```
/*
 * KirkTest.c
 * Created: 02-01-2022 20:00:03
 * Author : Erik
 * Program for a modified Arduino Nano with a ATmega328P/ATmega168PA
   microcontroller
 * to provide processing logic between interface electronics to a POT ( Kirk F61 )
 * and a Nokia 220 4G LTE mobil phone.
*/

#define F_CPU 2E6   // CPU clock 2 MHz with DIV8 fuse set with a 16 MHz x-tal osc.
#include <avr/io.h>
#include <stdio.h>
#include <util/delay.h>
#include <avr/interrupt.h>

void touchNumbButton(uint8_t key);
void touchONOFFbutton(void);
void touchLIFTbutton(void);
void touchUNLOCKbutton(void);
void touchSTARbutton(void);

void closeCall(void);
void closeDial(void);
void powerdownNokia(void);

void bellsEnable(void);
void bellsDisable(void);

#define ringONtime  2000                // Bell on time in msec
#define ringOFFtime 2000                // Bell off time in msec.
#define nokiapausescreentime 10000      // Nokia pause screen time in msec.
#define ringONdefault ringONtime/16     // bell on timer 16 msec ticks
#define ringOFFdefault ringOFFtime/16   // bell off timer 16 msec ticks
#define pausescreendefault nokiapausescreentime/16  // Nokia pause screen time
  ticks

#define maxDigitTime 15000 // max digit user dial period in msec
#define touchONperiod 100
#define touchOFFmin 200
#define maxDigits 20        // max phone dial digits
#define phone_no_of_digits 8 // dk number of phone digits

#define HOOK PORTD0     // Kirk F61 hook switch I/F
#define RING PORTD1     // Ring output control for Kirk bell coils I/F
#define DIAL PORTD2     // Kirk F61 dial switch I/F (INT0)
#define INGOING PORTD3      // Nokia "vibrator" ingoing call indicator pin
#define RING31HZ PORTD4     // 31.25 Hz output for ring gen.
#define BUT_STAR PORTD6     // Nokia button * control port pin
```

```
#define BUT_UNLOCK PORTD7   // Nokia button Unlock control port pin
#define ONOFF PORTB5        // Nokia on/off control port pin
#define LIFT PORTB4     // Nokia lift control port pin
#define BUT3 PORTB3     // Nokia button 3 control port pin
#define BUT2 PORTB2     // Nokia button 2 control port pin
#define BUT1 PORTB1     // Nokia button 1 control port pin
#define BUT6 PORTB0     // Nokia button 6 control port pin
#define BUT5 PORTC0     // Nokia button 5 control port pin
#define BUT4 PORTC1     // Nokia button 4 control port pin
#define BUT9 PORTC2     // Nokia button 9 control port pin
#define BUT8 PORTC3     // Nokia button 8 control port pin
#define BUT7 PORTC4     // Nokia button 7 control port pin
#define BUT0 PORTC5     // Nokia button 0 control port pin

// pin bit read macros
#define hookPIN (PIND & 0x01)       // mask out PORTD0 - HOOK
#define ingoingPIN (PIND & 0x08)    // mask out PORTD3 - INGOING

enum statename {idle, ringing, call_in, call_out, dialling };
enum statename state = idle;

static uint8_t dialled_digits[maxDigits], digit_idx=0;

static volatile uint8_t dial_ready = 0, dial_ticks = 0, dial_timeout = 0,
  dialled_number = 0;
static volatile uint8_t ring_enable = 0, ringONcnt = ringONdefault , ringOFFcnt =
  0;
static volatile uint8_t pausescreenTimeout = 0;

static volatile uint16_t pausescreenCnt = 0;

//static uint16_t last_dial_timeout_cnt = 0;

int main(void)
{
    ADCSRA = 0;  // disable all ADC subsystems for power saving
    DDRB = 0x3F; // PB0..PB5 as outputs for Nokia 220 4G button emulation
    DDRC = 0x3F; // PC0..PC5 as outputs for Nokia 220 4G button emulation
    DDRD = 0xD2; // Inputs, except PD1 (RING), PD4 (RING31HZ), PD6 og PD7 as
      outputs

    PORTB &= ~_BV(ONOFF);       // ONOFF control low
    PORTB &= ~_BV(LIFT);        // LIFT control low
    PORTB &= ~_BV(BUT3);        // Button '3' control low
    PORTB &= ~_BV(BUT2);        // Button '2' control low
    PORTB &= ~_BV(BUT1);        // Button '1' control low
    PORTB &= ~_BV(BUT6);        // Button '6' control low
    PORTC &= ~_BV(BUT5);        // Button '5' control low
    PORTC &= ~_BV(BUT4);        // Button '4' control low
```

```
    PORTC &= ~_BV(BUT9);        // Button '9' control low
    PORTC &= ~_BV(BUT8);        // Button '8' control low
    PORTC &= ~_BV(BUT7);        // Button '7' control low
    PORTC &= ~_BV(BUT0);        // Button '0' control low
