```
/*
 * PanaIR_rx.c
 *
 * Author : EH   Originally created: 2017.01.04
 * Receive specific remote Panasonic TV IR bit patterns, consisting of
 * frames of 48 bits e.g.  16 bit address + 32 bit data
 * Pressing a key on the Panasonic remote, it sends two identical frames
 * lasting approx. 63 msec. and approx. 75 msec. in between the two frames.
 * NOTE: The remote control PWR toggle button transmits 3 identical frames.
 *
 * The hardware is using an Attiny85 connected to 4 PNP 12V output transistors.
 * Refer to https://larsenhenneberg.dk for further details on the implementation.
 *
 * Panasonic remote buttons being decoded:
 * --------
 * IR code actions for 3D, left arrow, right arrow:
 * Pressing 3D twice toggles the PWM signals to the REDLEDPWM and GREENLEDPWM outputs
 * Note: the REDLEDPWM has the opposite duty cycle of GREENLEDPWM and vice versa
 * While the PWM signal is enabled, pressing left arrow will decrease the PWM duty cycle
 * While the PWM signal is enabled, pressing right arrow will increase the PWM duty cycle
 *
 * IR code action for the yellow button:
 * Pressing the yellow button twice toggles the YELLOWLED output pin
 *
 * IR code action for the blue button:
 * Pressing the blue button twice toggles the BLUELED output pin
 * --------
 *
 * 2018.11.22 - rewrite the code to include a state machine (STM) for button sequences
 * 2018.12.20 - left and right arrow only work from preceeding 3D, left-arrow,
 right-arrow
 * 2019.01.05 - included an extra STM state to refine LIGHT (GREENLEDPWM) control
 * 2020.09.25 - outputs are now active high, hardware version nov. 2020
 * 2020.12.21 - added remote yellow, blue buttons to toggle YELLOWLED and BLUELED outputs
 */

# define F_CPU 1000000UL    // 1 MHz clock with DIV8 fuse enabled
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/eeprom.h>
#include <util/delay.h>

// variables used by ISR for IR frame decoding
volatile unsigned int NextBit;
volatile unsigned int RecdAddress;
volatile unsigned long RecdDataH;
volatile unsigned long RecdDataL;
volatile unsigned long newFrame;

volatile uint8_t newFrameFlag;

#define PanaAddress 0x4004          // Remote Panasonic TV IR address code
#define but3Dbutton 0x0190ED7C      // Remote 3D button IR data code
#define leftButton 0x01007273       // Remote volume down button IR data code
#define rightButton 0x0100F2F3      // Remote volume down button IR data code
#define yellowButton 0x01004E4F     // Remote yellow button IR data code
#define blueButton 0x0100CECF       // Remote blue button IR data code

#define OCRfreq 152         // OCR1B PWM upper count , approx. 100 Hz

#define INT0_PIN PORTB2     // IR receiver active low output attiny pin 7
#define BLUELED PORTB0      // blue led active high output attiny pin 5
#define YELLOWLED PORTB1    // yellow led active high output attiny pin 6
#define REDLEDPWM PORTB3    // red led PWM output attiny pin 2
#define GREENLEDPWM PORTB4  // green led PWM output attiny pin 3

int main(void)
{
    volatile uint8_t state, OCR1Bsave;
```

```
69
70      DDRB &= _BV(INT0_PIN);      // set the INT0_PIN pin as an input
71      PORTB |= _BV(INT0_PIN);     // pull up resistor on INT0_PIN input
72
73      DDRB |= _BV(BLUELED);       // set the BLUELED pin as an output
74      PORTB &= ~_BV(BLUELED);     // turn off the output
75
76      DDRB |= _BV(YELLOWLED);     // set the YELLOWLED pin as an output
77      PORTB &= ~_BV(YELLOWLED);   // turn off the output
78
79      TCCR0A = 0;
80      TCCR0B = 3<<CS00;           // T0 Prescaler /64 , 64 usec. timer
81
82      GTCCR = 1<<PWM1B | 1<<COM1B0; // enable PWM1B, clear OCR1B output line
83      TCCR1 = 7<<CS10;            // set Timer1 clock to clk/64
84
85      OCR1Bsave = OCRfreq / 2;    // initial duty cycle (50%)
86      OCR1B = OCRfreq;            // no PWM at start
87      OCR1C = OCRfreq;
88
89      MCUCR |= (1<<ISC01);        // Interrupt on falling edge
90      GIMSK |= _BV(INT0);         // Enable external interrupt INT0
91
92      NextBit = 48;
93      newFrameFlag = 0;           // clear new frame indicator flag
94      state = 0;                  // start state for the IR button state-event handler
95
96      sei();                      // enable global interrupts
97
98      while (1)
99      {
100      if (newFrameFlag == 1)          // got a new frame
101      {
102         cli();                       // disable frame interrupt while decoding frame
         data
103         if (RecdAddress == PanaAddress)  //  frame with the Panasonic TV address ?
