```
/*
 * Kirk_F61_Nokia-220-4G.c
 *
 * Intial version created: 2022.01.02 20:00:03
 * Author : Erik, HW details at https://larsenhenneberg.dk/
 * Program for a modified Arduino Nano with an ATmega328P/ATmega168PA
 * to provide processing logic between interface electronics to the
 * Kirk F61 rotary dial phone and a Nokia 220 4G LTE mobile phone.
 * 2022.03.06: dialling 050 to power on Nokia, dial 040 to power down
 * 2022.03.14: changed button emulator i/f for button emul. hw v2
 * 2022.03.19: changed synchronization Nokia power on and power off
 * 2022.04.11: improved start up sync. after dialling 050
 * 2022.04.12: added the sleep state for CPU power down
 * 2022.05.22: a bit more specific contents of the "user manual" ;)
 * 2022.06.18: extended timer0 ISR to avoid the 200 msec. ingoing call
 *             input glitch to interrupt the bell ringing intervals.
 * 2022.06.28: extended timer0 ISR to tick the cpu sleep on hook down
 *             and wake up on hook up or on ingoing call
 *
 * User manual:
 * 1: power on the I/F board with Arduino Nano + button emul. board v.2
 * 2: insert the Nokia 220 4G battery
 * 3: power on Nokia: lift hook, dial 050, hook down.. WAIT for approx. 30 sec.
 * 4: ready for ingoing calls or to dial out ( outgoing call at 8 dialled digits )
 * 5: power down Nokia and Nano: lift hook, dial 040, hook down
 * 6: lift hook to power up Nano and WAIT for approx. 30 sec. to start Nokia
 * 7: hook down to continue the state machine from step 4
*/
// ATmega168PA CPU clk 2 MHz: DIV8 fuse set with a 16 MHz x-tal.
// *** fuse settings: EXTENDED=0xF8, HIGH=0xDD, LOW=0x7F
//  Initialize at startup:
//  TCCR0B = 0b00000100;    // set prescaler to 256
//  OCR0A = 125;            // no of counts for 16 msec. tmr0 ISR
//#define F_CPU 2E6

// Atmega328P, CPU clk 460750Hz: DIV8 fuse set with a 3,686 MHz x-tal.
// *** fuse settings: EXTENDED=0xFF, HIGH=0xDA, LOW=0x7D
//  Initialize at startup:
//  TCCR0B = 0b00000011;    // set prescaler to 64
//  OCR0A = 115;            // no of counts for 16 msec. tmr0 ISR
#define F_CPU 460750

// Atmega328P, CPU clk 500000Hz: DIV8 fuse set with a 4 MHz x-tal.
// *** fuse settings: EXTENDED=0xFF, HIGH=0xDA, LOW=0x7D
//  Initialize at startup:
//  TCCR0B = 0b00000011;    // set prescaler to 64
//  OCR0A = 125;            // no of counts for 16 msec. tmr0 ISR
//#define F_CPU 500000
```

```
#include <avr/io.h>
#include <stdio.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>

void buttonEmulate(uint8_t buttonVal);  // Nokia button touch emulation
void buttonSet(uint8_t bVal);           // low level Nokia button emulation I/F

void closeCall(void);                   // close an in-/outgoing call
void powerdownNokia(void);              // power down Nokia
void poweronNokia(void);                // power on Nokia and sync. to statemachine

void bellsEnable(void);                 // enable HW signals to the bell ring
  generator
void bellsDisable(void);                // disable HW signals to the bell ring
  generator

#define ringONtime  1900                // Bell on time in msec
#define ringOFFtime 2300                // Bell off time in msec.
#define hookSleepTime 900000            // Time to CPU sleep after hook down in
  msec. ( 15 min.)
#define sleepNowTicks hookSleepTime/16  // hook down cpu sleep timer ticks
#define ingoingGlitchTime   240         // ingoing call input glitch time in msec.
#define nokiapausescreentime 10000      // Nokia pause screen timeout value in
  msec.