    PORTD &= ~_BV(BUT_UNLOCK);  // Button 'Unlock' control low
    PORTD &= ~_BV(BUT_STAR);    // Button '*' control low
    PORTD &= ~_BV(RING);        // disable ring bell power
    PORTD &= ~_BV(RING31HZ);    // disable ring gen. interval pin
    PORTD |= _BV(DIAL);         // weak pull up at DIAL input

    TCCR0A = 0b00000010;        // timer 0 mode 2 - CTC
    TCCR0B = 0b00000100;        // set prescaler to 256
    OCR0A = 125;                // no of ticks, Output Compare Register
    TIMSK0 = 0b00000010;        // trigger interrupt when ctr (TCNT0) >= OCR0A

    EICRA |= (1<<ISC01);        // set INT0 to trigger on falling edge
    EIMSK |= (1 << INT0);     // Turns on INT0 interruption

    sei();                  // Enable interrupts

    // assume the nokia is powered down at this point
    PORTB |= _BV(ONOFF);        // ON/OFF button press
    _delay_ms(6000);            // delay to turn on
    PORTB &= ~_BV(ONOFF);       // the Nokia 220 4G phone and
    _delay_ms(27000);           // wait for Nokia pause screen
    touchONOFFbutton();         // touch ON/OFF to wake up screen

    while(1) //loop forever
    {
        switch (state)
        {
            case idle:  if (ingoingPIN != 0)    // if ingoing call detected
                        {
                            state = ringing;
                            bellsEnable();
                        }
                        else
                        {
                            if (hookPIN == 0)   // if hook lifted
                            {
                                digit_idx=0;                // prepare dialling
                    state
                                dial_ready = 0;
                                dialled_number = 0;
                                state = dialling;
                             }
                        }
            break;
            case ringing:   if (ingoingPIN == 0)    // if ingoing call unanswered
```

```
                        {
                            bellsDisable();
                            state = idle;       // back to idle
                        }
                        else
                        {                       // still ingoing call request
                            if (hookPIN == 0)   // if hook lifted, conversation started
                            {
                                touchLIFTbutton();
                                bellsDisable();
                                state = call_in;    // continue to call_in
                            }
                        }
                        pausescreenCnt = pausescreendefault;    // preset pause screen timeout count
                        pausescreenTimeout = 0;
                break;
                case call_in:   // call_in / call_out: close call if hook down and goto idle
                case call_out:  if (hookPIN != 0)           // if hook down
                        {
                            closeCall();
                            state = idle;       // back to idle
                        }
                break;
                case dialling:  if (hookPIN != 0)           // if hook down
                        {
                            if (digit_idx != 0)
                            {
                                closeDial();
                            }
                            state = idle;       // back to idle
                        }
                        else
                        {
                            if (dial_ready != 0)
                            {
                                dial_ready = 0;
                                touchNumbButton(dialled_number);
                                dialled_digits[digit_idx++] = dialled_number;
                                dialled_number = 0;
                                if (digit_idx == 3)
                                {
                                    if ((dialled_digits[0] == 0) && (dialled_digits[1] == 0) && (dialled_digits[2] == 0))
                                    {
                                        closeDial();
                                        powerdownNokia();
