```
/*
* Stiftuhr mit PWM
* Created: 16.08.2015 23:29:27
*  Author: Harri
*

a
I ----- I
I                   I
I f         I b
I                   I
I           g       I
-----
I                   I
I                   I
I e                 I c
I                   I
I ----- I
d
a = B4                          linke Anzeige = B0
b = B3                          rechte Anzeige = C0
c = C1                          LED grün = D4
d = C4                          LED gelb = D3
e = C3                          Servo rechts = D1                       0 = 8000, 90° = 14500
f = B1                          Servo links         = D2                0 = 14000, 90° = 7500
g = B2                          Servo Hub = D0                          schreiben =15000  , heben1 =13000   ,
vollhub =10000
dp = C2

Taste links = D5
Taste rechts = D7
Taste mitte = D6
*/

#define  F_CPU 8000000                                  // interner Oszilator 8MHz
#include <avr/io.h>
#include <avr/interrupt.h>                              // Interruptheader
#include "C:\Users\media\Desktop\Harri\Technik\Elektronik\Projekte\Stiftuhr\Servotest\servotest\servotest.h"

volatile uint16_t pulsweite2;
volatile uint16_t pulsweite3;
volatile uint16_t pulsweite1;                           // Arbeitsvariablen für Pulsweiten
volatile uint16_t periodencounter;
volatile uint16_t zeitcounter;
volatile uint8_t sekunde;
volatile uint8_t minute;
volatile uint8_t stunde;

//*****************************************************************************************************
************************************
//
        SINUSTABELLE
uint16_t winkelfunktion (uint16_t winkel)                                       // ermittelt den Winkel bei
gegebener Winkelfunktion
{

        uint8_t counter;
        counter = 0;
        uint16_t sintabelle[181] = {0,9,17,26,35,44,52,61,70,78,
                87,96,105,113,122,131,139,148,156,165,
                174,182,191,199,208,216,225,233,242,250,
                259,267,276,284,292,301,309,317,326,334,
                342,350,358,367,375,383,391,399,407,415,
```

```
                423,431,438,446,454,462,469,477,485,492,
                500,508,515,522,530,537,545,552,559,566,
                574,581,588,595,602,609,616,623,629,636,
                643,649,656,663,669,676,682,688,695,701,
                707,713,719,725,731,737,743,749,755,760,
                766,772,777,783,788,793,799,804,809,814,
                819,824,829,834,839,843,848,853,857,862,
                866,870,875,879,883,887,891,895,899,903,
                906,910,914,917,921,924,927,930,934,937,
                940,943,946,948,951,954,956,959,961,964,
                966,968,970,972,974,976,978,980,982,983,
                985,986,988,989,990,991,993,994,995,995,
                996,997,998,998,999,999,999,1000,1000,1000,
        1000,};

        for (counter = 0; counter <= 180; counter++)
        {
                if (sintabelle[counter] >= winkel)                                      // vergleicht den gegebnen
sinwert mit der Tabelle bis eine übereinstimmung erreicht ist und gibt dann den winkel zurück
                {
                        return counter;
                }
        }

        return counter;
}

//*****************************************************************************************************
**********************************************
//
                                Löschen
uint8_t loeschen (void)
{
        uint16_t counter;
        counter = 0;
        uint8_t douplecounter;
        douplecounter = 0;
        uint8_t punktcounter;
        punktcounter = 0;

        while(1)
        {                                                       // bewegt den stift zum löschtopf, senkt ab, wischt ab und bringt
den löschtopf zurück
