```
/*
 * LED_Cube.c
 *
 * Created: 20.04.2017 21:08:49
 *  Author: Roland
 */

#include "LED_Cube.h"

#include <avr/iotn2313.h>
#include <avr/interrupt.h>
#include <util/delay.h>

void set_ebene(u8 lx);
void clr_ebene(u8 lx);
void set_line(u8 lx, u8 bmask);
void clr_line(u8 lx, u8 bmask);
void set_all(void);
void clr_all(void);
void set_point(u8 a,u8 b,u8 c);
void clr_point(u8 a,u8 b,u8 c);

u16 data[3];
u8  pointer;
u8  pointer2;
u8  wa;
u8  mc;

// ------------------------------------------------------
// Portbelegung
// ------------------------------------------------------
// Port A 0 ......... P5
//        1 ......... P8
//        2 ......... RsT
// -------------------------
// Port B 0 ......... P1
//        1 ......... P0
//        2 ......... P2
//        3 ......... P3
//        4 ......... P4
//        5 ......... Mosi
//        6 ......... Miso
//        7 ......... Slk
// -------------------------
// Port D 0 .........
//        1 ......... P6
//        2 ......... Taste
//        3 ......... Z1
//        4 ......... Z2
//        5 ......... Z3
//        6 ......... P7
// ------------------------------------------------------
// Px Lage
// ------------------------------------------------------
//    +--------------------+
//    | Spannung           |    Linezuordnung
//    |                    |
//    | P6      P7      P8 |    L2      L5      L8
//    |                    |
//    |                    |
//    | P3      P4      P5 |    L1      L4      L7
//    |                    |
//    |                    |
//    | P0      P1      P2 |    L0      L3      L6
//    |                    |
//    | Taster     (ISP)   |   (Z1)    (Z2)    (Z3)
//    +--------------------+
//      Z1 ... unten
//      Z2 ... mitte
//      Z3 ... oben
// ------------------------------------------------------

// ------------------------------------------------------
```

```
// main
// -----------------------------------------------------------
int main(void)
{
    cli();

    // Port A
    DDRA = 0x03;    // 2Bit Output
    PORTA = 0x00;

    // Port B
    DDRB = 0x1F;    // 5Bit Output
    PORTB = 0x00;

    // Port D
    DDRD = 0x7A;    // 2Bit Output
    PORTD = 0x00;   // 3Bit Treiber

    // Timer_Counter_0
    OCR0A = 0x80;           // TV
    TCCR0A = (1 << WGM01);  // CTC
    TIFR |= (1 << OCF0A);
    TIMSK = (1 << OCIE0A);
    TCCR0B = (1 << CS01);   // VT 8

    // Taster INT0
    MCUCR |= (1 << ISC01);  // fallende Flanke
    GIMSK |= (1 << INT0);   // Taster

    pointer = 0;
    pointer2 = 0;
    wa = 2;
    mc = 0;

    sei();

    set_all();
    _delay_ms(500);
    clr_all();
    _delay_ms(500);

    u8* px = (u8*) pgm_read_word( &(musterTab[mc]) );
    u8 cmd,x,m,n,j;

    while(1)
    {
        cmd = pgm_read_byte( px );
        px++;
        switch (cmd)
        {
        // Alles
        case 'A':
            set_all();
            break;
        case 'B':
            clr_all();
            break;
        // Ebene
        case 'C':
            x = pgm_read_byte( px );
            px++;
            set_ebene(x);
            break;
        case 'D':
            x = pgm_read_byte( px );
            px++;
            clr_ebene(x);
            break;
        // Line
        case 'E':
            x = pgm_read_byte( px );
            px++;
            m = pgm_read_byte( px );
```

```