```

```
                                        state = idle;
                                    }
                                }
                                if (digit_idx == phone_no_of_digits)
                                {
                                    touchLIFTbutton(); // button LIFT to call
                    to the dialled number
                                    digit_idx = 0;
                                    state = call_out;
                                }
                            }
                        }
            break;
        }   // switch
    }   // while
}   // main

// closes an in-/outgoing Nokia call: use ON/OFF button, enter lock screen and
// leave lock screen to sync the state machine with the Nokia screen
void closeCall(void)
{
    if (pausescreenTimeout == 1)
    {
        touchONOFFbutton();         // Nokia light up from pause screen
    }
    touchONOFFbutton();                 // Nokia hook down or lock screen
_delay_ms(900);
    touchONOFFbutton();                 // make sure to enter lock screen
_delay_ms(900);
    touchUNLOCKbutton();                // make sure to enter menu screen
    touchUNLOCKbutton();                // do unlock screen with
    touchSTARbutton();                  // with the * button as well
}

// closes dialling, BEFORE the dialled phone number is called
void closeDial(void)
{
    if (pausescreenTimeout == 1)
    {
        touchONOFFbutton();         // Nokia light up from pause screen
    }
    touchONOFFbutton();     // empty Nokia number buffer and goto main screen
}

void touchUNLOCKbutton(void)
{
    PORTD |= _BV(BUT_UNLOCK);               // set UNLOCK button control high
    _delay_ms(touchONperiod);               // control signal high period
    PORTD &= ~_BV(BUT_UNLOCK);              // set UNLOCK button control low
```

```
    _delay_ms(touchOFFmin);                 // keep control signal low period
    pausescreenCnt = pausescreendefault;    // preset pause screen timeout count
    pausescreenTimeout = 0;
}

void touchSTARbutton(void)
{
    PORTD |= _BV(BUT_STAR);                 // set * button control high
    _delay_ms(touchONperiod);               // control signal high period
    PORTD &= ~_BV(BUT_STAR);                // set * button control low
    _delay_ms(touchOFFmin);                 // keep control signal low period
    pausescreenCnt = pausescreendefault;    // preset pause screen timeout count
    pausescreenTimeout = 0;
}

/* touchONOFFbutton()  emulates the touch of the ON/OFF button in order to:
   1. wake up the pause screen
   2. close an ongoing call */
void touchONOFFbutton(void)
{
    PORTB |= _BV(ONOFF);                    // set ON/OFF button control high
    _delay_ms(touchONperiod);               // control signal high period
    PORTB &= ~_BV(ONOFF);                   // set ON/OFF button control low
    _delay_ms(touchOFFmin);                 // keep control signal low period
    pausescreenCnt = pausescreendefault;    // preset pause screen timeout count
    pausescreenTimeout = 0;
}
/* touchLIFTbutton()
   emulates the touch of the LIFT button in order to start an ingoing call
*/
void touchLIFTbutton(void)
{
    PORTB |= _BV(LIFT);         // LIFT button touch
    _delay_ms(touchONperiod);   // control signal high period
    PORTB &= ~_BV(LIFT);        // the Nokia 220 4G
    _delay_ms(touchOFFmin);     // keep control signal low period
    pausescreenCnt = pausescreendefault;    // preset pause screen timeout count
    pausescreenTimeout = 0;
}

void powerdownNokia(void)
{
    PORTB |= _BV(ONOFF);        // ON/OFF button press
    _delay_ms(3000);            // delay for closing
    PORTB &= ~_BV(ONOFF);       // down Nokia 220 4G
    _delay_ms(6000);            //
}

void bellsEnable(void)
```

```
{
    ringONcnt = ringONdefault;  // restart bell on timer
    ringOFFcnt = 0;             // null bell off timer
    PORTD |= _BV(RING);         // enable ring bell power
    ring_enable = 1;            // start the bells, flag for timer0 int.
}

void bellsDisable(void)
{
    ring_enable = 0;            // stop the bells, !flag for timer0 int.