                punktcounter++;
                switch(punktcounter)
                {
                        case(1):
                        pulsweite3 = 10000;
                        pulsweite1 = 9000;
                        pulsweite2 = 12000;
                        break;

                        case(2):
                        pulsweite3 = 10000;
                        pulsweite1 = 6500;
                        pulsweite2 = 11000;
                        break;
                        case(3):
                        pulsweite3 = 14800;
                        pulsweite1 = 6500;
                        pulsweite2 = 11000;
                        break;
                        case(4):
```

```
                        pulsweite3 = 14800;
                        pulsweite1 = 13000;
                        pulsweite2 = 13500;
                        break;
                        case(5):
                        pulsweite3 = 14800;
                        pulsweite1 = 10000;
                        pulsweite2 = 10300;
                        break;

                        case(6):
                        pulsweite3 = 14800;
                        pulsweite1 = 13000;
                        pulsweite2 = 13500;
                        break;
                        case(7):
                        pulsweite3 = 14800;
                        pulsweite1 = 10000;
                        pulsweite2 = 10300;
                        break;
                        case(8):
                        pulsweite3 = 14800;
                        pulsweite1 = 9000;
                        pulsweite2 = 12000;

                        break;

                        case(9):
                        pulsweite3 = 14800;
                        pulsweite1 = 6500;
                        pulsweite2 = 11000;

                        break;
                        case(10):
                        pulsweite3 = 10000;
                        pulsweite1 = 6500;
                        pulsweite2 = 11000;
                        break;

                        case(11):
                        return 1;
                        break;

                }

                for(counter = 0; counter < 10000; counter++)
                {
                        for (douplecounter = 0; douplecounter < 10; douplecounter++)
                        {
                                // warteschleife verzögert die wischbewegung
                        }
                }
        }

        return 1;
}
//*****************************************************************************************************
*****************************************************+
//
                Ziffern
uint8_t ziffern (uint8_t zahl, uint8_t offset)
{
        uint8_t ziffer[16] [2];
```

```
uint8_t counter;
counter = 0;
uint16_t position;
position = 0;

for(counter = 0; counter <= 15; counter++)
{
	ziffer[counter][0] = 1;			// alle possitionen im Array auf null gesetzt
	ziffer[counter][1] = 1;
}

switch(zahl)
{
	case(1):
	ziffer[0][0] = 2+offset;
	ziffer[0][1] = 64;
	ziffer[1][0] = 0;
	ziffer[1][1] = 0;
	ziffer[2][0] = 10+offset;
	ziffer[2][1] = 70;
	ziffer[3][0] = 10+offset;
	ziffer[3][1] = 51;

	break;

	case(2):
	ziffer[0][0] = 2+offset;
	ziffer[0][1] = 66;
	ziffer[1][0] = 0;
	ziffer[1][1] = 0;
	ziffer[2][0] = 4+offset;
	ziffer[2][1] = 70;
	ziffer[3][0] = 8+offset;
	ziffer[3][1] = 71;
	ziffer[4][0] = 11+offset;
	ziffer[4][1] = 68;
	ziffer[5][0] = 2+offset;
	ziffer[5][1] = 51;
	ziffer[6][0] = 12+offset;
	ziffer[6][1] = 51;
	break;

	case(3):
	ziffer[0][0] = 2+offset;
	ziffer[0][1] = 66;
	ziffer[1][0] = 0;
	ziffer[1][1] = 0;
	ziffer[2][0] = 4+offset;
	ziffer[2][1] = 70;
	ziffer[3][0] = 8+offset;
	ziffer[3][1] = 71;
	ziffer[4][0] = 11+offset;
	ziffer[4][1] = 68;
	ziffer[5][0] = 6+offset;