#define ringONdefault ringONtime/16     // bell on timer 16 msec ticks
#define ringOFFdefault ringOFFtime/16   // bell off timer 16 msec ticks
#define ingoingGlitchTicksDefault   ingoingGlitchTime/16    // ingoing call input
  glitch 16 msec. ticks
#define pausescreendefault nokiapausescreentime/16  // Nokia pause screen time
  ticks

#define maxDigits 20            // max phone dial digits
#define touchONperiod 150       // button emulating ON delay in msec.
#define touchOFFmin 250         // button emulating OFF/release delay in msec.
#define phone_no_of_digits 8    // number of phone digits for Denmark inland

#define HOOK PORTD0      // Kirk F61 hook switch port pin
#define RING PORTD1      // Ring gen. enable for Kirk bell coils port pin
#define DIAL PORTD2      // Kirk F61 dial switch port pin (INT0)
#define INGOING PORTD3   // Nokia "vibrator" ingoing call indicator port pin
#define RING31HZ PORTD4  // 31.25 Hz square wave output for ring gen. port pin

// non-dial button emulation values for the button emulator hw v2 decoder
#define ONOFFval 10     // Nokia ON/OFF button emulate value
#define LIFTval 11      // Nokia LIFT button emulate value
#define UNLOCKval 12    // Nokia UNLOCK button emulate value
#define STARval 13      // Nokia * button emulate value
```

```
#define GATEval 14      // Nokia # button emulate value

#define Strobe 0x10     // PortC.bit4 = decoder strobe at button emul. hw v2
#define Inhibit 0x20    // PortC.bit5 = decoder inhibit at button emul. hw v2

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

// statemachine emunerations, state names
enum statename {idle, ringing, call_in, call_out, dialling, sleep };
enum statename state = idle;

uint8_t dialled_digits[maxDigits], digit_idx=0;

uint8_t dial_ready = 0, dial_ticks = 0, dialled_number = 0;
uint8_t sleepNow = 0, ring_enable = 0,  ringONcnt = ringONdefault , ringOFFcnt =
  0;
uint8_t ingoingCall = 0, ingoingGlitchTicks = 0;
uint8_t pausescreenTimeout = 1; // Assume Nokia pause screen timeout at Nokia power
   on

uint16_t pausescreenCnt = 0;        // Nokia pause screen tick timing variable
uint16_t sleepTimer = sleepNowTicks;    // cpu sleep tick timing variable

int main(void)
{
    ADCSRA = 0;  // disable all ADC subsystems for power saving
    DDRC = 0x3F; // PC0..PC5 as outputs for Nokia 220 4G button emulation
    DDRD = 0x12; // Inputs, except PD1 (RING), PD4 (RING31HZ)

    PORTC = Inhibit;            // inhibit button emul. hw decoder
    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 = 0b00000011;        // set prescaler to 64
    OCR0A = 115;                // no of counts for 16 msec. tmr0 ISR
    TIMSK0 = 0b00000010;        // trigger interrupt when ctr (TCNT0) >= OCR0A

    EICRA |= (1 << ISC01);      // set INT0 to trigger on falling edge
    EIMSK |= (1 << INT0);       // Enable INT0 interrupt ( on dial digit )

    EICRA |= (1 << ISC11);      // set INT1 to trigger on falling edge
    EIMSK |= (1 << INT1);       // Enable INT1 interrupt ( on ingoing call )

    sei();                  // Enable interrupts
```

```
    while(1) //loop forever
    {
        switch (state)
        {
            case idle:  if (ingoingCall == 1)   // 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;
                             }
                        }
                        if (sleepNow == 1)  // hook down for enough time for cpu
                      sleep ?
