X36SKD

lab

Interrupts, joystick

Miroslav Skrbek, Josef Hlaváč

Literature

[1] *8-bit AVR® Instruction Set*, Technical literature of ATMEL Corporation, 2002. http://www.atmel.com

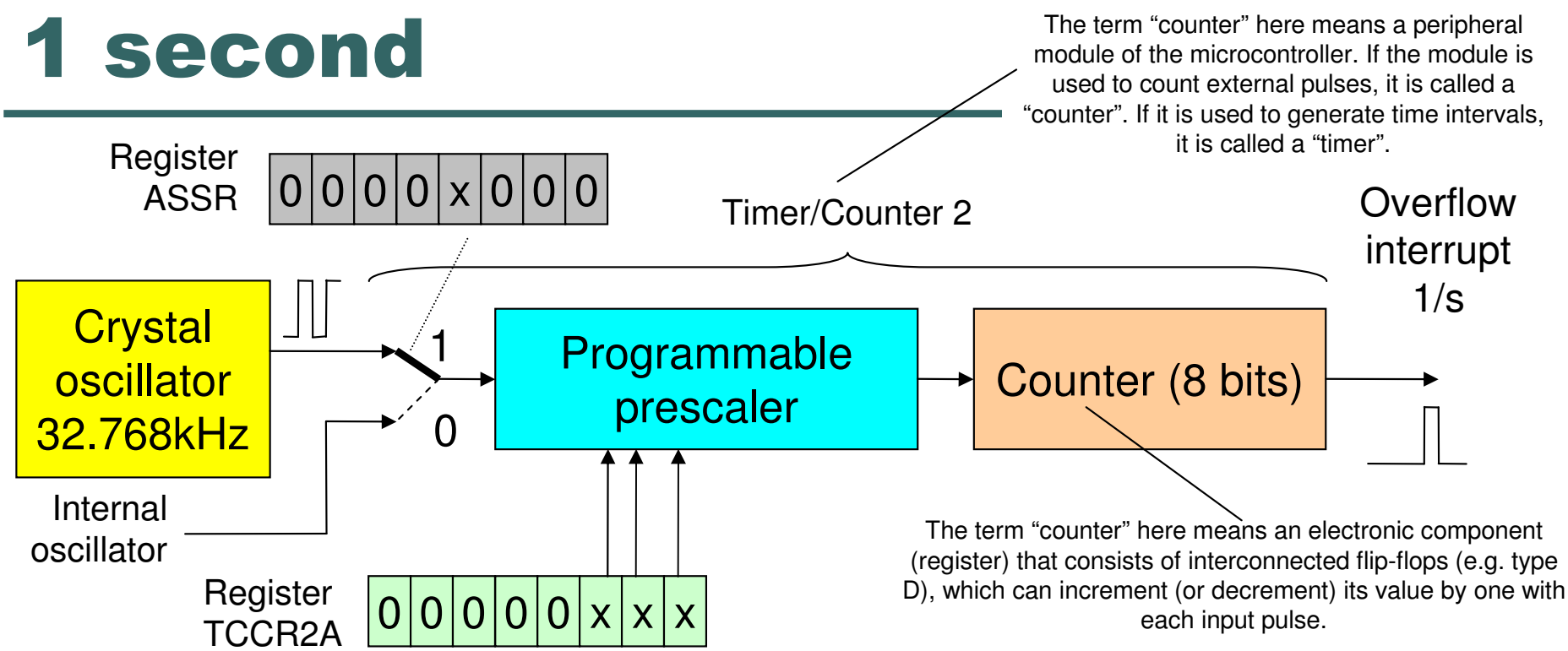
[2] *8-bit AVR® Microcontroller width 16K Bytes In-System Programmable Flash ATmega169*. Datasheet. Technical literature of ATMEL Corporation, 2003. http://www.atmel.com

[3] *AVR Assembler User Guide*. Technical literature of ATMEL Corporation, 2002. http://www.atmel.com

Task: Write a program that increments a displayed number (0, 1, ..., 9) every second.