	ziffer[5][1] = 61;
	ziffer[6][0] = 11+offset;
	ziffer[6][1] = 59;
	ziffer[7][0] = 11+offset;
	ziffer[7][1] = 56;
	ziffer[8][0] = 6+offset;
	ziffer[8][1] = 54;
	ziffer[9][0] = 3+offset;
	ziffer[9][1] = 56;
```

```
break;

case(4):
ziffer[0][0] = 2+offset;
ziffer[0][1] = 70;
ziffer[1][0] = 0;
ziffer[1][1] = 0;
ziffer[2][0] = 2+offset;
ziffer[2][1] = 56;
ziffer[3][0] = 12+offset;
ziffer[3][1] = 56;
ziffer[4][0] = 1;
ziffer[4][1] = 1;
ziffer[5][0] = 8+offset;
ziffer[5][1] = 63;
ziffer[6][0] = 0;
ziffer[6][1] = 0;
ziffer[7][0] = 8+offset;
ziffer[7][1] = 50;

break;

case(5):
ziffer[0][0] = 12+offset;
ziffer[0][1] = 69;
ziffer[1][0] = 0;
ziffer[1][1] = 0;
ziffer[2][0] = 2+offset;
ziffer[2][1] = 69;
ziffer[3][0] = 2+offset;
ziffer[3][1] = 62;
ziffer[4][0] = 8+offset;
ziffer[4][1] = 62;
ziffer[5][0] = 11+offset;
ziffer[5][1] = 59;
ziffer[6][0] = 11+offset;
ziffer[6][1] = 56;
ziffer[7][0] = 7+offset;
ziffer[7][1] = 54;
ziffer[8][0] = 2+offset;
ziffer[8][1] = 56;

break;

case(6):
ziffer[0][0] = 12+offset;
ziffer[0][1] = 69;
ziffer[1][0] = 0;
ziffer[1][1] = 0;
ziffer[2][0] = 5+offset;
ziffer[2][1] = 65;
ziffer[3][0] = 2+offset;
ziffer[3][1] = 61;
ziffer[4][0] = 4+offset;
ziffer[4][1] = 55;
ziffer[5][0] = 7+offset;
ziffer[5][1] = 54;
ziffer[6][0] = 11+offset;
ziffer[6][1] = 56;
ziffer[7][0] = 11+offset;
ziffer[7][1] = 59;
ziffer[8][0] = 7+offset;
```

```
ziffer[8][1] = 61;
ziffer[9][0] = 4+offset;
ziffer[9][1] = 63;
break;

case(7):
ziffer[0][0] = 2+offset;
ziffer[0][1] = 69;
ziffer[1][0] = 0;
ziffer[1][1] = 0;
ziffer[2][0] = 12+offset;
ziffer[2][1] = 69;
ziffer[3][0] = 8+offset;
ziffer[3][1] = 63;
ziffer[4][0] = 6+offset;
ziffer[4][1] = 52;
ziffer[5][0] = 1;
ziffer[5][1] = 1;
ziffer[6][0] = 5+offset;
ziffer[6][1] = 60;
ziffer[7][0] = 0;
ziffer[7][1] = 0;
ziffer[8][0] = 11+offset;
ziffer[8][1] = 60;

break;

case(8):
ziffer[0][0] = 7+offset;
ziffer[0][1] = 63;
ziffer[1][0] = 0;
ziffer[1][1] = 0;
ziffer[2][0] = 3+offset;
ziffer[2][1] = 65;
ziffer[3][0] = 2+offset;
ziffer[3][1] = 69;
ziffer[4][0] = 5+offset;
ziffer[4][1] = 70;
ziffer[5][0] = 9+offset;
ziffer[5][1] = 70;
ziffer[6][0] = 12+offset;
ziffer[6][1] = 65;
ziffer[7][0] = 7+offset;
ziffer[7][1] = 63;
ziffer[8][0] = 3+offset;
ziffer[8][1] = 60;
ziffer[9][0] = 2+offset;
ziffer[9][1] = 56;
ziffer[10][0] = 5+offset;
ziffer[10][1] = 54;
ziffer[11][0] = 7+offset;
ziffer[11][1] = 53;
ziffer[12][0] = 12+offset;
ziffer[12][1] = 56;
ziffer[13][0] = 12+offset;
ziffer[13][1] = 59;
ziffer[14][0] = 7+offset;
ziffer[14][1] = 63;
break;

case(9):
ziffer[0][0] = 12+offset;
ziffer[0][1] = 67;
```

```
ziffer[1][0] = 0;
ziffer[1][1] = 0;
ziffer[2][0] = 10+offset;
ziffer[2][1] = 64;
ziffer[3][0] = 7+offset;
ziffer[3][1] = 63;
ziffer[4][0] = 4+offset;
ziffer[4][1] = 69;
ziffer[5][0] = 2+offset;
ziffer[5][1] = 67;
ziffer[6][0] = 2+offset;
ziffer[6][1] = 69;
ziffer[7][0] = 3+offset;
ziffer[7][1] = 70;
ziffer[8][0] = 9+offset;
ziffer[8][1] = 70;
ziffer[9][0] = 12+offset;
ziffer[9][1] = 68;
ziffer[10][0] = 12+offset;
ziffer[10][1] = 67;
ziffer[11][0] = 12+offset;
ziffer[11][1] = 58;
ziffer[12][0] = 10+offset;
ziffer[12][1] = 55;
ziffer[13][0] = 7+offset;
ziffer[13][1] = 53;

break;

case(0):
ziffer[0][0] = 7+offset;
ziffer[0][1] = 69;
ziffer[1][0] = 0;
ziffer[1][1] = 0;
ziffer[2][0] = 4+offset;
ziffer[2][1] = 67;
ziffer[3][0] = 2+offset;
ziffer[3][1] = 65;
ziffer[4][0] = 2+offset;
ziffer[4][1] = 57;
ziffer[5][0] = 4+offset;
ziffer[5][1] = 56;
ziffer[6][0] = 7+offset;
ziffer[6][1] = 53;
ziffer[7][0] = 10+offset;
ziffer[7][1] = 56;
ziffer[8][0] = 12+offset;
ziffer[8][1] = 59;
ziffer[9][0] = 12+offset;
ziffer[9][1] = 65;
ziffer[10][0] = 10+offset;
ziffer[10][1] = 67;
ziffer[11][0] = 7+offset;
ziffer[11][1] = 69;

break;
case(99):
ziffer[0][0] = 56;                              // zeichnet den doppelpunkt
