Gewerbliche Schule Schwäbisch Hall

Klassenarbeit IT Übung Zeit 80 min TG12-1 Name _____

Einleitung

Ein Reaktionsspiel zum Beschäftigen von nervigen Kindern soll entwickelt werden. Ein Leuchtpunkt läuft immer zwischen PB0 und PB6 hin und her, an den Endpunkten wird ein Ton ausgegeben. Wird während die LED an PB3 leuchtet die Taste PD3 gedrückt erhöht sich der Level, dadurch erhöht sich die Tonhöhe der Endpunkttöne und die Geschwindigkeit des Leuchtpunktes. Das Spiel wird mit einem ATtiny2313 @1MHz und dem STK200 realisiert. Den Quellcode entnehmen Sie der Anlage.

1. Töne ausgeben — 20 Punkte

Unterprogramm (UP) ***ton(n:byte)*** Quelltextzeilen: 6..15; 55..56. Die Periodendauern der Töne sind im Programmspeicher abgelegt. Der Datentyp ist *unsigned int*, deshalb werden **2** Byte pro Feldelement verbraucht. Ihre Hauptaufgabe ist das Übersetzen in Assembler.

a) In Zeile 13 wird _delay_ms(..) aufgerufen, erstellen Sie den kommentierten Assemblerquelltext für *warte_ms*. 5P

b) In Zeile 10 wird dem Timer1 die Periodendauer für den Ton vorgegeben, erklären Sie den Ablauf der Befehlszeile, anhand zweier aussagekräftiger Beispiele. 3P

c) Welche Frequenzen werden bei ton(0) und ton(7) ausgegeben (Berechnung, Begründung) 2P

d) Assemblieren Sie das Unterprogramm *ton2(..)* unter Verwendung des UP-Aufrufs *warte_ms*: 10P

```
void ton2(unsigned char n){      // Ausgabe eines kurzen Tons auf PB4
  OCR1A = pgm_read_word(&tone[n%8])>>(n/8);// pro Oktave halbe Periodendauer
  OCR1B = OCR1A/2;
  DDRB |= 1<<PB4;    //Ton an
  _delay_ms(70);     //Tondauer
  DDRB &= ~(1<<PB4); //Ton aus
}
```

- ✳Code für Ablegen der Daten im Programmspeicher, verwenden Sie F_CPU und lassen den Assembler rechnen!
- ✳Code für UP ton: mit Parameter n in R20

2. ISR(TIMER0_COMPA_vect) — 10 Punkte

Zeile 32..50: In der ISR wird der Leuchtpunkt bewegt. Im Hauptprogramm in Zeile 74 bzw. 79 wird die Aufrufgeschwindigkeit dem Spiellevel angepasst.

a) Warum sind l*euchtPunkt* und *piep* ausserhalb der ISR als *volatile* definiert? 2P

b) Wie lang ist der Abstand der ISR-Aufrufe bei Level0 und Level10 in ms? 2P

c) Wieso wird *ton(..)* nicht in der ISR sondern im Hauptprogramm aufgerufen, welche Funktion hat dabei die Variable *piep*?. 2P

d) Erstellen Sie ein Struktogramm für die ISR 4P

3. Cheat-Modus — 4 Punkte

Zum Testen des Programms wurde ein Cheat-Modus eingebaut.

