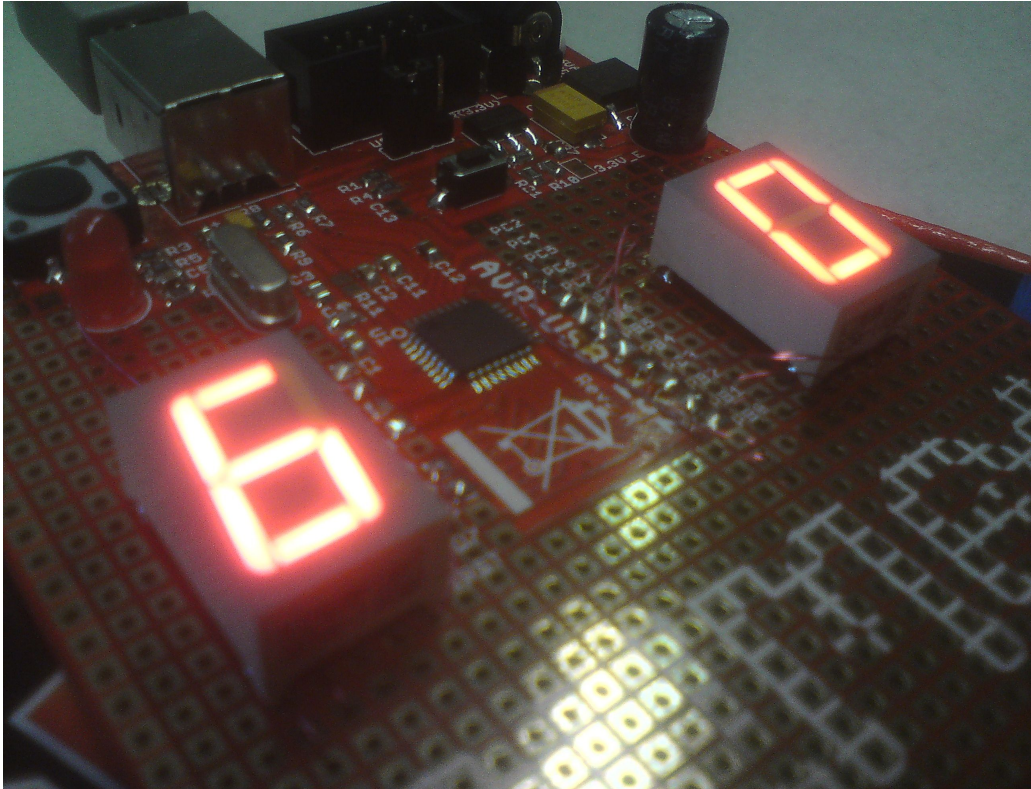


Up and Down - Projekt mit dem AT90USB162



Über diese Dokumentation:

Projekt geplant, durchgeführt und ausgearbeitet von:

Florian Patzer

Erweiterte Projektangaben:

Dieses Projekt wurde im Rahmen des Datenverarbeitungsunterrichts unter der Aufsicht von Herrn Gernoth durchgeführt. Aufgabenstellung war die Planung, Durchführung, sowie die Dokumentation eines Projekts mit dem hier vorgeführten Entwicklerboard.

Inhaltsverzeichnis

Über diese Dokumentation:.....	1
Projekt geplant, durchgeführt und ausgearbeitet von:.....	1
Erweiterte Projektangaben:.....	1
Hardware:.....	3
Hardware im Detail:	3
Projektbestandteile:.....	3
Projektziel:.....	3
Pinbelegung der 7-Segment-Anzeigen:.....	4
Links:.....	4
Rechts:.....	5
Quellcode:	5

Hardware:

- AT90USB162 auf Entwicklerboard AVR-USB162 von Olimex
- erweitert um zwei 7-Segment-Anzeigen

Hardware im Detail:

Der Mikrocontroller AT90USB162 von Atmel bietet 16kB Flash, 512B RAM und 512B EEPROM. Der Chip hat zusätzlich ein USB-Interface für die Programmierung und USART-Kommunikation integriert. Die Entwicklerplatine AVR-USB-162 von Olimex übernimmt die Spannungsversorgung des Controllers, zusätzlich sind ein Reset-Button, ein Taster, eine LED und ein externer Quarz mit 8MHz im Lieferzustand enthalten. Die I/O-Pins sind auf Lötunkten herausgeführt und die Platine hat einen Lochrasterbereich, so kann sie leicht mit Bauteilen erweitert werden. Für dieses Projekt wurden zwei 7-Segment-Anzeigen von Kingbright (Abb.1) per Fädeldraht auf Port B, Port C und Port D angelötet, die genaue Belegung der Ports zeigt folgende Tabelle.

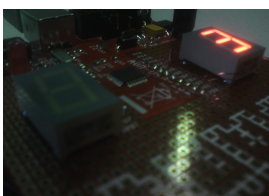
Projektbestandteile:

Dieses Projekt enthält folgende Elemente:

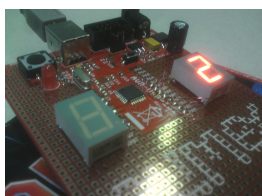
- Countdown
- Blinkende Anzeige „GO“
- Zählen nach oben (mit beiden Anzeigen + Sprung von 99 auf 0)
- Zählen nach unten (mit beiden Anzeigen + Sprung von 0 auf 99)
- Buttonabfrage mit Polling-Verfahren
- Richtungswechsel des Zählvorgangs
- Speichern der Wendepunkte
- Auswerten und Ausgeben der Wendepunkte

Projektziel:

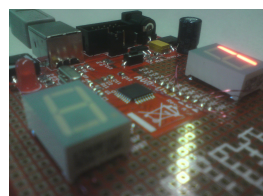
Das Programm soll mit einem Countdown starten, welcher mit der Ausgabe „GO“ abgeschlossen wird („GO“ wird durch „60“ dargestellt und blinkt).



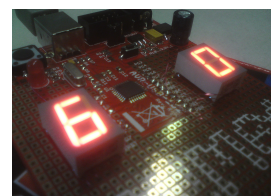
Rechte Anzeige: 3



Rechte Anzeige: 2



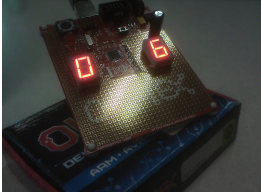
Rechte Anzeige: 1



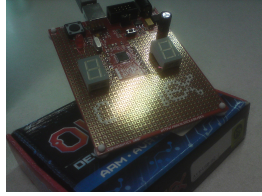
Anzeige: 6.....0

Daraufhin startet der Hochzählvorgang, der unendlich weiter läuft. Bei einem Überlauf von 99 auf 0 wird das Programm nicht unterbrochen. Wird nun der Taster gedrückt, ändert der Zählvorgang die Richtung und der Wert im Moment der Änderung wird gespeichert.

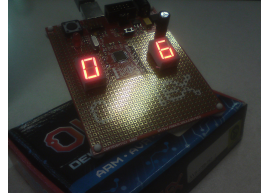
Bei einem Überlauf von 0 auf 99 während des Herunterzählens läuft das Programm ebenfalls weiter und fängt einfach wieder bei 99 an. Wenn die maximale Anzahl der möglichen zu speichernden Elemente erreicht ist, werden die Zahlen nacheinander ausgegeben. Die Ausgabe erfolgt automatisch. Jede Zahl ist eine Sekunde zu sehen und fängt dann das blinken an. Darauf folgt die nächste gespeicherte Zahl.



