“Lotto”

eine Projektarbeit in DVT

von Sven Schwab

IAV 2

Nürnberg, den 21.07.2010

Agenda

1) Idee

2) Projektbestandteile
- Software
- Hardware

3) Projektdetails
- Ablauf
- Fehler die auftreten können

4) Quellcode
- mit ausreichenden Kommentaren

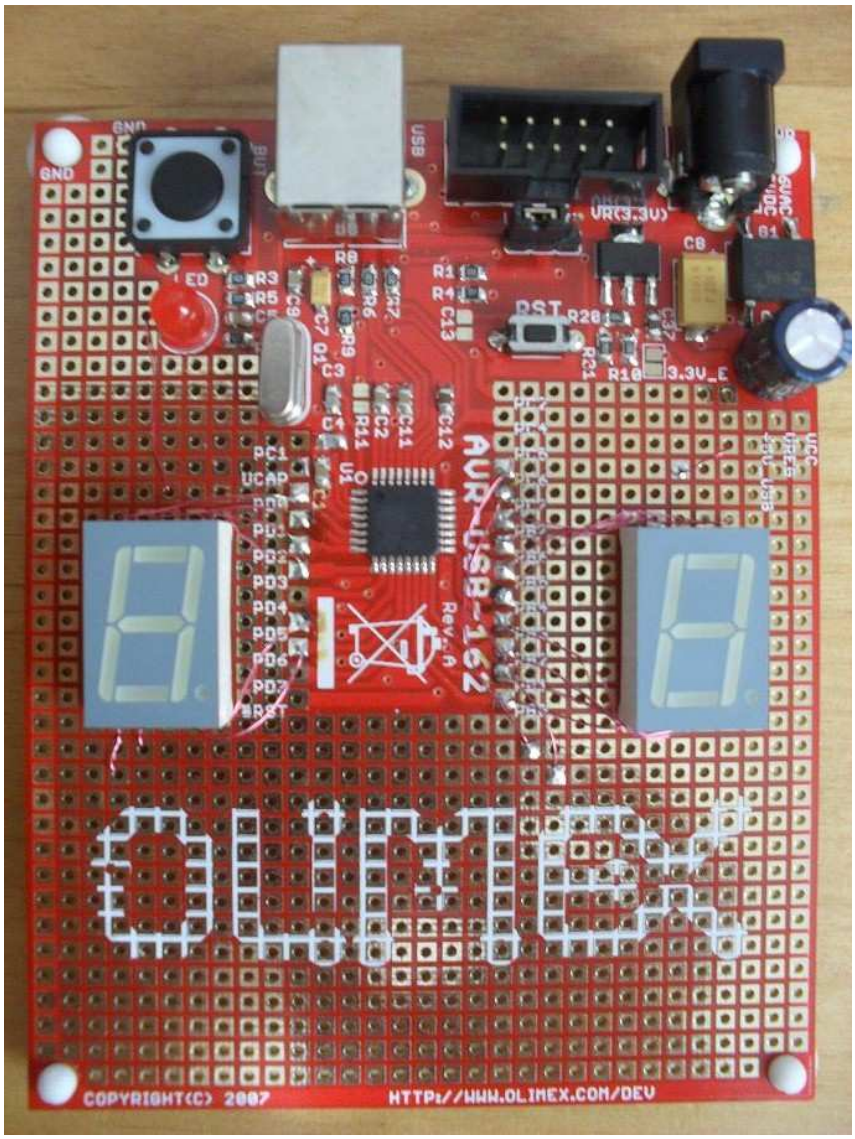
1) Idee

Mit einem Microcontroller sollen Lottozahlen generiert und über 7-Segmentanzeigen ausgegeben werden.

2) Projektbestandteile

Software: AVR Studio 4 und Atmel Flip

Hardware: AVR-USB-162 Board der Fa. OLIMEX mit einem 8-Bit AT90USB162 Mikrocontroller von ATMEL



3) Projektdetails

Der Microcontroller wartet solange bis der Taster gedrückt und losgelassen wird.
Durch den Taster wird die erste Lottozahl, in einer Schleife von 1 – 49, generiert und daraus (über rand()-Funktion) werden die restlichen Zahlen erzeugt.
Nach Erzeugung der Zahlen läuft ein Counter gegen 0 und anschließend werden die Lottozahlen sortiert und an den 7-Segmentanzeigen ausgegeben.
Die Anzeige der Zahlen läuft endlos, bis der MC einen reset bekommt.

Fehler die auftreten können:

1) doppelte Zahlen:

Werden durch eine Schleife und mehrmaligem Prüfen, sowie Neuerzeugung bei einer doppelten, stark reduziert.

Zusatz: Es ist möglich die Anzahl der Schleifendurchgänge zu erhöhen, um die Chance auf doppelte Zahlen noch geringer zu halten.
Getestet wurde mit 5 Schleifendurchgänge und nach ca. 30 Tests, gab es noch keine doppelte Zahl.