ziffer[0][1] = 66;
ziffer[1][0] = 0;
ziffer[1][1] = 0;
ziffer[2][0] = 56;
ziffer[2][1] = 62;
```

```
                ziffer[3][0] = 1;
                ziffer[3][1] = 1;
                ziffer[4][0] = 56;
                ziffer[4][1] = 56;
                ziffer[5][0] = 0;
                ziffer[5][1] = 0;
                ziffer[6][0] = 56;
                ziffer[6][1] = 52;
                break;

        }

        for (counter = 0; counter <= 15; counter++)                     //übergibt das x/y wertepaar an
die umsetzungsfunktion
        {
                position = (ziffer[counter][0]) * 100 + ziffer[counter][1];
                stellwinkel(position);
        }

        return 1;

}

//*****************************************************************************************************
**************************************
//                                                                                              Wurzelziehen
uint16_t wurzel (uint16_t zahl)
{
        uint32_t vergleich;
        vergleich = 0;

        for(vergleich = zahl/100; vergleich*vergleich <= zahl; vergleich++)
        {
                // ermittlung der Quadratwurzel (näherungsweise)
        }

        return vergleich;
}

//*****************************************************************************************************
**************************************
//                                                                              Uhrzeit in Einzelzahlen
zerlegen
uint8_t zeitaufteiler (void)
{
        uint8_t zeitzahl;
        zeitzahl = 0;
        uint8_t counter;
        counter = 1;

        while(counter)
        {
                if (counter == 1)
                {
                        loeschen();
                        zeitzahl = stunde /10;
                        ziffern(zeitzahl, 30);                  // darzustellende zahl mit x-Verschiebung
                }
                if (counter == 2)
                {
                        zeitzahl = stunde %10;
                        ziffern(zeitzahl, 42);
                        ziffern(99,60);
```

```
                }
                if (counter == 3)
                {
                        zeitzahl = minute /10;
                        ziffern(zeitzahl, 57);
                }
                if (counter == 4)
                {
                        zeitzahl = minute % 10;
                        ziffern(zeitzahl, 67);
                        pulsweite1 = 12000;
                        pulsweite2 = 11000;
                        break;
                }

                counter++;

        }

        return 1;

}

//*****************************************************************************************************
*************************************
//
        TASTENABFRAGE
uint8_t tastenabfrage (uint8_t verstell, uint16_t wiederholung)
{

        if (!(PIND & (1 << PD5)))                                       // Taste "hoch" wird abgefragt
        {
                verstell = 1;

                if (wiederholung >= 2800)
                {
                        verstell = 4;                   // Verstellwert für gedrückte hochtaste
                }

        }
        if (!(PIND & (1 << PD6)))                                       // Taste "runter" wird abgefragt
        {
                verstell = 2;

                if (wiederholung >= 2800)
                {
                        verstell = 5;                                           // Verstellwert für gedrückte
runtertaste
                }
        }
        if (!(PIND & (1 << PD7)))                                       // Taste "umschalt" wird abgefragt
        {
                verstell = 3;

                if (wiederholung >= 10000)                              // verstellwert für gedrückte