Anzeige: 0.....6

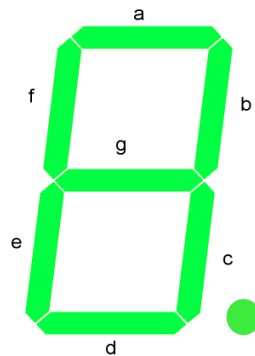


Anzeige: aus



Anzeige: 0.....6 → danach die nächste Zahl

Pinbelegung der 7-Segment-Anzeigen:



Links:

LEDs	Pins
a	D2
b	D3
c	D6
d	C7
e	C6
f	D1
g	D0
Punkt	D5

Rechts:

LEDs	Pins
a	B5
b	B4
c	B2
d	B1
e	B0
f	B6
g	B7
Punkt	B3

Quellcode:

Das Programm für den Mikrocontroller wurde mit AVR Studio in C realisiert und mit Atmel Flip per USB übertragen. Nachfolgend der Quellcode des Programms:

```
//Autor: Florian Patzer    Programm: Up and Down    21.07.2010

#include<avr/io.h>                //AVR input output Header
#define F_CPU 1000000            //Taktrate festlegen
#include<util/delay.h>           //Header für _delay_ms()
#define ANZAHL_STABIL 100       //Anzahl der Entprell-Abfragen, ->
                                //bis Pin als "stabil" eingestuft

#define STOREX_MAX 5            //Anzahl der zu speichernden

void entprellen(int pin){       //Funktion zum Schalterentprellen
    int i=0, current_state, last_state=0;

    while(i<ANZAHL_STABIL){     //Schleife bis ANZAHL_STABIL erreicht
        i++;
        current_state=(PIND&(1<<pin)); //entsprechenden Pin prüfen
        if(current_state!=last_state){ //Bei Springen des Schalters erneut
            i=0;
            last_state=current_state;
        }
    }

int check_button(char* direction){ //Funktion zum Button auswerten
    if(PIND&(1<<7)){             //Wenn nicht aktiv,=> verlassen
        return 0;
    }
    entprellen(7);
    if((PIND&(1<<7))==0){
        if(*direction=='u'){     //Richtungsvariable ändern
            *direction='d';
        }
        else{
            *direction='u';
        }
        entprellen(7);
        return 1;               //bei Änderung 1 zurückgeben
    }
    return 0;
}
```

```

void countdown_start(void) {                                     //Funktion stellt Countdown dar
    int i=0;
    PORTB=0;

    //3
    decide_i(3);                                               //setzt rechte Anzeige auf 3

    _delay_ms(1000);                                           //1sek warten
    PORTB=0;                                                  //Anzeige löschen

    //2
    decide_i(2);                                               //setzt rechte Anzeige auf 2

    _delay_ms(1000);                                           //usw.
    PORTB=0;

    //1
    decide_i(1);

    _delay_ms(1000);

    for(i=0;i<4;i++){                                         //GO (bzw. 60) blinkt 4 Mal
        //G
        decide_y(6)

        //0
        decide_i(0);

        _delay_ms(400);
        PORTD=0;                                               //beide Anzeigen löschen
        PORTC=0;                                               //
        PORTB=0;                                               //
        _delay_ms(400);
    }
}

void decide_y(int y) {                                         //Funktion belegt LEDs (linke Anzeige)
    PORTD=0;
    PORTC&=~(1<<PORTC6);                                       //PORTC6 ausschalten
    PORTC&=~(1<<PORTC7);                                       //PORTC7 ausschalten
    //Die einzelnen LEDs werden hier gesteuert und nur bei bestimmten Zahlen angeschalten->
    //so entsteht eine Zahl auf der Anzeige
    //a
    if(y!=1&&y!=4) {
        PORTD|=(1<<PORTD2);
    }
    //b
    if(y!=5&&y!=6) {
        PORTD|=(1<<PORTD3);
    }
    //c
    if(y!=2) {
        PORTD|=(1<<PORTD6);
    }
    //d
    if(y!=1&&y!=4&&y!=7) {
        PORTC|=(1<<PORTC7);
    }
    //e
    if(y==0||y==2||y==6||y==8) {
        PORTC|=(1<<PORTC6);
    }
    //f
    if(y!=1&&y!=2&&y!=3&&y!=7) {
        PORTD|=(1<<PORTD1);
    }
}