2) Unter 1 und über 49:

Fehler wurde durch klar definierte Anweisungen ausgemerzt.

„% 49“ um unter 49 zu bleiben
„+1“ um über 1 zu kommen.

4) Quellcode

```
#include <avr/io.h>
#include <stdlib.h>
#define F_CPU 1000000
#include <util/delay.h>
#define PRUEFUNGEN 5        //Je höher, desto genauer ist die Prüfung auf doppelte

//Array mit den Bits für die einzelnen Zahlen der rechten 7-Seg.Anzeige
static unsigned char seg[] = { 0x77, 0x14, 0xb3, 0xb6, 0xd4, 0xe6, 0xe7, 0x34, 0xf7, 0xf6 };

void zahl(int);                                 //Funktionsprototype für linke 7-Segmentanzeige
void blinken(int);
void lotto_zeigen(int []);
int vgl(const void* el1, const void* el2);      //Sortieranweisung für QSORT

int main(void)
{
        int rando[6];           //Array für Lottozahlen
        int randzahl = 1;       //Zähler für erste Randomzahl
        int gedrueckt = 0;      //Wurden die Zahlen (durch drücken des Tasters) erzeugt?
        int i = 1, j, k, z;     //Zählervariablen
        int itmp = 0;                   //Tempvariable für Einerstelle

        DDRB = 0xff;            //Rechte 7-Seg Ports auf Ausgang
        PORTB = 0x80;           //Linke 7-Seg Ports auf Ausgang

        DDRD &= ~(1 << DDD7);           //Button Port auf Eingang
        PORTD |= (1 << PORTD7);         //Button reagiert auf 0

        DDRC |= ((1 << DDC6) | (1 << DDC7));        //7-Seg Ports auf High (Ausgang)
        DDRD |= ((1 << DDD0) | (1 << DDD1) | (1 << DDD2) | (1 << DDD3) | (1 << DDD4) | (1 << DDD5) | (1 << DDD6));

        while(1)
        {
                if(gedrueckt == 0)      //Solang noch keine Taste gedrückt wurde
                {                               //gibt es ein sinnloses blinken
                        PORTB ^= (1 << PORTB7);
                        PORTD ^= (1 << PORTD0);
                        _delay_ms(200);
                }
                if(gedrueckt == 1)      //Wenn gedrückt wurde, startet der Counter
                {                               //und die Zahlen werden ausgegeben
                        itmp = i / 10;
                        zahl(itmp);                     //Anzeige der Zahl für linke 7-Seg
                        PORTB = seg[(i % 10)];          //Anzeige der Zahl für rechte 7-Seg

                        i--;
                        _delay_ms(100);
                }
                while(!(PIND & (1 << PORTD7)))      //Button setzt srand, erste Lottozahl und
                {                                           //und Timer auf X Sekunden
                        i = 30;                             //30 = 3 Sekunden
                        if(randzahl >= 49)
                                randzahl = 1;
                        randzahl++;
```

```
                gedrueckt = 1;
        }

        srand(randzahl+586);            //Initialisieren des Startwerts für rand()
        rando[0] = randzahl;                    //Erzeugen der Zufallszahlen
        rando[1] = 1+(rand() % 49);             //+1 und %49, damit die Zahlen im Lottobereich sind
        rando[2] = 1+(rand() % 49);
        rando[3] = 1+(rand() % 49);
        rando[4] = 1+(rand() % 49);
        rando[5] = 1+(rand() % 49);

        for(z = 0; z < PRUEFUNGEN; z++)         //Generell werden die Zahlen
        {                                       //X mal geprüft
                for(k = 1; k < 6; k++)
                {
                        for(j = 0; j < 6; j++)
                        {
                                if(j == k)      //Eigenen Vergleich vermeiden,
                                        j++;            //sonst Endlosschleife
                                if(rando[k] == rando[j])
                                {
                                        rando[k] = 1+(rand() % 49);
                                        z--;            //Falls doppelte vorkam, noch
                                }                       // einmal mehr prüfen
                        }
                }
        }
        //Wenn i = 0 (counter) und Taste gedrückt, anzeigen der Lottozahlen
        if(i <= 0 && gedrueckt == 1)
        {
                blinken(7);
                lotto_zeigen(rando);
        }
    }
    return 0;
}
void lotto_zeigen(int zahlen[])         //Lottozahlen
{
    int i = 0, j = 0;
    int zehnertmp, einertmp;