- Use external oscillator 32.768kHz
- Use Timer 2 with prescaler
- Use Timer/Counter 2 Overflow interrupt
- Use the `show_char` subroutine from the last lab

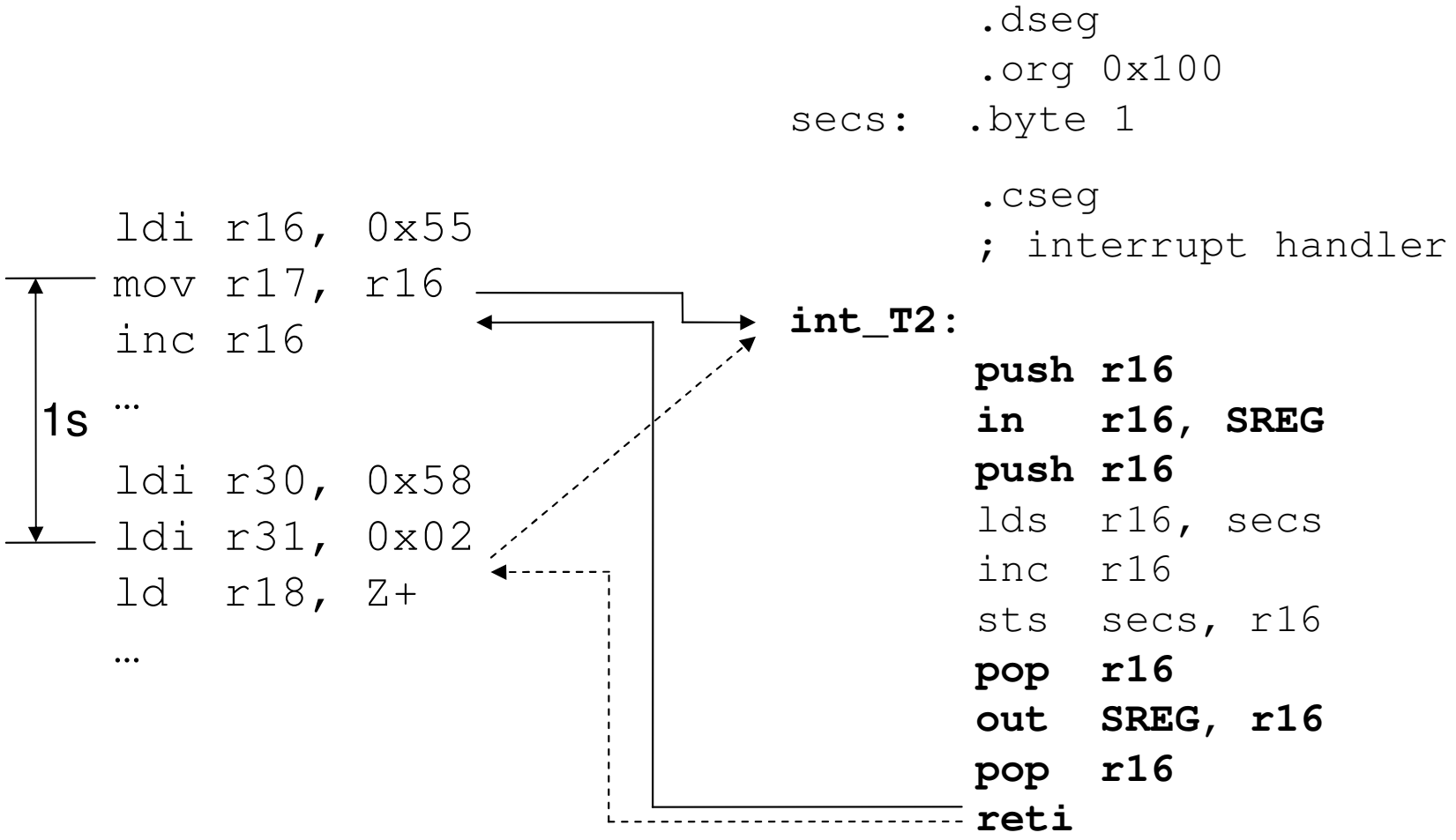
Interrupt source with a period of 1 second