```

```

    //g
    if(y!=1&&y!=7&&y!=0) {
        PORTD|=(1<<PORTD0);
    }
}

void decide_i(int i){
    //Funktion belegt LEDs (rechte Anzeige)
    PORTB=0;
    //Die einzelnen LEDs werden hier gesteuert und nur bei bestimmten Zahlen angeschalten->
    //so entsteht eine Zahl auf der Anzeige
    //a
    if(i!=1&&i!=4) {
        PORTB|=(1<<PORTB5);
    }
    //b
    if(i!=5&&i!=6) {
        PORTB|=(1<<PORTB4);
    }
    //c
    if(i!=2) {
        PORTB|=(1<<PORTB2);
    }
    //d
    if(i!=1&&i!=4&&i!=7) {
        PORTB|=(1<<PORTB1);
    }
    //e
    if(i==0||i==2||i==6||i==8) {
        PORTB|=(1<<PORTB0);
    }
    //f
    if(i!=1&&i!=2&&i!=3&&i!=7) {
        PORTB|=(1<<PORTB6);
    }
    //g
    if(i!=1&&i!=7&&i!=0) {
        PORTB|=(1<<PORTB7);
    }
}

void end(int store[STOREX_MAX][2]){
    //Abschlussfunktion zum auswerten
    int element,times;
    while(1) {
        //Elemente werden durchgegangen und angezeigt
        for(element=0;element<STOREX_MAX;element++){
            decide_y(store[element][0]); //linke Anzeige
            decide_i(store[element][1]); //rechte Anzeige
            _delay_ms(1000); //1sek warten
            for(times=0;times<4;times++){ //Blinken beider Anzeigen
                decide_y(store[element][0]);
                decide_i(store[element][1]);
                _delay_ms(400);
                PORTB=0;
                PORTC=0;
                PORTD=0;
                _delay_ms(400);
            }
        }
    }
}

int main(void){
    int i=0,y=0,saved_i=0,saved_y=0;
    int changed=0, store[5][2], storex=0;
    char direction='u'; //Standartrichtung "up"
    DDRB=0xff; //B auf Ausgang setzen
    DDRD=0xff; //D auf Ausgang setzen
    DDRD&=~(1<<DDD7); //D7 auf Eingang setzen
    DDRC|=((1<<DDC6)|(1<<DDC7)); //c6 und c7 auf Eingang setzen
}

```

```

countdown_start();

while(1){
//Hoch und runter zählen:
  if(direction=='u'){ //Wenn Richtung Aufwärts
    _delay_ms(400);
    changed=0;
    for(y=saved_y;y<10;y++){ //0 (oder gespeicherter Wert) bis 9->
                                //(oder bis Änderung) linke Anzeige
      decide_y(y); //y ausgeben
      changed=check_button(&direction); //Button prüfen
      if(changed){
        if(storex<STOREX_MAX){ //wenn noch gespeichert werden darf:->
          store[storex][0]=y; //Wert von Anzeige(links) speichern
          store[storex][1]=saved_i; //Wert von Anzeige(rechts) speichern
          storex++; //Nächstes Element
        }
        else end(store); //Abbrechen und Ende-Funktion aufrufen
        break; //wenn geändert Schleife verlassen
      }
      for(i=saved_i;i<10;i++){ //0 (oder gespeicherter Wert) bis 9->
                                //(oder bis Änderung) rechte Anzeige
        saved_i=0; //gespeicherten Wert zurücksetzen->
                    //für nächsten Durchlauf
        decide_i(i); //i anzeigen

        _delay_ms(600);
        changed=check_button(&direction); //Button prüfen
        if(changed){ //s.o.
          if(storex<STOREX_MAX){
            store[storex][0]=y;
            store[storex][1]=i;
            storex++;
          }
          else end(store);
          break;
        }
      }

      if(changed){ //wenn Schleife wegen Veränderung->
                    //verlassen wurde
        saved_i=i; //i speichern
        saved_y=y; //y speichern
        break;
      }
    }

    if(changed==0) saved_y=0; //Schleife verlassen ohne Veränderung->
                                //gespeichertes y auf 0
  }

else{ //Wenn Richtung = abwärts

  _delay_ms(400);
  changed=0; //change zurücksetzen
  for(y=saved_y;y>=0;y--){ //9 (oder gespeicherter Wert) bis 0->
                                //(oder bis Änderung) linke Anzeige
    decide_y(y); //y anzeigen
    changed=check_button(&direction); //Button prüfen
    if(changed){ //s.o.
      if(storex<STOREX_MAX){
        store[storex][0]=y;
        store[storex][1]=saved_i;
        storex++;
      }
      else end(store);
      break;
    }
  }
}
}