umschalttaste
                {

                        verstell = 6;
                }

        }
```

```
        if ((PIND & (1 << PD5)) && (PIND & (1 <<PD6)) && (PIND & (1 << PD7)))                    //
erst wenn die Taste losgelassen wird  wird der Veränderungswert übernemmen
        {
                switch(verstell)
                {
                        case (1):
                        verstell = 7;                                   // einzelverstellung hoch, Zahl wird im
aufrufenden Programm verändert
                        break;

                        case (2):
                        verstell = 8;                                   // einzelverstellung runter, Zahl wird im
aufrufenden Programm  verändert
                        break;

                        case (3):

                        verstell = 9;                                   //umschaltung  wird im aufrufenden
Programm durchgeführt
                        break;

                }

        }

        return verstell;
        // gibt den verstellwert zurück

}

//*****************************************************************************************************
*************************************

//                                                                                                  7-
SEGMENTANZEIGE DARSTELLUNG
uint8_t anzeige (uint8_t multiplex, uint8_t zahl, uint8_t dotpoint)
{

        uint8_t stelle;
        stelle = 0;

        if (multiplex == 60)                                    // Zurücksetzen des Multiplexzählers
        {
                multiplex = 0;
        }

        if ((multiplex > 0) && (multiplex < 30))                        // die ersten 30 Durchläufe
wird die Einerstelle dargestellt
        {
                PORTB |= (1 << PB0);                                                            //
linke Anzeige aus
                PORTC &= ~(1 << PC0);                                                           //
recht Anzeige ein
                stelle = zahl % 10;
                // Darstellen der Einerstelle
                if(dotpoint == 1)
                {
                        PORTC |= (1 << PC2);
                }
        else
                {
                        PORTC &= ~( 1 << PC2);
                }
```

```
	}
else
	{
		// Die zweiten 30 Durchläufe werden die Zehnerstelle dargestellt
		PORTB &= ~(1 << PB0);                                   // linke Anzeige ein
		PORTC |= (1 << PC0);                                    // recht Anzeige aus
		stelle = zahl/10;                                       // Darstellen der Zehnerstelle
		if(dotpoint == 1)                                       // Darstellen des
Dezimalpunktes
		{
			PORTC |= (1 << PC2);
		}
	else
		{
			PORTC &= ~( 1 << PC2);
		}
	}

	switch (stelle)
	{
		case (1):                                       //Darstellen an der 7-Segmentanzeige
		PORTB |= (1 << PB3);
		PORTC |= (1 << PC1);
		PORTC &= ~((1 << PC3) | (1 << PC4));
		PORTB &= ~((1 << PB2) | ( 1 << PB1) | (1 << PB4));

		break;

		case (2):
		PORTB |= (1 << PB2) | (1 << PB3) | (1 << PB4);
		PORTC |= (1 << PC3) | (1 << PC4);
		PORTB &= ~(1 << PB1);
		PORTC &= ~(1 << PC1);
		break;

		case (3):
		PORTB |= (1 << PB2) | (1 << PB3) | (1 << PB4);
		PORTC |= (1 << PC1) | (1 << PC4);
		PORTB &= ~(1 << PB1);
		PORTC &= ~(1 << PC3);
		break;

		case (4):
		PORTB |= (1 << PB1) | (1 << PB2) | (1 << PB3);
		PORTC |= (1 << PC1);
		PORTC &= ~((1 << PC3) | ( 1 << PC4));
		PORTB &= ~(1 << PB4);

		break;

		case (5):
		PORTB |= (1 << PB1) | (1 << PB2) | (1 << PB4);
		PORTC |= (1 << PC1) | (1 << PC4);
		PORTB &= ~(1 << PB3);
		PORTC &= ~(1 << PC3);
		break;

		case (6):
		PORTB |= (1 << PB1) | (1 << PB2) | (1 << PB4);
		PORTC |= (1 << PC1) | (1 << PC3) | (1 << PC4);