            px++;
            set_line(x,m);
            break;
        case 'F':
            x = pgm_read_byte( px );
            px++;
            m = pgm_read_byte( px );
            px++;
            clr_line(x,m);
            break;
        // Piont
        case 'G':
            x = pgm_read_byte( px );
            px++;
            m = pgm_read_byte( px );
            px++;
            n = pgm_read_byte( px );
            px++;
            set_point(x,m,n);
            break;
        case 'H':
            x = pgm_read_byte( px );
            px++;
            m = pgm_read_byte( px );
            px++;
            n = pgm_read_byte( px );
            px++;
            clr_point(x,m,n);
            break;
        // Pause
        case 'P':
            for(j=0; j<wa; j++){
                _delay_ms(100);
            }
            break;
        // Wait
        case 'W':
            x = pgm_read_byte( px );
            px++;
            for(j=0; j<x; j++){
                _delay_ms(100);
            }
            break;
        // Muster Ende (Schluß)
        case 'X':
            mc++;
            px = (u8*) pgm_read_word( &(musterTab[mc]) );
            break;
        // Liste zu Ende
        case 'Z':
            mc = 0;
            px = (u8*) pgm_read_word( &(musterTab[mc]) );
        // No Operation
        default:
            break;
        }
    }
    return 0;
}

// ------------------------------------------------------
void set_all(void)
{
    set_ebene(0);
    set_ebene(1);
    set_ebene(2);
}

// ------------------------------------------------------
void clr_all(void)
{
    clr_ebene(0);
    clr_ebene(1);
```

```
    clr_ebene(2);
}

// -------------------------------------------------------
void set_ebene(u8 lx)
{
    x_ebene( lx, 1);
}

// -------------------------------------------------------
void clr_ebene(u8 lx)
{
    x_ebene( lx, 0);
}

// -------------------------------------------------------
void x_ebene(u8 lx, u8 wx)
{
    u8 i;
    u16 a;
    u16 mask = 0;

    if(lx<3){
        if( wx ){
            data[lx] = 0x01FF;
        }else{
            data[lx] = 0x0000;
        }
    }else{
        lx -= 3;
        lx *= 3;
        mask = (0x07 << lx );
/*
        if( wx ){
            data[0] |= mask;
            data[1] |= mask;
            data[2] |= mask;
        }else{
            mask = !mask;
            data[0] &= mask;
            data[1] &= mask;
            data[2] &= mask;
        }
*/
        for(i=0;i<3;i++){
            a = data[i];
            if( wx ){
                a |= mask;
            }else{
                a &= (!mask);
            }
            data[i] = a;
        }
    }
}

// -------------------------------------------------------
void set_ledline(u8 grp, u8 pos, u8 bmask)
{
    u16 a = data[grp];
    switch (pos)
    {
    case 0:
        if( (bmask&1)>0 ) a |= 0x0001;
        if( (bmask&2)>0 ) a |= 0x0002;
        if( (bmask&4)>0 ) a |= 0x0004;
        break;
    case 1:
        if( (bmask&1)>0 ) a |= 0x0008;
        if( (bmask&2)>0 ) a |= 0x0010;
        if( (bmask&4)>0 ) a |= 0x0020;
        break;
    case 2:
```

```
        if( (bmask&1)>0 ) a |= 0x0040;
        if( (bmask&2)>0 ) a |= 0x0080;
        if( (bmask&4)>0 ) a |= 0x0100;
        break;
    }
    data[grp] = a;
}

// ------------------------------------------------------
