

Addieren und Subtrahieren mit Mikrocontroller Atmega AT90162USB

Projekt:	Markus Sellner
Hardware:	AT90USB162 auf Entwicklerplatine AVR-USB-162 von Olimex erweitert um zwei 7-Segment-Anzeigen

Aufgabe:

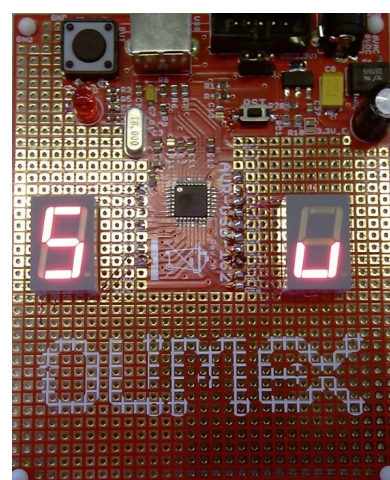
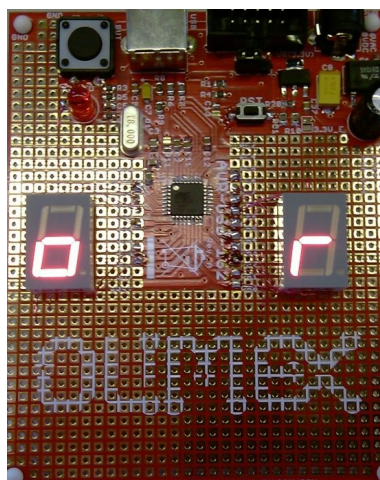
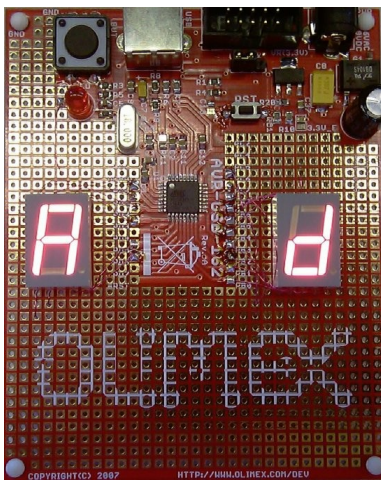
Mit dem Olimex-Testboard AVR-USB-162 wurde hier ein Addierer und Subtrahierer realisiert, der mit zwei 7-Segmentanzeigen und einem Taster bestückt ist. Addiert und Subtrahiert werden je 2 Zahlen mittels Tastendruck.

Realisierung allgemein:

Damit man mit nur einem Taster eine Zahl auswählen kann, wurde ein Zähler eingesetzt der bei Tastendruck anhält und die Zahl in einer Variablen für die 1. Zahl speichert. Ein weiterer Zähler läuft dann für die 2. Zahl die ebenfalls in einer Variablen gespeichert wird. Das Ergebnis wird dann berechnet und danach ausgegeben.

Zur Unterscheidung zwischen Addition und Subtraktion wird am Anfang ein Menü angezeigt, das ebenfalls via Tastendruck ausgewählt werden kann.

Menüpunkt 1 ist die Addition mit der Anzeige „Ad“
Menüpunkt 2 ist die Subtraktion mit der Anzeige „Su“



Realisierung Addition:

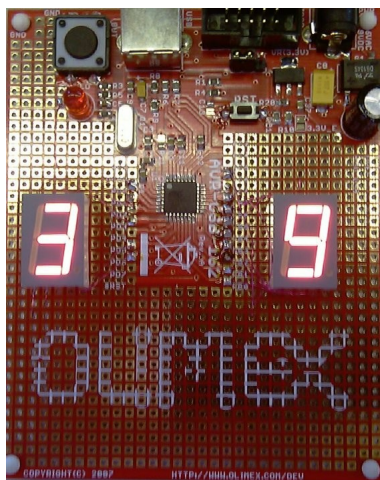
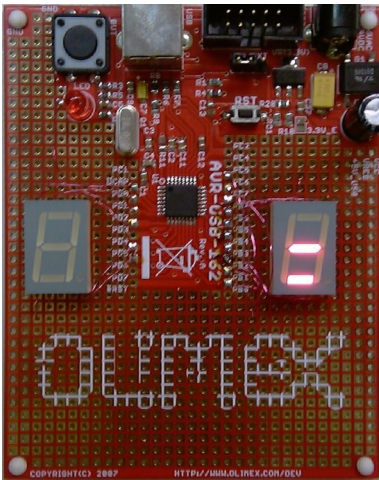
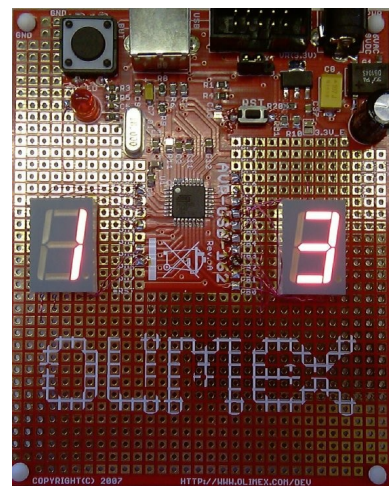
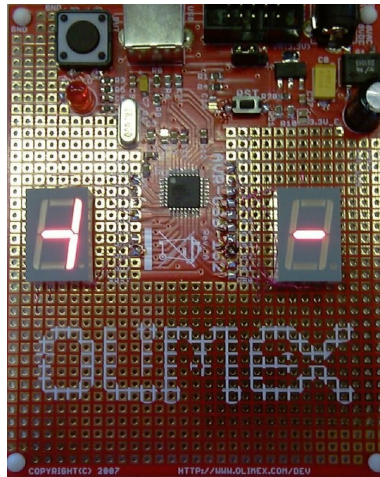
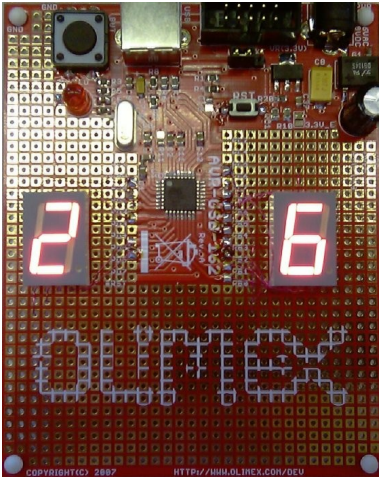
Bei der Addition läuft ein Zähler von 0 – 49. Sobald man den Taster drückt, wird die angezeigte Zahl mit den Variablen „i“ (Zehnerstelle) und „j“ (Einerstelle) an die Funktion „berechnung“ übergeben.

Dort werden Einer- und Zehnerstelle zusammengeführt und wieder der Funktion „addition“ zurückgegeben. Diese wird in der Variablen „zahl1“ gesichert.

Danach folgt ein weiterer Zähler von 0 – 49 (Dies ist erforderlich, damit bei der Addition nicht über die 99 hinaus gerechnet wird) mit gleicher Bearbeitung mit der Variablen „zahl2“.

Die Addition von „zahl1“ und „zahl2“ ergibt dann das Ergebnis (gesichert in „ergebnis“).

Um das Ergebnis wieder anzuzeigen, muss sie wieder in Einer- und Zehnerstelle zerlegt werden.