		PORTB &= ~(1 << PB3);
		break;
```

```
                case (7):
                PORTB |= (1 << PB3) | (1 << PB4);
                PORTC |= (1 << PC1) | (1 << PC4);
                PORTC &= ~((1 << PC3) | (1 << PC4));
                PORTB &= ~((1 << PB2) | ( 1 << PB1));

                break;

                case (8):
                PORTB |= (1 << PB1) | (1 << PB2) | (1 << PB3) | (1 << PB4);
                PORTC |= (1 << PC1) | (1 << PC3) | (1 << PC4);
                break;

                case (9):
                PORTB |= (1 << PB1) | (1 << PB2) | (1 << PB3) | (1 << PB4);
                PORTC |= (1 << PC1) | (1 << PC4);
                PORTC &= ~(1 << PC3);
                break;

                case (0):
                PORTB |= (1 << PB1) | (1 << PB3) | (1 << PB4);
                PORTC |= (1 << PC1) | (1 << PC3) | (1 << PC4);
                PORTB &= ~(1 << PB2);
                break;
        }

        multiplex++;                                                                    //
Multiplexzähler um eins höhergesetzt
        return multiplex;                                                               // Zurückgeben des
Multiplexwertes

}

//*************************************************************************************************
************************************
//                                      Übergabe der Uhrzeit an Zählwerk

uint16_t zeitjustiern (uint16_t uhrzeit)
{
        uint8_t stun;
        uint8_t min;

        stun = uhrzeit / 100;                                   // berechnet aus der 4-stelligen uhrzeit die stunden und
minuten und übergibt den wert an die globalen variablen
        min = uhrzeit % 100;

        if ((stun >= 24) || (min >= 60))
        {
                return 1;
        }
else
        {
                stunde = stun;
                minute = min;
        }

        return 1;
}

//                                                                                      Overflowroutine für
TIMER0
```

```
ISR (TIMER0_OVF_vect)
{
	zeitcounter++;

	if (zeitcounter >= 31250)			// ergibt bei 8Mhz und vorteiler 1 = 1 sek
	{
		sekunde++;
		zeitcounter = 0;				// Uhrzählwerk
	}
	if (sekunde == 60)
	{
		minute++;
		sekunde = 0;
	}
	if (minute == 60)
	{
		stunde++;
		minute = 0;
	}
	if (stunde == 24)
	{
		stunde = 0;
	}
}
//*****************************************************************************************************
*********************************
//
	VERGLEICHSÜBEREINSTIMMUNGS ROUTINE
//

ISR (TIMER1_COMPA_vect)
{

	if (periodencounter == 0)
	{
		PORTD &= ~(1 << PD0);			// servoausgang hub auf low gesetzt

	}
	if (periodencounter == 1)
	{
		PORTD &= ~(1 << PD1);			// servoausgang rechts auf low gesetzt

	}
	if (periodencounter == 2)
	{
		PORTD &= ~(1 << PD2);			// servoausgang links auf low gesetzt

	}

}

//*****************************************************************************************************
***************************************
//
	Pulsstartroutine
ISR (TIMER1_OVF_vect)
{
	periodencounter++;

	if (periodencounter == 1)
	{
		PORTD |= (1 << PD1);
		OCR1A = pulsweite1;				// servoausgang rechts auf high gesetzt
```

```
	}

	if (periodencounter == 2)
	{
		PORTD |= (1 << PD2);
		OCR1A = pulsweite2;				// servoausgang links auf high gesetzt
	}
	if (periodencounter == 3)
	{
		PORTD |= (1 << PD0);
		OCR1A = pulsweite3;				// servoausgang HUB auf high gesetzt
		periodencounter = 0;

	}

}
//*****************************************************************************************************
*****************************************
//
	Stellwinkelberechnung
uint8_t stellwinkel (uint16_t koordinaten)		// x und y koordinaten des Punktes werden als 4stellige
zahle übergeben
{
	uint16_t xwert;
	uint16_t ywert;
	uint16_t yhebel;
	uint16_t yarm;