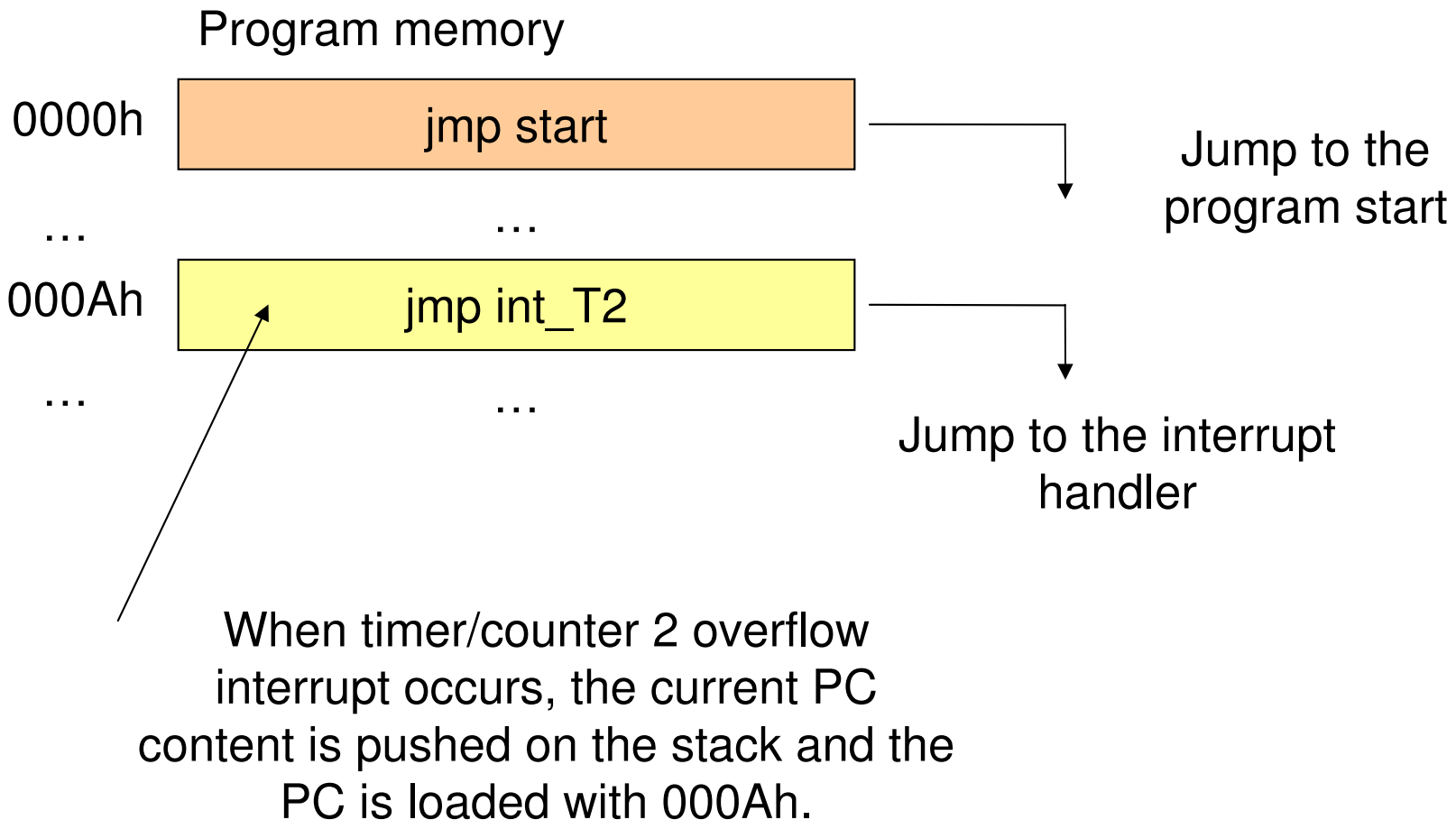

```
000b – stop
001b – divide by 1
010b – divide by 8
011b – divide by 32
100b – divide by 64
101b – divide by 128
110b – divide by 256
111b – divide by 1024
```

Question: What ratio should be selected for the prescaler ?

Interrupt when timer/counter 2 overflows



Where is the interrupt vector for timer/counter 2 overflow?