a) Wie aktiviert man den Cheat-Modus?. 2P

b) Wie kann das Spiel dann betrügerisch gespielt werden? 2P

Total Punkte 34

```
// *** Reaktions-Spiel V1.1 (c) Oliver Mezger 18.06.2018 ***
#include <avr/io.h>        // Definitionen laden
#include <avr/interrupt.h> // Interrupts laden
#include <util/delay.h>    // Delay-Bibliothek laden
#include <avr/pgmspace.h>  // Flashzugriffe laden
// C-Dur Frequenzen als Periodendauer F_CPU ist Systemtakt
const unsigned int PROGMEM tone[]={F_CPU/264,F_CPU/297,F_CPU/330,F_CPU/352,
                                   F_CPU/396,F_CPU/440,F_CPU/495};
void ton(unsigned char n){      // Ausgabe eines kurzen Tons auf PB4
  OCR1A = pgm_read_word(&tone[n%7])>>(n/7);// pro Oktave halbe Periodendauer
  OCR1B = OCR1A/2;
  DDRB |= 1<<PB4;     //Ton an
  _delay_ms(70);      //Tondauer
  DDRB &= ~(1<<PB4); //Ton aus
}
unsigned char keyOld = 0;        // alter Tasten-Zustand
unsigned char keyEnter,keyExit; // gedrueckte und losgelassene Tasten
void keyCheck(){           // Tastaturabfrage mit Flankendedektion
  unsigned char keyTest,tmp;
  keyEnter = 0, keyExit = 0;
  keyTest = ~PIND & 0b01111111; // Einlesen und zurechtschieben
  if (keyOld != keyTest){       // hat sich was getan
    _delay_ms(10);              // Prellen abwarten
    tmp = ~PIND & 0b01111111;   // nochmal Einlesen und zurechtschieben
    if (tmp == keyTest){        // ist es stabil?
      keyEnter = (~keyOld) & keyTest; // steigende Flanke !alt und neu
      keyExit = keyOld & (~keyTest);  // fallende Flanke alt und !neu
      keyOld = keyTest;
    }
  }
}
volatile unsigned char leuchtPunkt=1,piep=0;
ISR(TIMER0_COMPA_vect){       //bewegt den Leuchtpunkt
  static unsigned char up=1;
  if (up){
    leuchtPunkt = leuchtPunkt << 1;
    if (leuchtPunkt >= 64){ //wenn oben angekommen
      up=0;
      piep=1;               // einen tiefen Piep ausgeben
    }
  }
  else{
    leuchtPunkt = leuchtPunkt >> 1;
    if (leuchtPunkt == 1){  //wenn unten angekommen
      up =1;
      piep=2;               // einen hohen Piep ausgeben
    }
  }
  PORTB = ~leuchtPunkt;     //ausgeben Leuchtpunkt
}
int main(){
  unsigned char level=0,cheat=0;
  DDRB  = 0b01111111;  // PB0..6 als Ausgang
  PORTB = 0b01111111;  // alle LED aus
  TCCR1A= 0b00100011;  // Ausgang OC1B bei 0 setzen und bei OCR1B loeschen
  TCCR1B= 0b00011001;  // Waveform Generation Mode: Fast PWM it OCR1A als Top, Timer mit CPU-CLK
  TCCR0B= 5;           // Phi / 1024
  TCCR0A= 1<<WGM01;    // CTC Mode mit OCR0A als TOP
  TIMSK = 1<<OCIE0A;   // Interrupt frei geben
  OCR0A=200-level;     // bei (200-i)*1024µs Interrupt
  if ((PIND & 1) == 0) cheat=1;
  sei();
  while (1){           // Endlosschleife
    keyCheck();        // Tastaturabfrage
    if (keyEnter == 1<<PD3){ // wurde Taste gedrueckt?
      if (leuchtPunkt == 1<<PD3){
        ton(12);ton(24); // Ton fuer getroffen
        level++;         // naechster Level
      }
      else{
        ton(14);ton(8);ton(0);// Ton fuer daneben
        level=0;         // zurueck auf unteren Level
      }
      OCR0A=200-(level*10);  // Geschwindigkeit anpassen
    }
    if (cheat && keyEnter == 1<<PD4){
      ton(12);ton(24);
      level++;
      OCR0A=200-(level*10);
    }
    if (piep){  // wenn Eckpunkt erreicht Ton ausgeben
      ton(level+(piep-1)*8);
      piep=0;
    }
  }
}
```