	uint16_t xhebel;
	uint16_t xarm;
	uint16_t stelllinks;
	uint16_t stellrechts;
	stellrechts = 0;
	stelllinks = 0;
	yarm = 0;
	yhebel = 0;
	xhebel = 0;
	xarm = 0;
	uint8_t counter;
	counter = 0;

	uint32_t arm;
	arm = 0;

	if (koordinaten == 0)				// stift in schreibposition gesenkt
	{
		pulsweite3 = 14800;
		return 1;
	}
	if (koordinaten == 101)				// stift in bewgungsposition gehoben
	{
		pulsweite3 = 13000;
		return 1;
	}

	xwert = koordinaten / 100;			// Zahl wird in x und y wert zerlegt
	ywert = koordinaten % 100;

	for(counter = 0; counter < 2; counter++)	// berechnen der servostellwinkel
	{

		for(yhebel = 0; yhebel <= 40; yhebel++)
		{
			yarm = ywert - yhebel;
```

```
                    xhebel = 1600-(yhebel*yhebel);
                    xhebel = wurzel (xhebel);

                    if (counter == 0)
                    {
                        xarm = xwert + xhebel -44;
                        arm = (xarm * xarm) + (yarm * yarm);
                        arm = wurzel(arm);

                        if (arm <= 55)
                        {
                            stelllinks = yhebel*25;
                            stelllinks = winkelfunktion(stelllinks);
                            yhebel = 40;
                        }
                    }
                    if (counter == 1)
                    {

                        xarm = xhebel + 76 - xwert;
                        arm = (xarm * xarm) + (yarm * yarm);
                        arm = wurzel(arm);

                        if (arm <= 55)
                        {
                            stellrechts = yhebel*25;
                            stellrechts = winkelfunktion(stellrechts);
                            yhebel = 40;
                        }
                    }
                }
        }

        stelllinks = 14500 - (stelllinks * 34);                    // übergeben der werte an die globale
servovariable
        stellrechts =  8000 + (stellrechts * 33);

        while(1)
        {
            for (counter = 0; counter < 130; counter++)            // verlangsamt die stellbewegungen
            {
                // zeitschleife
            }
            if (pulsweite1 < stelllinks)
            {
                pulsweite1++;
            }
            if (pulsweite1 > stelllinks)
            {
                pulsweite1--;
            }
            if (pulsweite2 < stellrechts)
            {
                pulsweite2++;
            }
            if (pulsweite2 > stellrechts)
            {
                pulsweite2--;
            }

            if ((pulsweite1 == stelllinks) && (pulsweite2 == stellrechts))
            {
```

```
                        break;
                }
        }

        return 1;

}

//*****************************************************************************************************
*********************************************

//      HAUPTPROGRAMM

int main(void)
{
        DDRB |= (1 << DDB0) | (1 << DDB1) | (1 << DDB2) | (1 << DDB3) | (1 << DDB4);                    //
Ausgänge für 7-SegmentAnzeige
        DDRC |= (1 << DDC0) | (1 << DDC1) | (1 << DDC2) | (1 << DDC3) | (1 << DDC4);
        DDRD |= (1 << DDD0) | (1 << DDD1) | (1 << DDD2) | (1 << DDD3) | (1 << DDD4);                    //
Ausgänge für Servo und LED

        PORTD |= (1 << PD5) | (1 << PD6) | (1 << PD7);
                                // interne Pull-up Widerstände für Tasteneingänge gesetzt

        TIMSK |= (1 << TOIE0);                                  // Overflowinterrupt für timer0 erlaubt
        TCCR0 |= (1 << CS00);                                   // Timer 0 mit CPU-Takt gestartet
        TIMSK |= (1 << TOIE1);                                  // Overflowinterrupt für timer1 erlaubt
        TCCR1B |= (1 << CS10);                                  // Timer1 im CPU takt gestartet