    PORTD &= ~_BV(RING);        // disable ring bell power
    PORTD &= ~_BV(RING31HZ);    // set ring gen. interval pin low
    ringONcnt = ringONdefault;
    ringOFFcnt =  0;
}

void touchNumbButton(uint8_t key)
{
  if (pausescreenTimeout == 1) touchONOFFbutton();  // wake up the display if
    paused
  switch (key)
  {
    case 0 : PORTC |= _BV(BUT0); _delay_ms(touchONperiod); PORTC &= ~_BV(BUT0);
      _delay_ms(touchOFFmin); break;
    case 1 : PORTB |= _BV(BUT1); _delay_ms(touchONperiod); PORTB &= ~_BV(BUT1);
      _delay_ms(touchOFFmin); break;
    case 2 : PORTB |= _BV(BUT2); _delay_ms(touchONperiod); PORTB &= ~_BV(BUT2);
      _delay_ms(touchOFFmin); break;
    case 3 : PORTB |= _BV(BUT3); _delay_ms(touchONperiod); PORTB &= ~_BV(BUT3);
      _delay_ms(touchOFFmin); break;
    case 4 : PORTC |= _BV(BUT4); _delay_ms(touchONperiod); PORTC &= ~_BV(BUT4);
      _delay_ms(touchOFFmin); break;
    case 5 : PORTC |= _BV(BUT5); _delay_ms(touchONperiod); PORTC &= ~_BV(BUT5);
      _delay_ms(touchOFFmin); break;
    case 6 : PORTB |= _BV(BUT6); _delay_ms(touchONperiod); PORTB &= ~_BV(BUT6);
      _delay_ms(touchOFFmin); break;
    case 7 : PORTC |= _BV(BUT7); _delay_ms(touchONperiod); PORTC &= ~_BV(BUT7);
      _delay_ms(touchOFFmin); break;
    case 8 : PORTC |= _BV(BUT8); _delay_ms(touchONperiod); PORTC &= ~_BV(BUT8);
      _delay_ms(touchOFFmin); break;
    case 9 : PORTC |= _BV(BUT9); _delay_ms(touchONperiod); PORTC &= ~_BV(BUT9);
      _delay_ms(touchOFFmin); break;
  }
  pausescreenCnt = pausescreendefault;  // preset pause screen timeout count
  pausescreenTimeout = 0;
}

// Dial interrupt, counts number of dial pulses and restart "last-pulse" time-out
ISR(INT0_vect) {
```

```
    if (dial_ready == 0)
    {
        dialled_number++;
        dial_ticks = 6;     // last dial pulse time-out TIMER0 ticks: 6*16 msec. =
          96 msec.
    }
}

//  16 msec. timer0 interrupt.
// The ISR provides the following functions:
// 1. Checks if the ring generator signal is to be generated
// 2. Flags timeout on the dial pulse counter
// 3. Flags timeout on the pause screen counter

ISR(TIMER0_COMPA_vect) {
    if (ring_enable != 0)
    {
        if (ringONcnt != 0)             // ring bell ON time interval ?
        {
            PORTD ^= _BV(RING31HZ);     // toggle 31.25 Hz ring (bell) generator
              port pin
            ringONcnt--;
            if (ringONcnt == 0)             // end of ring bell ON time interval ?
            {
                ringOFFcnt = ringOFFdefault;    // restart bell off timer
                PORTD &= ~_BV(RING);            // disable ring bell power
                PORTD &= ~_BV(RING31HZ);    // set 31.25 Hz ring (bell) generator
                  port pin low
            }
        }
        if (ringOFFcnt != 0)                    // ring bell OFF time interval ?
        {
            ringOFFcnt--;
            if (ringOFFcnt == 0)                // end of ring bell OFF time
              interval ?
            {
                ringONcnt = ringONdefault;      // restart bell on timer
                PORTD |= _BV(RING);             // enable ring bell power
            }
        }
    }

    // check time-out on last rotary dial pulse and flag on time-out
    if (dial_ticks != 0 )
    {
        dial_ticks--;
        if (dial_ticks == 0)
        {
            if (dialled_number == 10) dialled_number = 0;
```

```
            dial_ready = 1;
        }
    }

    if (pausescreenCnt != 0)
    {
        pausescreenCnt--;
        if (pausescreenCnt == 0)
        {
            pausescreenTimeout = 1;
        }
    }
}
```