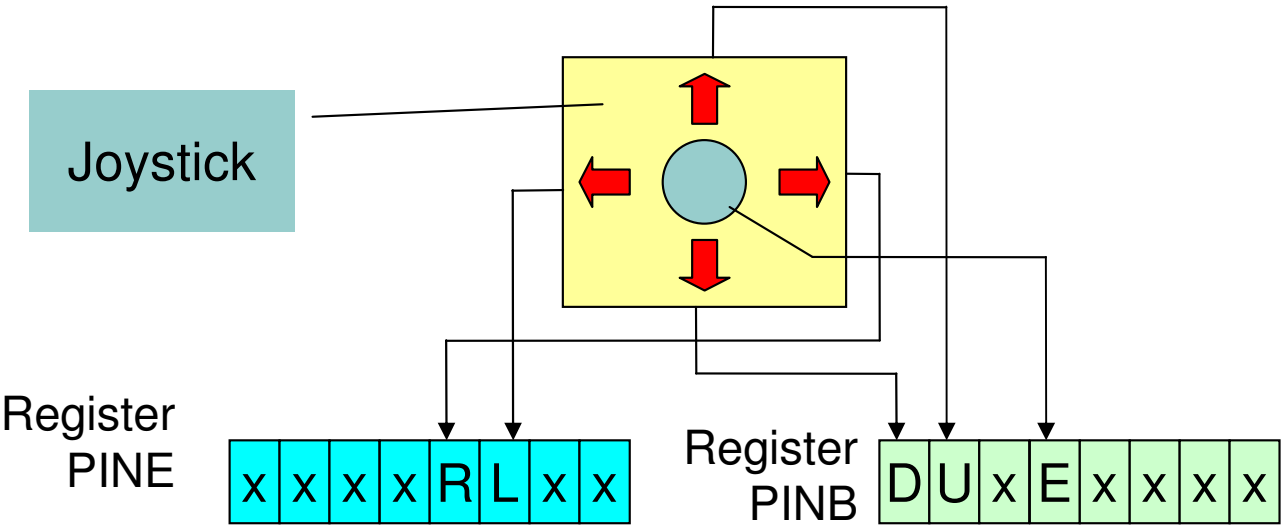
Timer/counter initialization

```
.include "m169def.inc"

.cseg
.org    0x0000
jmp     start
.org    0x000A
jmp     int_T2


.org    0x0100

ldi     r16, low(RAMEND)
out     SPL, r16
ldi     r16, high(RAMEND)
out     SPH, r16
```

```
        cli         ; disable interrupts globally

        ldi r16, 0b00001000
        sts ASSR, r16     ; select clock source
                          ; xtal oscillator
                          ; 32768 Hz
        ldi r16, 0b00000001
        sts TIMSK2, r16   ; enable interrupt
                          ; from timer 2
        ldi r16, 0b00000101
        sts TCCR2A, r16   ; start counter,
                          ; presc. ratio 128

        clr r16           ; disable joystick
        sts PCMSK0, r16   ; interrupts

        sei       ; enable interrupts globally

        … further instructions …
        jmp …

int_T2: … interrupt handler …
        reti
```

Continue solving the task yourselves

Joystick control



bit/direction	value 0	value 1
R (right)	pressed	released
L (left)	pressed	released
U (up)	pressed	released
D (down)	pressed	released
E (Enter)	pressed	released

Registers PINE and PINB are read-only!

Bits marked X can be 0 or 1 regardless of the joystick position

Initializing the joystick inputs

```
start:
        …
        in      r17, DDRE
        andi    r17, 0b11110011
        in      r16, PORTE
        ori     r16, 0b00001100
        out     DDRE, r17
        out     PORTE, r16
        ldi     r16, 0b00000000
        sts     DIDR1, r16
        in      r17, DDRB
        andi    r17, 0b00101111
        in      r16, PORTB
        ori     r16, 0b11010000
        out     DDRB, r17
        out     PORTB, r16
        …
```