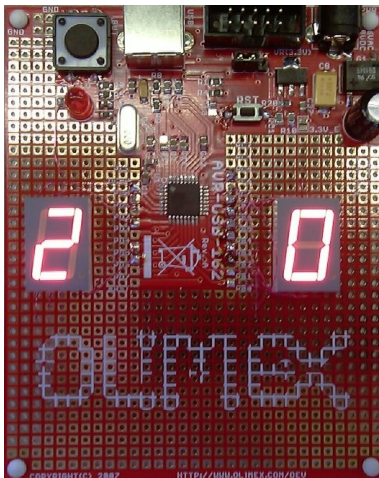
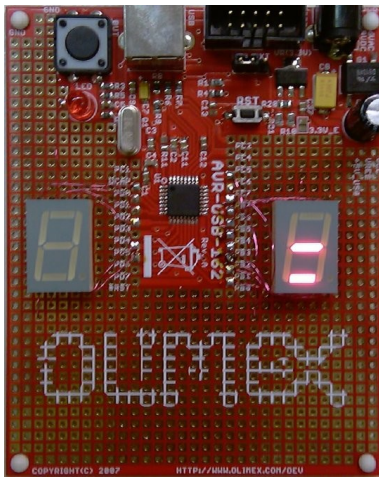
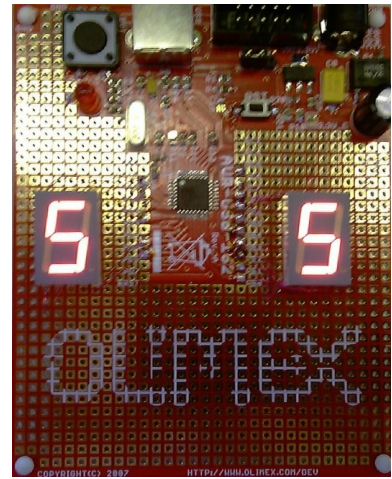
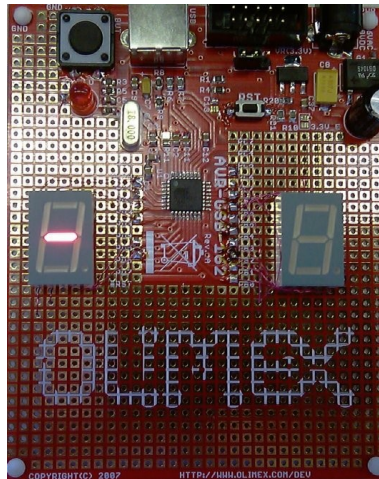
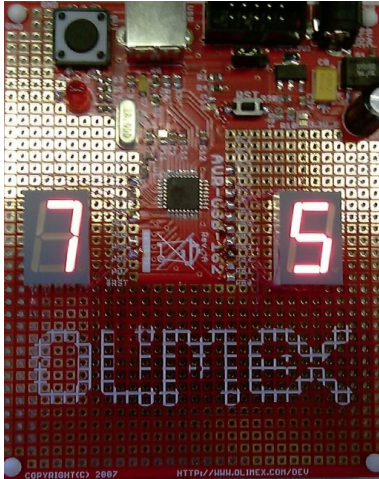
Realisierung Subtraktion:

Bei der Subtraktion läuft zuerst ein Zähler von $99 - 0$. Sobald man den Taster drückt, wird die angezeigte Zahl mit den Variablen „i“ (Zehnerstelle) und „j“ (Einerstelle) an die Funktion „berechnung“ übergeben. Dort werden Einer- und Zehnerstelle zusammengeführt und wieder der Funktion „subtraktion“ zurückgegeben. Diese wird in der Variablen „zahl1“ gesichert.

Danach folgt ein weiterer Zähler von $0 - 1$. Zahl (Dies ist erforderlich, damit bei der Subtraktion kein negativer Wert herauskommt) mit gleicher Bearbeitung mit der Variablen „zahl2“.

Die Subtraktion von „zahl1“ und „zahl2“ ergibt dann das Ergebnis (gesichert in „ergebnis“).

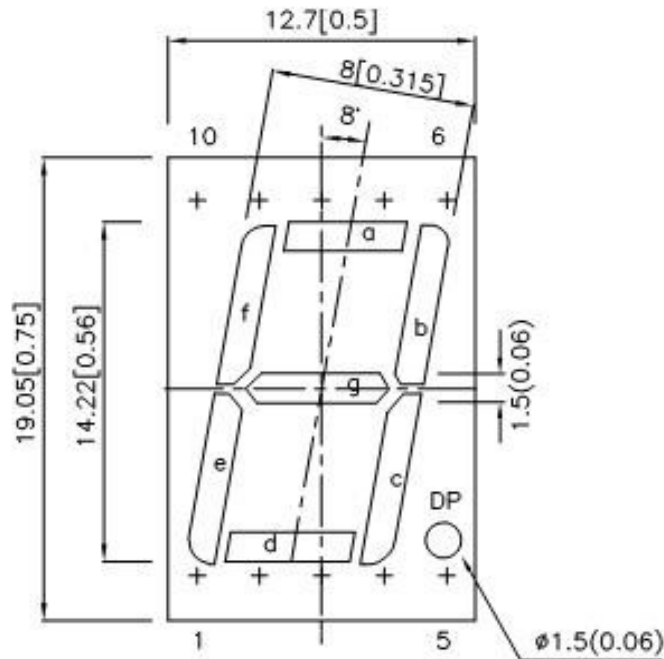
Um das Ergebnis wieder anzuzeigen, muss sie wieder in Einer- und Zehnerstelle zerlegt werden.



Hardware:

Der Mikrocontroller AT90USB162 von Atmel bietet 16kB Flash, 512B RAM und 512B EEPROM. Der Chip hat zusätzlich ein USB-Interface für 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-Segmentanzeigen von Kingbright (Abb.1) per Fädeldraht auf Port B, Port C und Port D angelötet, die genaue Belegung der Ports zeigt folgende Tabelle.