                        {
                            state = sleep;
                        }
                break;
                case ringing:   if (ingoingCall == 0)   // if ingoing call unanswered
                                {
                                    bellsDisable();
                                    state = idle;       // back to idle
                                }
                                else
                                {                       // still ingoing call request
                                    if (hookPIN == 0)   // if hook lifted, conversation
                                 started
                                    {
                                        buttonEmulate(LIFTval); // emulate hook lift
                                        bellsDisable();
                                        state = call_in;    // continue to call_in
                                    }
                                }
                                  // during ringing the Nokia screen is active => preset
                              screen timeout timer
                                       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)
                                {
                                    closeCall();
                                }
                                state = idle;       // back to idle
                            }
                            else
                            {
                                if (dial_ready != 0)
                                {
                                    dial_ready = 0;
                                    buttonEmulate(dialled_number);
                                    dialled_digits[digit_idx++] = dialled_number;
                                    dialled_number = 0;
                                    if (digit_idx == 3)
                                    {
                                        if ((dialled_digits[0] == 0) &&
                    (dialled_digits[1] == 4) && (dialled_digits[2] == 0))
                                        {
                                            while (hookPIN == 0);   // wait for
                    hook down
                                            closeCall();
                                            powerdownNokia();
                                            state = sleep;
                                        }
                                        if ((dialled_digits[0] == 0) &&
                    (dialled_digits[1] == 5) && (dialled_digits[2] == 0))
                                        {
                                            poweronNokia();
                                            while (hookPIN == 0);   // wait for
                    hook down
                                            state = idle;
                                        }
                                    }
                                    if (digit_idx == phone_no_of_digits)
                                    {
                                        buttonEmulate(LIFTval); // button LIFT to
                    call to the dialled number
                                        digit_idx = 0;
                                        state = call_out;
                                    }
```

```
                                }
                            }
            break;
            case sleep:
                set_sleep_mode(SLEEP_MODE_PWR_DOWN);
                cli();
                PCICR |= 0b00000100;    // turn on bit change interrupt at port D
                PCMSK2 |= 0b00001001;   // turn on pin PCINT16 (hook) and PCINT19
                  (ingoing)
                sleep_enable();         // sets the SE (sleep enable) bit
                sei();
                sleep_cpu();            // sleep now and.. wake up on PD0 bit
                  change
    // *** HERE the cpu wakes up after call to PCINT2 ISR ( hook lifted ) ***
                sleep_disable();        // deletes the SE bit
                cli();
                sleepNow = 0;           // unflag the cpu sleep timeout flag
                sleepTimer = sleepNowTicks;  // reset the cpu sleep timeout counter
                sei();
                state = idle;           // enter the state machine loop after wake
                  up
            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)
    {
        buttonEmulate(ONOFFval);            // Nokia light up from pause screen
    }
    buttonEmulate(ONOFFval);                // Nokia hook down or lock screen
    _delay_ms(900);
    buttonEmulate(ONOFFval);                // make sure to enter lock screen
    _delay_ms(900);
    buttonEmulate(UNLOCKval);               // make sure to enter menu screen
    buttonEmulate(UNLOCKval);               // do unlock screen with
    buttonEmulate(STARval);                 // with the * button as well
}

void powerdownNokia(void)
{
    PORTC = ONOFFval + Strobe + Inhibit;
    _delay_ms(1);               // latching ONOFFVal at decoder strobe
    PORTC = ONOFFval + Inhibit;
    _delay_ms(1);
```

```
    PORTC = ONOFFval;
    _delay_ms(3000);            // ON/OFF button emulation delay to close Nokia
    PORTC = ONOFFval + Inhibit;
    _delay_ms(6000);            // wait for Nokia pause screen
}

void poweronNokia(void)
{
    PORTC = ONOFFval + Strobe + Inhibit;
    _delay_ms(1);               // latching ONOFFVal at decoder strobe
    PORTC = ONOFFval + Inhibit;
    _delay_ms(1);
    PORTC = ONOFFval;
    _delay_ms(6000);            // ON/OFF button delay to turn on Nokia
    PORTC = ONOFFval + Inhibit;
    _delay_ms(27000);           // wait for Nokia pause screen timeout
}

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;            // flag timer0 ISR to ring the bells
}

void bellsDisable(void)
{
    ring_enable = 0;
    PORTD &= ~_BV(RING);        // disable ring bell power
    PORTD &= ~_BV(RING31HZ);    // set ring generator pin low
    ringONcnt = ringONdefault;
    ringOFFcnt =  0;
}

void buttonEmulate(uint8_t buttonVal)
{
    if (pausescreenTimeout == 1) buttonSet(ONOFFval);   // wake up the display
    buttonSet(buttonVal);                               // if paused
    pausescreenCnt = pausescreendefault;    // preset pause screen timeout count
    pausescreenTimeout = 0;
}

// set values for button emulation hw v2
void buttonSet(uint8_t bVal)
{
    PORTC = (bVal & 0x0f) + Strobe + Inhibit;
    _delay_ms(1);               // latching buttonVal at decoder strobe
    PORTC = (bVal & 0x0f) + Inhibit;
```

```
    _delay_ms(1);
    PORTC = (bVal & 0x0f);
    _delay_ms(touchONperiod);
    PORTC = (bVal & 0x0f) + Inhibit;
    _delay_ms(touchOFFmin);
}

// Dial interrupt, counts the dial pulses, restarts "last-pulse" time-out
ISR(INT0_vect) {
    if (dial_ready == 0)
    {
        dialled_number++;   // a pulse from dial pulse mechanism might
        dial_ticks = 6;     // be last -> reset the time-out to:
    }                       // Timer0 ticks: 6*16 msec. = 96 msec.