Settings for port E

Settings for port B

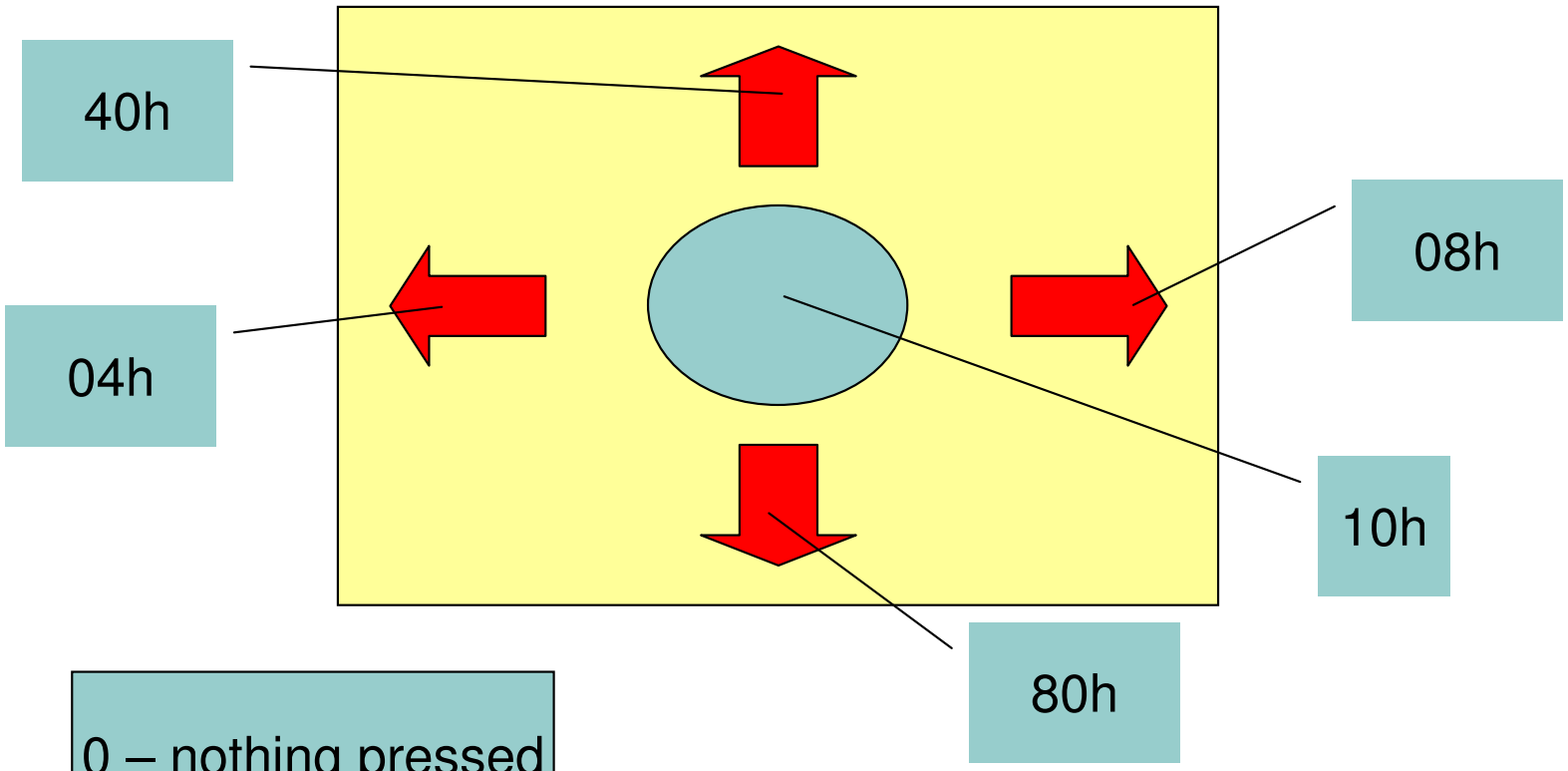
This code sets the joystick ports to inputs with pull-up resistors (port B, E) and disables analog inputs at bits 2 and 3 of port E (DIDR1).

Recommended algorithm for reliable evaluation of joystick state

1. Set the ports as recommended.
2. Read the PINE and PINB registers and convert their values into a single 8-bit register (e.g. r16).
3. Wait a few milliseconds.
4. Read the registers PINE and PINB and convert their values into a single 8-bit register (e.g. r17).
5. Compare the contents of r16 and r17.
6. If the register contents in step 5 differ, go to step 2. Otherwise, continue with step 7.
7. …

Note: do not use interrupts for the joystick task

Task: Write a program that displays joystick state as a number.



To obtain the codes: mask, invert and unite bits controlled by the joystick (that is, selected bits from registers PINE and PINB).