Die 7-Segmentanzeigen sind an Port B und Port C – D angeschlossen.



Die Belegung für PORT B (rechte 7-Segmentanzeige):

a = PORT B5 Hex 20
b = PORT B4 Hex 10
c = PORT B2 Hex 04
d = PORT B1 Hex 02
e = PORT B0 Hex 01
f = PORT B6 Hex 40
g = PORT B7 Hex 80

Die Belegung für PORT C – D (linke 7-Segmentanzeige):

a = PORT B5 Hex 20
b = PORT B4 Hex 10
c = PORT B2 Hex 04
d = PORT B1 Hex 02
e = PORT B0 Hex 01
f = PORT B6 Hex 40
g = PORT B7 Hex 80

Durch Addition der einzelnen Hex-Werte kann eine Zahl angezeigt werden.

Die Zahl 0 wird z. B. durch Hex 77 an Port B aufgerufen.

Die LED a,b,c,d,e,f wird mit Hex 20,10,04,02,01,40 zu Hex 77

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.

```
/*
Copyright (c) 2009 Markus Sellner
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 © 2009 Markus Sellner
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 bei zulegen.
DIE SOFTWARE WIRD OHNE JEDE AUSDRÜCKLICHE ODER IMPLIZIERTE GARANTIE
BEREITGESTELLT, EINSCHLIESSLICH DER GARANTIE ZUR BENUTZUNG FÜR DEN VORGEGEHEHENEN
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.
*/
```

Quellcode:

```
#include<avr\io.h>
#include<avr\wdt.h>

#ifndef F_CPU
#warning F_CPU wird nun mit 1000000 1MHz definiert
#define F_CPU 1000000
#endif
#include<util\delay.h>
#define ANZAHL_STABIL 1000 //Anzahl der Entprell-Abfragen, bis PIN als "stabil" eingestuft wird

void entprellen(int);
int check_button();
void addition();
void subtraktion();
int berechnung(int, int);

static unsigned int arr_B[10] = {0x77,0x14,0xB3,0xB6,0xD4,0xE6,0xE7,0x34,0xF7,0xF6}; //Arrays mit Zahlen 0 - 9
static unsigned int arr_C[10] = {0xC0,0x00,0xC0,0x80,0x00,0x80,0xC0,0x00,0xC0,0x80}; //Array für rechte Anzeige
static unsigned int arr_D[10] = {0x4E,0x48,0x0D,0x4D,0x4B,0x47,0x47,0x4C,0x4F,0x4F}; //Array für linke Anzeige

void entprellen(int pin)
{
    int i = 0;
```

```

int CURRENT_STATE, LAST_STATE = 0;
while(i < ANZAHL_STABIL)
{
    i++;

    CURRENT_STATE = (PIND & (1 << pin));
    if(CURRENT_STATE != LAST_STATE)
    {
        i = 0;
        LAST_STATE = CURRENT_STATE;
    }
}

int check_button()
{
    if(PIND & (1 << 7))
    {
        return 0;
    }
    entprellen(7);
    while((PIND & (1 << 7)) == 0);
    entprellen(7);
    return 1;
}

void addition()
{
    int i, j, push = 0;
    int zahl1 = 0, zahl2 = 0, ergebnis = 0;
    while(push != 1)
    {
        for(i=0; i<5; i++)
        {
            PORTC = arr_C[i];
            PORTD = arr_D[i];
            for(j=0; j<10; j++)
            {
                PORTB = arr_B[j];
                _delay_ms(400);
                push = check_button();
                if(push == 1)
                {
                    zahl1 = berechnung(i, j); //Berechnung aus Einer- und Zehner-Zahl
                    i = 10;
                    j = 10;
                    break;
                }
            }
        }
    }

    push = 0;
    PORTB = 0x00;
    PORTC = 0x00;
    PORTD = 0x00;
    _delay_ms(1500);
    PORTB = 0x80;
    PORTD = 0x49;
    _delay_ms(3500);
    PORTB = 0x00;
    PORTC = 0x00;
    PORTD = 0x00;
}

```

*//Wenn der Pin nicht im last_state ist,
//den Counter zurücksetzen und den
//last_state anpassen. Es wurde ein
//prellen erkannt.*

//Rückgabe 1 wenn Taster gedrückt.