104         {
105             switch (state)
106             {
107                 case 0:
108                     switch (newFrame)
109                     {
110                         case but3Dbutton: state = 1; break;
111                         case yellowButton: state = 3; break;
112                         case blueButton: state = 4; break;
113                     }
114                 break;
115                 case 1:
116                     switch (newFrame)
117                     {
118                         case yellowButton: state = 3; break;
119                         case blueButton: state = 4; break;
120                         case but3Dbutton:
121                             if (OCR1B == OCRfreq)       // is PWM disabled
122                             {
123                                 DDRB |= _BV(GREENLEDPWM);   // set the GREENLED pin as
                                 an output
124                                 DDRB |= _BV(REDLEDPWM); // set the REDLEDPWM pin as an
                                 output
125                                 OCR1B = OCR1Bsave;      // enable PWM with previous
                                 setting
126                                 state = 2;
127                             }
128                             else
129                             {
130                                 DDRB &= _BV(GREENLEDPWM);   // set the GREENLED pin as
                                 an input (no PWM out !)
131                                 DDRB &= _BV(REDLEDPWM); // set the REDLEDPWM pin as an
                                 input (no PWM out !)
```

```
                                OCR1Bsave = OCR1B;      // save PWM setting
                                OCR1B = OCRfreq;        // disable PWM
                                state = 0;
                            }
                        break;
                        default: state = 0;
                    }
                break;
                case 2:         // PWM out is active ( from state 1), check for left-
                and right-arrow
                    switch (newFrame)
                    {
                        case yellowButton: state = 3; break;
                        case blueButton: state = 4; break;
                        case rightButton: if (OCR1B > 15) OCR1B = OCR1B - 10; break; //
                        increase duty-cycle
                        case leftButton:  if (OCR1B < OCRfreq - 20) OCR1B = OCR1B + 10;
                        break;
                        case but3Dbutton: state = 1; break;
                        default: state = 0;
                    }
                break;
                case 3: // enter here when the yellow button pressed once from state 0
                    switch (newFrame)
                    {
                        case but3Dbutton: state = 1; break;
                        case blueButton: state = 4; break;
                        case yellowButton: PORTB ^= _BV(YELLOWLED); state = 0; break;
                        default: state = 0;
                    }
                break;
                case 4: // enter here when the blue button pressed once from state 0
                    switch (newFrame)
                    {
                        case but3Dbutton: state = 1; break;
                        case yellowButton: state = 3; break;
                        case blueButton: PORTB ^= _BV(BLUELED); state = 0; break;
                        default: state = 0;
                    }
                break;
            }
        }
        _delay_ms(150);                 // delay for one interframe break plus one frame
        newFrameFlag = 0;               // clear newFrame flag
        sei();                          // enable interrupt for next IR frame
     }
    }
}

// Interrupt service routine, invoked on every falling edge of PB2, e.g. start of an IR
pulse
// Timer0 increments of 64 usec are used for measuring IR pulse width

ISR(INT0_vect) {
  int BitTime = TCNT0;
  int Overflow = TIFR & 1<<TOV0;
  if (NextBit == 48)    // looking for the Panasonic header period
  {                     // in between 4700 and 5700 usec.
    if ((BitTime >= 73) && (BitTime <= 89) && (Overflow == 0))
    {
      RecdAddress = 0;
      RecdDataH = 0;
      RecdDataL = 0;
      NextBit = 0;
    }   // got header, now ready to receive 48 bit data
  }
  else
  {
    if ((BitTime > 30) || (Overflow != 0))
```

```
197         NextBit = 48; // max bit period exceeded, restart
198       else
199       {
200         if (BitTime > 15) // if bit time > "0"-time, e.g. add "1" into the bit position
201         {
202             if (NextBit <= 15) RecdAddress = RecdAddress | ((unsigned int)
                1<<(15-NextBit));
203             else
204             {
205               if ((NextBit <= 31) && (NextBit > 15)) RecdDataH = RecdDataH | ((unsigned
                  int) 1<<(15-(NextBit-16)));
206               else RecdDataL = RecdDataL | ((unsigned int) 1<<(15-(NextBit-32)));
207             }
208         }
209         NextBit++;
210         if (NextBit == 48)
211         {
212           newFrame = (RecdDataH << 16) | (RecdDataL & 0x0000FFFF);
213           newFrameFlag = 1;
214         }
215       }
216     }
217     TCNT0 = 0;            // Clear counter
218     TIFR |= 1<<TOV0;      // Clear overflow
219     GIFR |= 1<<INTF0;     // Clear INT0 flag
220   }
221
```