void set_line(u8 lx, u8 bmask)
{
    switch (lx)
    {
    case 0:
    case 1:
    case 2:
        set_ledline(0,lx,bmask);
        break;
    case 3:
    case 4:
    case 5:
        set_ledline(1,lx-3,bmask);
        break;
    case 6:
    case 7:
    case 8:
        set_ledline(2,lx-6,bmask);
        break;
    }
}

// ------------------------------------------------------
void clr_ledline(u8 grp, u8 pos, u8 bmask)
{
    u16 a = data[grp];
    switch (pos)
    {
    case 0:
        if( (bmask&1)>0 ) a &= 0xFFFE;
        if( (bmask&2)>0 ) a &= 0xFFFD;
        if( (bmask&4)>0 ) a &= 0xFFFB;
        break;
    case 1:
        if( (bmask&1)>0 ) a &= 0xFFF7;
        if( (bmask&2)>0 ) a &= 0xFFEF;
        if( (bmask&4)>0 ) a &= 0xFFDF;
        break;
    case 2:
        if( (bmask&1)>0 ) a &= 0xFFBF;
        if( (bmask&2)>0 ) a &= 0xFF7F;
        if( (bmask&4)>0 ) a &= 0xFEFF;
        break;
    }
    data[grp] = a;
}

// ------------------------------------------------------
void clr_line(u8 lx, u8 bmask)
{
    switch (lx)
    {
    case 0:
    case 1:
    case 2:
        clr_ledline(0,lx,bmask);
        break;
    case 3:
    case 4:
    case 5:
        clr_ledline(1,lx-3,bmask);
        break;
    case 6:
    case 7:
```

```
    case 8:
        clr_ledline(2,lx-6,bmask);
        break;
    }
}

void x_point(u8 a,u8 b,u8 c,u8 d)
{
    u16 mask = 0;
    switch(a)
    {
    case 0:
        mask = 0x01;
        break;
    case 1:
        mask = 0x02;
        break;
    default:
        mask = 0x04;
        break;
    }
    switch(b)
    {
    case 1:
        mask = (mask << 3);
        break;
    case 2:
        mask = (mask << 6);
        break;
    default:
        break;
    }
    if(d){
        data[c] |= mask;
    }else{
        data[c] &= (!mask);
    }
}

void set_point(u8 a,u8 b,u8 c)
{
    x_point(a,b,c,1);
}

void clr_point(u8 a,u8 b,u8 c)
{
    x_point(a,b,c,0);
}

// ------------------------------------------------------
// Taster
// ------------------------------------------------------
ISR (SIG_INTERRUPT0 )
{
    sei();
    while ( (PIND & (1 << PIND2)) == 0 )
    {
        _delay_ms(100);
    }
    wa++;
    if(wa>10) wa=1;
}

// ------------------------------------------------------
// MULTIPLEXAUSGABE
// ------------------------------------------------------
ISR (SIG_TIMER0_COMPA)
{
    u16 a;
    u8  mask1  = 0;
    u8  mask2  = 0;
    u8  mask3  = 0;
```

```
    PORTB &= 0xE0; // 5 LED = AUS
    PORTD &= 0x05; // 2 LED + 3 Ebenen = AUS
    PORTA &= 0x40; // 2 LED = AUS

    if ( pointer2== 1 ){
        pointer2 = 0;
        return;
    }
    pointer2++;

    a = data[pointer];

    if( (a & 0x0001)>0 ) mask1 |= 0x02; //P0
    if( (a & 0x0002)>0 ) mask1 |= 0x01; //P1
    if( (a & 0x0004)>0 ) mask1 |= 0x04; //P2
    if( (a & 0x0008)>0 ) mask1 |= 0x08; //P3
    if( (a & 0x0010)>0 ) mask1 |= 0x10; //P4
    if( (a & 0x0020)>0 ) mask3 |= 0x01; //P5
    if( (a & 0x0040)>0 ) mask2 |= 0x02; //P6
    if( (a & 0x0080)>0 ) mask2 |= 0x40; //P7
    if( (a & 0x0100)>0 ) mask3 |= 0x02; //P8

    PORTB |= mask1;
    PORTD |= mask2;
    PORTA |= mask3;

    switch (pointer)
    {
    case 0:
        PORTD |= 0x08;
        pointer++;
        break;
    case 1:
        PORTD |= 0x10;
        pointer++;
        break;
    case 2:
        PORTD |= 0x20;
        pointer=0;
        break;
    }
}
```