Fundusze Europejskie
Wiedza Edukacja Rozwój

Rzeczpospolita Polska

Unia Europejska
Europejski Fundusz Społeczny

Politechnika Śląska jako Centrum Nowoczesnego Kształcenia
opartego o badania i innowacje

POWR.03.05.00-IP.08-00-PZ1/17

Projekt współfinansowany przez Unię Europejską ze środków Europejskiego Funduszu Społecznego

# Microprocessor and Embedded Systems

**Faculty of Automatic Control, Electronics and Computer Science, Informatics, Bachelor Degree**

# Lecture 14

## AVR-family single-chip microcomputers
## Part 2
## Built-in peripherials

Bartłomiej Zieliński, PhD, DSc

# AVR (2)

Program:

(last week)

- AVR families & structure
- I/O ports

(today)

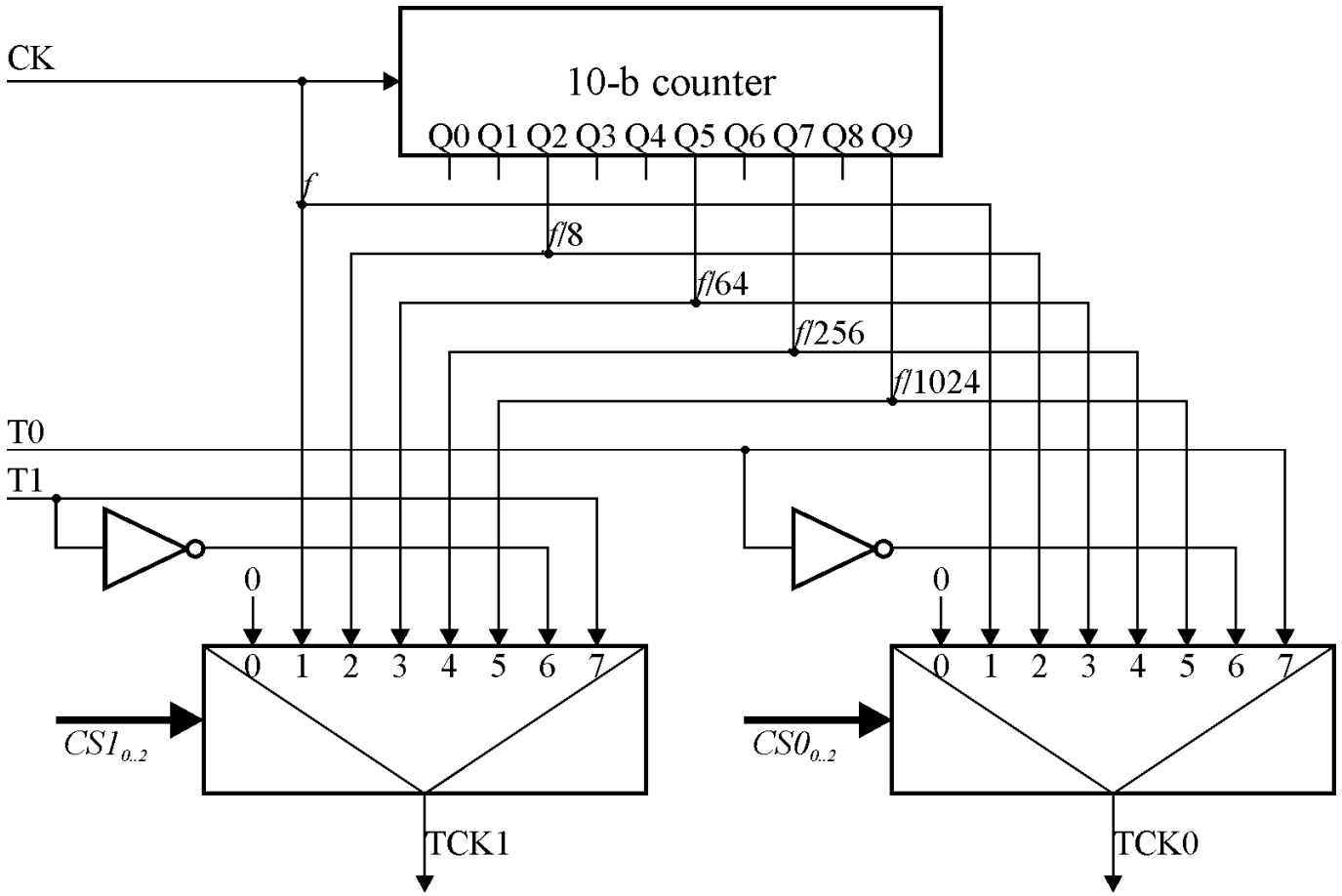
- Built-in peripherials
  - Timers/counters
  - Analogue circuits
  - Serial port
  - Interrupts
- Commands