        TIMSK |= (1 << OCIE1A);                                         // Interrupt bei
Übereinstimmung mit Vergleichswert
        PORTD |= (1 << PD0);                                    // Ausgang hub auf high gesetzt

        sei();                                                          // globale Interrupts
zulassen

        OCR1A = 2000;                                                   // Vergleichswert A
voreingestellt auf 2,5msek (staffelung der Signale
        // Vergleichswert B wird auf Mittelstellung servo (1,5msek) PW

        uint8_t verstell;                               // Variablen für Tastenabfrage und Anzeige
        uint16_t wiederholung;
        verstell = 0;
        wiederholung = 0;
        uint8_t multiplex;
        multiplex = 0;
        uint8_t umschalt;
        umschalt = 0;
        uint8_t zahl;
        zahl = 0;
        uint8_t schleifencount;
        schleifencount = 1;
        uint8_t min;
        min = 0;
        // Pulsweiten Mittelwert als Grundeinstellung,
        pulsweite1 = 14500;
        pulsweite2 = 14500;
        pulsweite3 = 13000;
        periodencounter = 0;
        sekunde = 0;
        minute = 0;
        stunde = 0;
```

```
	uint16_t uhrzeit;
	uhrzeit = 0;

	while(1)
	{

		while(schleifencount)
		{
			verstell = tastenabfrage(verstell, wiederholung);		// Tastenabfrage wird aufgerufen

			if ((verstell > 0) && ( verstell < 4))			// Eine Taste wurde gedrück und der wiederholzähler aktiviert
			{
				wiederholung++;
			}
			switch(verstell)
			{
				case(0):
				wiederholung = 0;			// wurde keine Taste gedrückt wird Wiederholzähler auf null gesetzt
				break;

				case(4):
				uhrzeit = uhrzeit + 100;
				verstell = 0;				// Wird die rechte taste gehalten	schnelles hinaufzählen der uhrzeit
				wiederholung = 0;
				break;

				case(5):
				uhrzeit = uhrzeit - 100;
				verstell = 0;				// Wird die mittlere Taste gehalten  schnelles herunterzählen der uhrzeit
				wiederholung = 0;
				break;

				case(6):
				zeitjustiern(uhrzeit);
				schleifencount = 0;
				wiederholung = 0;
				verstell = 0;
								// Wird die Umschalttaste gehalten wird die Uhrzeitvariable an Uhrzählwerk übergeben
				break;

				case(7):
				uhrzeit++;
				verstell = 0;				// wird die rechte taste losgelassen  wird die uhrzeit um eins erhöht
				wiederholung = 0;
				break;

				case(8):
				uhrzeit--;
				verstell = 0;				// wird die mittlere taste losgelassen uhrzeit um 1 verringert
				wiederholung = 0;
				break;

				case(9):
				if(umschalt == 0)
				{
```

```
                                        umschalt = 1;
                                }

                        else
                                {
                                        umschalt = 0;
                                }
                                verstell = 0;                   // Wird die Umschalttaste losgelassen wird in der Anzeige zwischen zehner und tausender anzeige umgeschalten
                                wiederholung = 0;
                                break;
                        }

                        if (umschalt == 0)
                        {
                                zahl = uhrzeit % 100;
                        }
                else
                        {
                                zahl = uhrzeit / 100;
                        }

                        multiplex = anzeige(multiplex,zahl,umschalt);

                }

                schleifencount = 1;

                while(schleifencount)
                {
                        if (min != minute)
                        {
                                min = minute;
                                zeitaufteiler();
                        }

                }

        }
}
```