

3 10.3 **2**

3.1 10.3.2 2

3.2 10.3.3 2

3.3 10.3.4 2 3

3.4 10.3.4 3 3

3.5 10.3.5 4

3.6 10.3.5 4

1 Allgemein

Das folgende Versuchsprotokoll bezieht sich auf den 10. Termin des Praktikums „Elektotechnische Grundlagen der Informatik“ im Somersemester 2007. Durchgeführt wurden die Versuche am Freitag, den 29. Juni 2007.

2 10.2

2.1 10.2.1 1

Aufgabenstellung: Modifizieren Sie das Beispielprogramm so, dass ein Ringzähler entsteht, der in der Schleife temp von 0xFF auf 0x00 herunterzählt und danach wieder bei 0xFF beginnt (Hinweis:Wenn ein Register den Wert 0x00 besitzt und um eins dekrementiert wird, nimmt es den Wert 0xFF an).

```
1 ; Only relevant parts
2 mainloop:
3         DEC temp          ; decrement temp
4         OUT PORTA,temp    ; output temp at port A
5         RJMP mainloop     ; infinite loop
```

3 10.3

3.1 10.3.2

Der Mikrocontroller wurde angeschlossen, das Programm übertragen. Der vorbereitete Code funktionierte wie erwartet.

3.2 10.3.3

Aufgabenstellung: Modifizieren Sie nun das Programm so, dass der Ringzähler nur im Intervall 0xFF bis 0xF0 - [Ihre Gruppennummer] durchläuft.
Wir errechneten 0xEE als untere Grenze und benutzten folgenden Code.

```
1 mainloop:
2 ; homework:
3         DEC temp          ; decrement temp
4 ; limitation:
5         CPI temp, 0xEE    ; compare temp to 0xEE
6         BRSH outp         ; jump to output if cmp >= 1
```

```
7           SER temp            ; reset temp to 0xFF
8 ; output and infinite loop:
9 outp:
10          OUT PORTA, temp     ; output temp at port A
11          RJMP mainloop       ; infinite loop
```

Das Ergebnis fiel wie erwartet aus.

3.3 10.3.4 2

Aufgabenstellung: Bisher konnten Sie das Zählen nur mit Hilfe des Debugmodus und eines Haltepunktes beobachten. Ergänzen Sie Ihr Programm nun um eine Abfrage, die das Drücken eines Tasters als Bedingung verwendet, um den Zähler zu dekrementieren. Verbinden Sie dazu einen der Taster mit dem Port B und konfigurieren Sie diesen als Eingang.
Nach verbinden des Tasters mit dem Port veränderten wir den Quellcode wie folgt.

```
1 .DEF inp = r17   ; define inp as universal register
2           SER inpu            ; temp = 0xFF
3           OUT DDRB, inp       ;
4 mainloop:
5 ; read in port and decrement temp if wanted
6           IN inp, PINB        ; read pinb into temp
7           CPI inp, 0xFF       ; compare input to 0xFF (btn not pressed)
8           BRLO outc           ; jump if inp is lt 0xFF (btn pressed)
9           DEC temp            ; decrement temp
10 ; write temp to porta and rejump to beginning
11 outc:
12          OUT PORTA, temp     ; output temp at port A
13          RJMP mainloop       ; infinite loop
```

3.4 10.3.4 3

Aufgabenstellung: Testen Sie den Mikrocontroller. Warum zeigt er nicht das gewünschte Verhalten?
Im Einzelschrittmodus war der Fehler nicht unmittelbar zu erkennen, aber schnell erkannten wir, dass die Bearbeitung in Echtzeit natürlich weitaus schneller stattfindet und somit ein einzelner Tastendruck zu unkontrollierbar vielen Dekrementierungen führte.