}

// The 16 msec. timer0 ISR provides the following functions:
// 1. Flags timeout on the cpu sleep time counter
// 2. Check ingoing call input change and the glitch time-out
// 3. Checks if the ring generator signal is to be generated
// 4. generate the 31.25 Hz bell output in intervals
// 5. Flags timeout on the dial pulse counter
// 6. Flags timeout on the pause screen counter
ISR(TIMER0_COMPA_vect) {
    if (hookPIN == 0)  // hook lifted ?
    {
        sleepTimer = sleepNowTicks; // reset sleep cpu timer
    }
    else   // hook down
    {
        if ((sleepNow == 0) && (sleepTimer != 0))
        {
            sleepTimer--;
            if (sleepTimer == 0)
            {
                sleepNow = 1;       // flag time to cpu sleep
            }
        }
    }
    if (ingoingPIN == 0)               // When no ingoing signal
    {
        if (ingoingGlitchTicks != 0)    //  check ingoing glicth timer
        {
            ingoingGlitchTicks--;
            if ((ingoingGlitchTicks == 0) && (ingoingPIN == 0)) ingoingCall = 0;
            else ingoingCall = 1;
        }
    }
    else ingoingCall = 1;
```

```
    if (ring_enable == 1)           // Ring bell signal generator enabled ?
    {
        if (ringONcnt != 0)             // Check for ongoing ring ON interval
        {
            ringONcnt--;
            if (ringONcnt == 0)              // Ring bell ON interval expired ?
            {                                //  Yes, bell ON interval has expired
                ringOFFcnt = ringOFFdefault; //  restart bell off timer
                PORTD &= ~_BV(RING);         //  disable ring bell power
                PORTD &= ~_BV(RING31HZ);     //  set 31.25 Hz ring (bell) gen. port
                   pin low
            }
            else
            {                               //  NO:
                PORTD ^= _BV(RING31HZ);     //   toggle 31.25 Hz ring (bell)
                  generator port pin
            }
        }
        else
        {
            if (ringOFFcnt != 0)                    // Check for ongoing ring OFF
              interval
            {
                ringOFFcnt--;
                if (ringOFFcnt == 0)                // ring bell OFF interval
                  expired ?
                {                                   //  Yes:
                    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;
        }
    }

    // check time-out for the Nokia pause screen
    if (pausescreenCnt != 0)
    {
```

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

// invoked on bit change at PD0 ( Kirk F61 hook switch port pin )
// takes place after entering power down mode in state "sleep"
ISR(PCINT2_vect)
{
    PCICR &= ~0b00000100;   // turn off bit change interrupt at port D
    PCMSK2 &= ~0b00001001;  // turn off pins PD0, PCINT16, PCINT19
}

// invoked on falling edge of port pin input for ingoing calls
ISR (INT1_vect)
{
    ingoingGlitchTicks = ingoingGlitchTicksDefault;
}
```