# AVR (2)

- Timers/counters
  - TC0 (8-b), TC1 (16-b)
  - Common 10-b prescaler
  - Registers:
    - TIMSK – interrupt mask (common TC1&TC2)
    - TIFR – interrupt flags (common TC1&TC2)
    - TCNT0 – TC0 work register
    - TCNT1 – TC1 work register
    - ICR1 – TC1 capture?
    - TCCR1A, TCCR1B – TC1 control register

# AVR (2)

- Timers/counters
  - Common prescaler for TC0, TC1

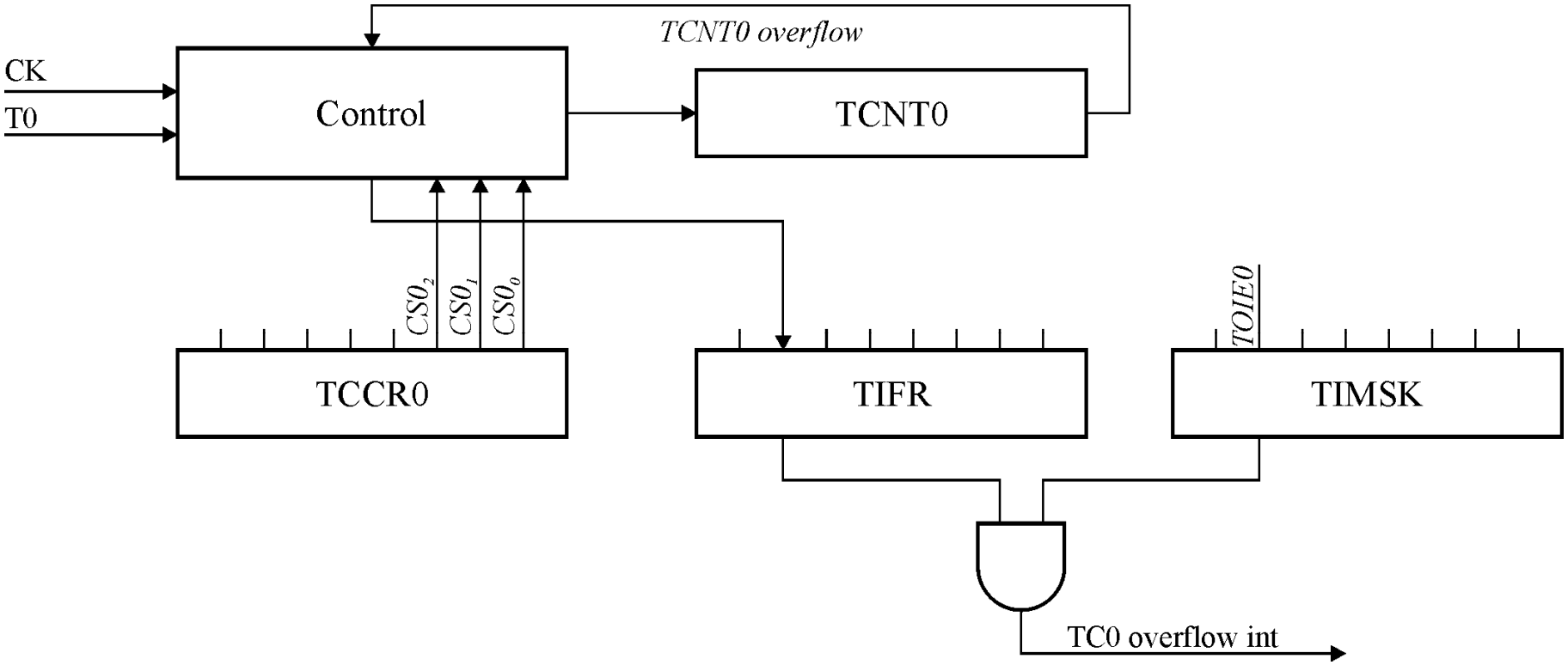


# AVR (2)

- Timers/counters
  - TC0
    - T0 input sychronised with clock
      - $f_{TC0} \leq \frac{f_{XTAL}}{2}$
    - T0 as input no matter if the pin is in or out
      - Software control of the counting proces
    - Counts up
      - 0 = overflow
    - TCNT0 rd/wr
      - write during clk pulse:
        - » 1. TCNT0 increment
        - » 2. write new value