// Ausführen, solange kein Tastendruck

//Zähler für die 1. Zahl von 0 - 49

//Zähler für die 1. Zahl von 0 - 49

*//Abfrage ob Taste gedrückt wurde
//Wenn Taste gedrückt wurde ->*

//Ende des Zählers für 1. Zahl

//Keine Ausgabe auf 7-Segmentanzeige

*//Ausgabe eines + Zeichens über
//PORTB
//PORTC - D*

//Keine Ausgabe auf 7-Segmentanzeige

```

while(push != 1) // Ausführen, solange kein Tastendruck
{
    for(i=0;i<5;i++) //Zähler für die 2. Zahl von 0 - 49
    {
        PORTC = arr_C[i];
        PORTD = arr_D[i];
        for(j=0;j<10;j++) //Zähler für die 2. Zahl von 0 - 49
        {
            PORTB = arr_B[j];
            _delay_ms(400);
            push = check_button(); //Abfrage ob Taste gedrückt wurde
            if(push == 1) //Wenn Taste gedrückt wurde ->
            {
                zahl2 = berechnung(i,j); //Berechnung aus Einer- und Zehner-Zahl
                i = 10;
                j = 10;
                break; //Ende des Zählers für 2. Zahl
            }
        }
    }
}

push = 0;
PORTB = 0x00; //Keine Ausgabe auf 7-Segmentanzeige
PORTC = 0x00;
PORTD = 0x00;
_delay_ms(1500); //Ausgabe eines = Zeichens über
PORTB = 0x82; //PORTB
_delay_ms(3500);
PORTB = 0x00; //Keine Ausgabe auf 7-Segmentanzeige
PORTC = 0x00;
PORTD = 0x00;

ergebnis = zahl1 + zahl2; //1. Zahl addiert mit 2. Zahl ergibt Ergebnis
i = ergebnis / 10; //Zehner-Zahl berechnen vom Ergebnis
j = ergebnis % 10; //Einer-zahl berechnen vom Ergebnis

PORTB = arr_B[j]; //Ausgabe Zehner-Zahl vom Ergebnis
PORTC = arr_C[i]; //Ausgabe Einer-Zahl vom Ergebnis
PORTD = arr_D[i]; //Ausgabe Einer-Zahl vom Ergebnis
_delay_ms(10000);
}

```

```

void subtraktion()
{
    int i,j,max_i = 0,max_j = 0,push = 0;
    int zahl1 = 0, zahl2 = 0, ergebnis = 0;
    while(push != 1) // Ausführen, solange kein Tastendruck
    {
        for(i=9;i>=0;i--) //Zähler für die 1. Zahl von 99 - 0
        {
            PORTC = arr_C[i];
            PORTD = arr_D[i];
            for(j=9;j>=0;j--)
            {
                PORTB = arr_B[j];
                _delay_ms(400);
                push = check_button(); //Abfrage ob Taste gedrückt wurde
                if(push == 1) //Wenn Taste gedrückt wurde ->
                {
                    zahl1 = berechnung(i,j); //Berechnung aus Einer- und Zehner-Zahl
                    max_i = i;
                    max_j = j;
                }
            }
        }
    }
}

```

```

        i = 0;
        j = 0;
        break;
    }
}

}

push = 0;
PORTB = 0x00;
PORTC = 0x00;
PORTD = 0x00;
_delay_ms(1500);
PORTB = 0x00;
PORTD = 0x01;
_delay_ms(3500);
PORTB = 0x00;
PORTC = 0x00;
PORTD = 0x00;

while(push != 1)
{
    for(i=0;i<=max_i;i++)
    {
        PORTC = arr_C[i];
        PORTD = arr_D[i];
        if(i != max_i)
        {
            for(j=0;j<10;j++)
            {
                PORTB = arr_B[j];
                _delay_ms(400);
                push = check_button();
                if(push == 1)
                {
                    zahl2 = berechnung(i,j); //Berechnung aus Einer- und Zehner-Zahl
                    i = 10;
                    j = 10;
                    break;
                }
            }
        }
        else
        {
            for(j=0;j<max_j;j++)
            {
                PORTB = arr_B[j];
                _delay_ms(400);
                push = check_button();
                if(push == 1)
                {
                    zahl2 = berechnung(i,j); //Berechnung aus Einer- und Zehner-Zahl
                    i = 10;
                    j = 10;
                    break;
                }
            }
        }
    }
}

push = 0;
PORTB = 0x00;
PORTC = 0x00;

