Lösungen

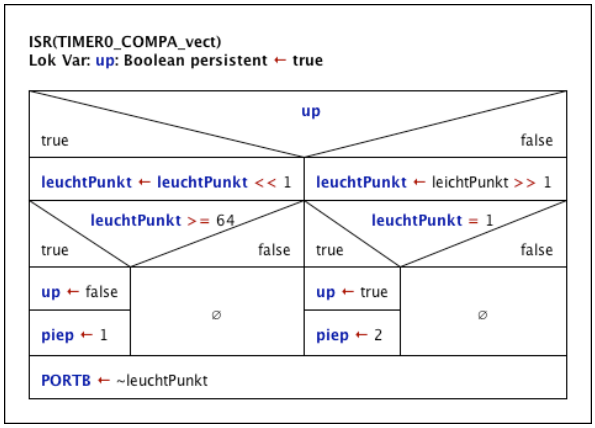
1a)

```
warte_ms:
  ldi aussen,70
loop_aussen:
  ldi innen,249       ;initialisieren
loop_innen:
  dec innen
  nop
  brne loop_innen
  nop
  dec aussen
  brne loop_aussen
ret
```

b) Beispiel1 n=1: 1 modulo 7 ist 1; 1/7 ist 0; Zugriff auf tone[1] und direkte Ausgabe in OCR1A
Beispiel2 n=8: 8 modulo 7 ist 1; 8/7 ist 1; Zugriff auf tone[1] und Halbierung der Periodendauer mit >>1. Durch Modulo 7 wird der Grundton ermittelt und durch /7 in die jeweilige Oktave durch Frequenzverdopplung transponiert.

c) ton(0) hat 264 Hz; ton(7) hat 528 Hz

d) Assemblerquellcode




```
1  .def null = R2
2  .def n    = R20
3  .def tmp  = R16
4  .def tmpH = R17
5  ton:
6    mov tmp,n
7    andi tmp,7 ;%8
8    ldi ZH,high(tone*2) ; lade Tabellenanfang tone[]
9    ldi ZL,low(tone*2)
10   lsl tmp             ; tmp * 2 wegen Word-Array
11   add ZL,tmp          ; addiere ton
12   adc ZH,null         ; addiere Uebertrag
13   lpm tmp,Z+          ; lade low von tone[ton]
14   lpm tmpH,Z          ; lade high
15   lsr n               ; /2
16   lsr n               ; /2
17   lsr n               ; /2
18 oktave:
19   breq tonaus
20   lsr tmpH            ; /2 durch rechtsschieben Bit 0 in Carry
21   ror tmp             ; teile durch 2 aber schiebe Carry oben rein
22   dec n               ; oktave erreicht
23   rjmp oktave
24 tonaus:
25   out OCR1AH, tmpH    ; speichere zuerst High
26   out OCR1AL, tmp     ; speichere dann Low
27   lsr tmpH            ; /2 durch rechtsschieben Bit 0 in Carry
28   ror tmp             ; teile durch 2 aber schiebe Carry oben rein
29   out OCR1BH, tmpH    ; speichere zuerst High
30   out OCR1BL, tmp     ; speichere dann Low
31   sbi DDRB,PB4        ; ton ausgeben
32   rcall warte_ms;
33   cbi DDRB,PB4
34   ret
35 tone: .dw F_CPU/264,F_CPU/297,F_CPU/330,F_CPU/352
36       .dw F_CPU/396,F_CPU/440,F_CPU/495,F_CPU/528;
```

2a) beide Variablen werden in der ISR verändert und der Compiler optimiert sie nun nicht weg.

b) Timer0 wird mit 1024 μs Takt betrieben. Bei Level0 ergibt dies 1024μs*201 = **205,8 ms**. Bei Level10 1024μs*101= **103,4 ms**.

c) Um sicher zu gewährleisten, dass die ISR immer nach der eingestellten Zeit auch gleichmäßig aufgerufen werden kann. Durch einen direkten Aufruf von *ton(..)* wäre sie für 70ms beschäftigt, bei höheren Level könnte dies zum Blockieren von Aufrufen führen. Die Variable *piep* dient zur Interprozesskommunikation zwischen ISR und Hauptprogramm, sie löst im HP den Aufruf von *ton(..)* aus.

d) Siehe oben

3a) Beim Einschalten Taste PD0 gedrückt halten.

b) Statt auf Taste PD3 zu drücken kann durch Druck auf Taste PD4 der Level unabhängig vom Treffen der mittleren LED erhöht werden.