3.5 10.3.5

Aufgabenstellung: Ändern Sie Ihr Programm so, dass der Zähler bei jedem Tastendruck genau um eins dekrementiert wird.
Zum kontrollieren der Dekrementierungen musste eine "Warteschleife„erstellt werden.

```
1 mainloop:
2 ; read in port and decrement temp if wanted
3         IN inp,PINB      ; read pinb into temp
4         CPI inp,0xFF     ; compare input to 0xFF (btn not pressed)
5         BRE mainloop     ; jump if inp is 0xFF (btn not pressed)
6         DEC temp         ; decrement temp
7         OUT PORTA,temp   ; output temp at port A
8 ; wait until button is released again
9 wait:
10        IN inp,PINB      ; read pinb into temp
11        CPI inp,0xFF     ; compare input to 0xFF (btn not pressed)
12        BRLO wait        ; jump if inp is lt 0xFF (btn pressed)
13        RJMP mainloop    ; infinite loop
```

3.6 10.3.5

Aufgabenstellung: Interrupts, z.B. IRQ0. (Hinweis: Sie müssen in der Interruptroutine nicht das loslassen der Taste in einer Schleife abwarten.)
Es wurde folgender Quellcode erstellt und erfolgreich getestet.

```
1 ;-----------------------------------------------------------------
2 ; 1.: include definition
3 ;-----------------------------------------------------------------
4 .NOLIST ; pause list output
5 .INCLUDE <m16def.inc>    ; load definition for ATMEGA16
6 .LIST    ; resume list output
7 .CSEG    ; mark begin of code segment
8
9 ;-----------------------------------------------------------------
10 ; 2.: start address 0000
11 ;-----------------------------------------------------------------
12 reset:
13        JMP     init     ; Reset Handler
14
15 ;-----------------------------------------------------------------
16 ; 3.: interrupt service routine vectors
```

```
17 ;-----------------------------------------------------------------------------
18         JMP handlebtn    ; IRQ0 Handler
19         RETI     ; IRQ1 Handler
20         RETI     ; timer2 Compare Handler
21         RETI     ; timer2 Overflow Handler
22         RETI     ; timer1 Capture Handler
23         RETI     ; timer1 CompareA Handler
24         RETI     ; timer1 CompareB Handler
25         RETI     ; timer1 Overflow Handler
26         RETI     ; timer0 Overflow Handler
27         RETI     ; SPI Transfer Complete Handler
28         RETI     ; USART RX Complete Handler
29         RETI     ; UDR Empty Handler
30         RETI     ; USART TX Complete Handler
31         RETI     ; ADC Conversion Complete Handler
32         RETI     ; EEPROM Ready Handler
33         RETI     ; analog Comparator Handler
34         RETI     ; two-wire Serial Interface Handler
35         RETI     ; IRQ2 Handler
36         RETI     ; timer0 Compare Handler
37         RETI     ; store Program Memory Ready Handler
38
39 ;-----------------------------------------------------------------------------
40 ; 4.: initialize stack
41 ;-----------------------------------------------------------------------------
42 init:
43         LDI      r24,high(RAMEND)         ; set stack pointer
44         OUT      SPH,r24 ; "RAMEND" according to m16def.inc
45         LDI      r24,low(RAMEND) ;
46         OUT      SPL,r24 ;
47
48 ;-----------------------------------------------------------------------------
49 ; 5.: custom initialization
50 ;-----------------------------------------------------------------------------
51 ; insert custom initialization code here
52 .DEF count = r16          ; define count as universal register
53         SER count         ; count = FF
54         OUT DDRA,count    ; set data transfer for port A to output
55 .DEF temp = r17 ; define temp as universal register
56         CLR temp ; temp = 0x00
57         OUT DDRD, temp ; set port D to input (because temp == 0x00)
58         SER temp ; temp = 0xFF
59         OUT PORTD, temp ; use pullups
```

```
60          LDI temp, 0b00001010 ; configure INT0 and INT1
61          OUT MCUCR, temp ;
62          LDI temp, 0b11000000 ; activate INT0 and INT1
63          OUT GICR, temp ;
64          SEI ; activate interrupts globaly
65
66 ;-----------------------------------------------------------------------
67 ; 6.: main loop
68 ;-----------------------------------------------------------------------
69 mainloop:
70          OUT PORTA, count ; output count at port A
71          RJMP mainloop    ; infinite loop
72
73 handlebtn:
74          DEC count        ; decrement count
75          RETI     ; return
76
77 ;-----------------------------------------------------------------------
78 ; 7.: end
79 ;-----------------------------------------------------------------------
80 end:
81          RJMP end         ; infinite loop
```