    qsort(zahlen, 6, sizeof(int), vgl);             //Zahlen aufsteigend sortieren

    while(1)                                        //Endlosanzeige der Zahlen
    {
        for(i = 0; i < 6; i++)
        {
                zehnertmp = zahlen[i] / 10;             //Vorbereitung der Zehnerstelle
                einertmp = zahlen[i] % 10;              //Vorbereitung der Einerstelle
                zahl(zehnertmp);                                //Anzeige der Zehnerstelle
                PORTB = seg[einertmp];                          //Anzeige der Einerstelle
                _delay_ms(1500);
        }
        blinken(5);
    }
}
```

```
void blinken(int blinkanzahl)                //Funktion fürs blinken
{
    //Alle Led’s werden erstmal ausgeschalten
    PORTC &= ~((1 << PORTC6) | (1 << PORTC7));
    PORTD &= ~((1 << PORTD0) | (1 << PORTD1) | (1 << PORTD2) | (1 << PORTD3) | (1 << PORTD5) | (1 << PORTD6));

        int i = 0;
        //Anfang - Rundlauf
                PORTB = 0x00;
                PORTB |= (1 << PORTB2);
                PORTD |= (1 << PORTD6);
                _delay_ms(200);
                PORTB |= (1 << PORTB1);
                PORTC |= (1 << PORTC7);
                _delay_ms(200);
                PORTB |= (1 << PORTB0);
                PORTC |= (1 << PORTC6);
                _delay_ms(200);
                PORTB |= (1 << PORTB7);
                PORTD |= (1 << PORTD0);
                _delay_ms(200);
                PORTB |= (1 << PORTB4);
                PORTD |= (1 << PORTD3);
                _delay_ms(200);
                PORTB |= (1 << PORTB5);
                PORTD |= (1 << PORTD2);
                _delay_ms(200);
                PORTB |= (1 << PORTB6);
                PORTD |= (1 << PORTD1);
                _delay_ms(200);
        //Ende Rundlauf
        //Anfang blinken
        for(i = 0; i < blinkanzahl; i++)
        {
                PORTC &= ~((1 << PORTC6) | (1 << PORTC7));
                PORTD &= ~((1 << PORTD0) | (1 << PORTD1) | (1 << PORTD2) | (1 << PORTD3) | (1 << PORTD4) |
                                (1 << PORTD5) | (1 << PORTD6));
                if(i % 2 == 0)
                {
                        PORTB = 0x77;
                        PORTC |= ((1 << PORTC6) | (1 << PORTC7));
                        PORTD |= ((1 << PORTD1) | (1 << PORTD2) | (1 << PORTD3) | (1 << PORTD6));
                }
                else
                {
                        PORTB = 0x80;
                        PORTD |= (1 << PORTD0);
                        PORTD |= (1 << PORTD4);
                }
                _delay_ms(100);
        }
        //Ende blinken
}
void zahl(int i)  //Funktion für die Zahl der linken 7-Seg. Durch verschiedene Register war kein Array möglich
{
        PORTC &= ~((1 << PORTC6) | (1 << PORTC7));
```

```
	PORTD &= ~((1 << PORTD0) | (1 << PORTD1) | (1 << PORTD2) | (1 << PORTD3) | (1 << PORTD5) | (1 << PORTD6));
	switch(i)
	{
		case 1:
			PORTD |= ((1 << PORTD6) | (1 << PORTD3));
			break;
		case 2:
			PORTD |= ((1 << PORTD2) | (1 << PORTD3) | (1 << PORTD0));
			PORTC |= ((1 << PORTC6) | (1 << PORTC7));
			break;
		case 3:
			PORTD |= ((1 << PORTD2) | (1 << PORTD3) | (1 << PORTD0) | (1 << PORTD6));
			PORTC |= ((1 << PORTC7));
			break;
		case 4:
			PORTD |= ((1 << PORTD1) | (1 << PORTD0) | (1 << PORTD3) | (1 << PORTD6));
			break;
		case 5:
			PORTD |= ((1 << PORTD2) | (1 << PORTD1) | (1 << PORTD0) | (1 << PORTD6));
			PORTC |= ((1 << PORTC7));
			break;
		case 6:
			PORTD |= ((1 << PORTD2) | (1 << PORTD1) | (1 << PORTD0) | (1 << PORTD6));
			PORTC |= ((1 << PORTC6) | (1 << PORTC7));
			break;
		case 7:
			PORTD |= ((1 << PORTD6) | (1 << PORTD3) | (1 << PORTD2));
			break;
		case 8:
			PORTC |= ((1 << PORTC6) | (1 << PORTC7));
			PORTD |= ((1 << PORTD0) | (1 << PORTD1) | (1 << PORTD2) | (1 << PORTD3) | (1 << PORTD6));
			break;
		case 9:
			PORTD |= ((1 << PORTD2) | (1 << PORTD1) | (1 << PORTD0) | (1 << PORTD6) | (1 << PORTD3));
			PORTC |= ((1 << PORTC7));
			break;
		case 0:
			PORTC |= ((1 << PORTC6) | (1 << PORTC7));
			PORTD |= ((1 << PORTD1) | (1 << PORTD2) | (1 << PORTD3) | (1 << PORTD6));
			break;
	}
}
int vgl(const void* el1, const void* el2)		//Definieren der Sortierreihenfolge für QSORT
{
	int *l1, *l2;
	l1 = (int*)el1;
	l2 = (int*)el2;

	if(*l1 == *l2)
		return 0;
	else
		if(*l1 < *l2)
			return -1;
		else
			return 1;
}
```