```

//Ende des Zählers für 1. Zahl

//Keine Ausgabe auf 7-Segmentanzeige

//Ausgabe eines - Zeichens über
//PORTB
//PORTC - D

//Keine Ausgabe auf 7-Segmentanzeige

// Ausführen, solange kein Tastendruck

//Zähler für die 2. Zahl von 0 - 1. Zahl

//Abfrage ob Taste gedrückt wurde
//Wenn Taste gedrückt wurde ->

//Berechnung aus Einer- und Zehner-Zahl

//Ende des Zählers für 2. Zahl

//Abfrage ob Taste gedrückt wurde
//Wenn Taste gedrückt wurde ->

//Berechnung aus Einer- und Zehner-Zahl

//Ende des Zählers für 2. Zahl

//Keine Ausgabe auf 7-Segmentanzeige

```

PORTD = 0x00;
_delay_ms(1500);
PORTB = 0x82;
_delay_ms(3500);
PORTB = 0x00;
PORTC = 0x00;
PORTD = 0x00;

//Ausgabe eines = Zeichens über
//PORTB

//Keine Ausgabe auf 7-Segmentanzeige

ergebnis = zahl1 - zahl2;
i = ergebnis / 10;
j = ergebnis % 10;

//1. Zahl subtrahiert mit 2. Zahl ergibt Ergebnis
//Zehner-Zahl berechnen vom Ergebnis
//Einer-zahl berechnen vom Ergebnis

PORTB = arr_B[j];
PORTC = arr_C[i];
PORTD = arr_D[i];
_delay_ms(10000);

//Ausgabe Zehner-Zahl vom Ergebnis
//Ausgabe Einer-Zahl vom Ergebnis
//Ausgabe Einer-Zahl vom Ergebnis
}

int berechnung(int i, int j)
{
    int ergebnis;
    i = i * 10;
    j = j * 1;
    ergebnis = i + j;
    return ergebnis;
}

//Da die Einer- und Zehnerstelle der Zahl
// getrennt sind, werden sie hier zusammen-
// gefügt
//Einer- und Zehner-Zahl miteinander addieren
//Rückgabe Ergebnis an die jeweilige Funktion

int main(void)
{
    int push = 0;
    DDRB = 0xff;
    DDRC = 0xC0;
    DDRD = 0xCF;
    PORTB = 0x00;
    PORTC = 0x00;
    PORTD = 0x00;
    DDRD &= ~(1 << 7);
    _delay_ms(2500);

    //PORTB 0-7 als Ausgang
    //PORTC 6-7 als Ausgang
    //PORTD 0-3,6 als Ausgang
    //PORTB keine Ausgabe
    //PORTC keine Ausgabe
    //PORTD keine Ausgabe

    while(1)
    {
        PORTB = 0x97;
        PORTC = 0x40;
        PORTD = 0x4F;
        _delay_ms(2500);
        //Ausgabe von Menüpunkt 1 „Ad“
        //PORTB
        //PORTC
        //PORTD

        push = check_button();
        if(push == 1)
        {
            //Abfrage ob Taste gedrückt wurde
            //Wenn Taste gedrückt wurde ->

            addition();
            //Führe Additionsprogramm aus

            PORTB = 0x00;
            PORTC = 0x00;
            PORTD = 0x00;
            _delay_ms(1000);
            //Keine Ausgabe auf 7-Segmentanzeige

            PORTB = 0x81;
            PORTC = 0xC0;
            PORTD = 0x41;
            _delay_ms(2500);
            //Ausgabe von „,or“ für oder(english)
            //PORTB
            //PORTC
            //PORTD

            PORTB = 0x00;
            PORTC = 0x00;
            PORTD = 0x00;
            _delay_ms(1000);
            //Keine Ausgabe auf 7-Segmentanzeige

            PORTB = 0x07;
            PORTC = 0x80;
            PORTD = 0x47;
            //Ausgabe von Menüpunkt 1 „Su“
            //PORTB
            //PORTC
            //PORTD

```

```
    _delay_ms(2500);  
    push = check_button();  
    if(push == 1)  
        {  
            subtraktion();  
        }  
    PORTB = 0x00;  
    PORTC = 0x00;  
    PORTD = 0x00;  
}  
return 0;  
}
```

*//Abfrage ob Taste gedrückt wurde
//Wenn Taste gedrückt wurde ->

//Führe Subtraktionsprogramm aus

//Keine Ausgabe auf 7-Segmentanzeige*