```

```

for(i=saved_i;i>=0;i--){          //9 (oder gespeicherter Wert) bis 0->
                                //(oder bis Änderung) rechte Anzeige
    saved_i=9;                  //gespeicherten Wert auf 9 falls keine->
                                //Änderung erfolgt
    decide_i(i);                //i ausgeben
    _delay_ms(600);
    changed=check_button(&direction); //Button prüfen

    if(changed){                //s.o.
        if(storex<STOREX_MAX){
            store[storex][0]=y;
            store[storex][1]=i;
            storex++;
        }
        else end(store);
        break;
    }
    if(changed){                //wenn Änderung, Werte speichern
        saved_i=i;
        saved_y=y;
        break;
    }
    if(changed==0){            //wenn Durchlauf ohne Änderung->
        saved_y=9;              //y zurücksetzen
    }
}
}
return 0;
}

```

Copyright (c) 2010 Florian Patzer

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so,

subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Copyright © 2010 Florian Patzer

Hiermit wird unentgeltlich, jeder Person, die eine Kopie der Software und der zugehörigen Dokumentationen (die "Software") erhält, die Erlaubnis erteilt, uneingeschränkt zu benutzen, inklusive und ohne Ausnahme, dem Recht, sie zu verwenden, kopieren, ändern, fusionieren, verlegen, verbreiten, unterlizenzieren und/oder zu verkaufen, und Personen, die diese Software erhalten, diese Rechte zu geben, unter den folgenden Bedingungen:

Der obige Urheberrechtsvermerk und dieser Erlaubnisvermerk sind in alle Kopien oder Teilkopien der Software beizulegen.

DIE SOFTWARE WIRD OHNE JEDE AUSDRÜCKLICHE ODER IMPLIZIERTE GARANTIE BEREITGESTELLT, EINSCHLIESSLICH DER GARANTIE ZUR BENUTZUNG FÜR DEN VORGESEHENEN ODER EINEM BESTIMMTEN ZWECK SOWIE JEDLICHER RECHTSVERLETZUNG, JEDOCH NICHT DARAUf BESCHRÄNKT. IN KEINEM FALL SIND DIE AUTOREN ODER COPYRIGHTINHABER FÜR JEDLICHEN SCHADEN ODER SONSTIGE ANSPRÜCHE HAFTBAR ZU MACHEN, OB INFOLGE DER ERFÜLLUNG EINES VERTRAGES, EINES DELIKTES ODER ANDERS IM ZUSAMMENHANG MIT DER SOFTWARE ODER SONSTIGER VERWENDUNG DER SOFTWARE ENTSTANDEN.

Weitere Projekte, Tutorials und Ausarbeitungen des Autors finden sie unter www.florianpatzer.de .