# AVR (2)

- Timers/counters
  - TC0

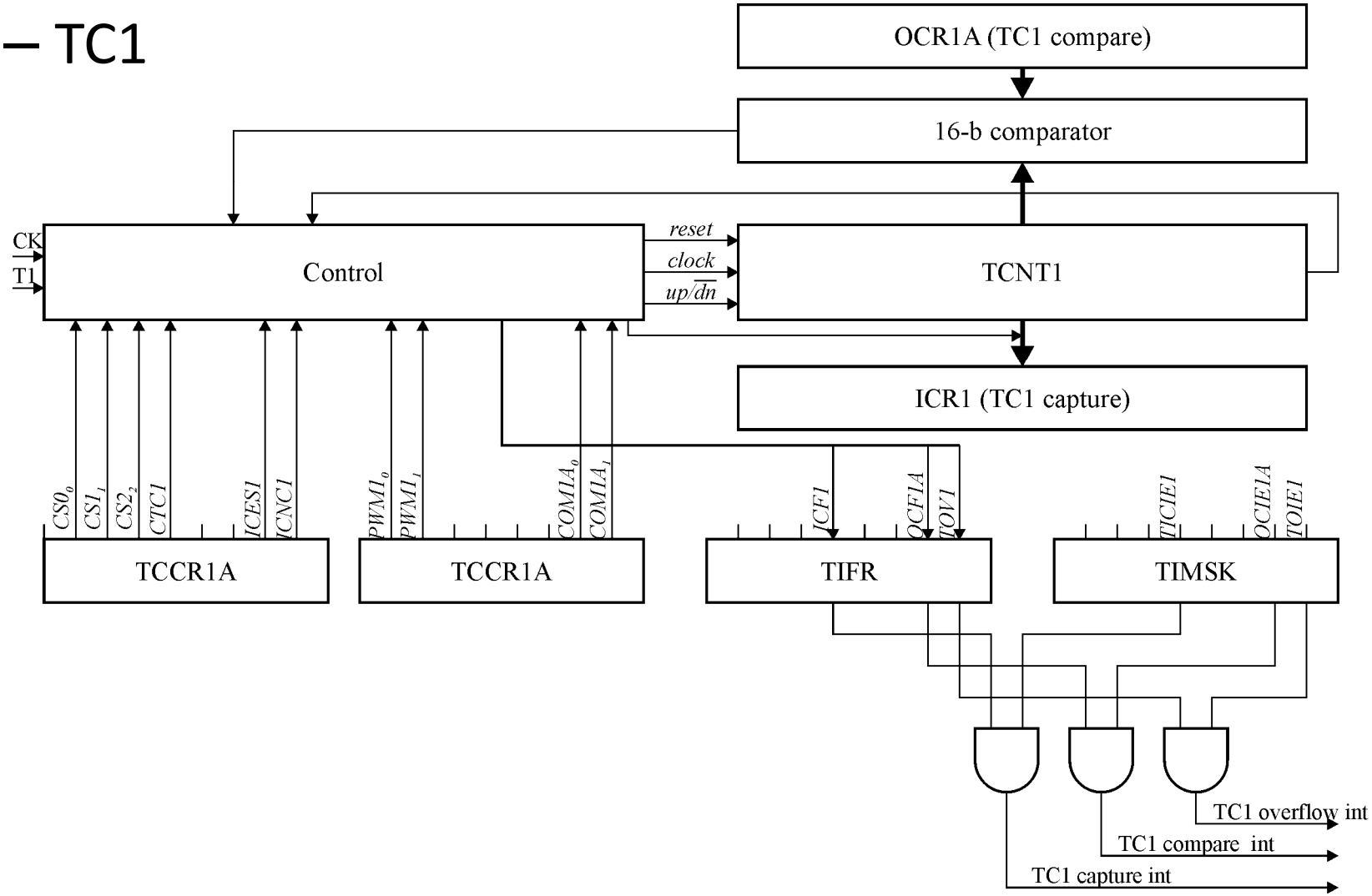


# AVR (2)

- Timers/counters
  - TC1 (as in AT90S2313)
    - T0 input sychronised with clock
      - $f_{TC0} \leq \frac{f_{XTAL}}{2}$
    - Modes:
      - Timer
      - Counter
      - Output compare
      - Input capture
      - PWM

# AVR (2)

- Timers/counters
  - TC1



# AVR (2)

- Timers/counters
  - TC1 registers
    - TCCR1A
      - COM1A1, COM1A0
        - » OC1 output disconnected from TC1 circuit
        - » Positive compare → OC1 change
        - » Positive compare → OC1=0
        - » Positive compare → OC1=1
      - PWM11, PWM10
        - » PWM for TC1 off
        - » TC1 as 8-b PWM
        - » TC1 as 9-b PWM
        - » TC1 as 10-b PWM

# AVR (2)

- Timers/counters
  - TC1 registers
    - TCCR1B
      - ICNC1 – noise canceller for TC1
        - » Off→trigger on 1st ICP input active edge
        - » On→trigger if 4 consecutive, equal samples found on ICP
        - » Usable e.g. if works with analogue comparator
      - ICES1 – TCNT1→ICR1 on rising (1)/falling (0) ICP edge
      - CTC1 – 1→TCNT1=0 after compare condition fulfilled
        - » Depends on prescaler
        - » No meaning in PWM mode
      - $CS1_{2..0}$ – clock source selection

# AVR (2)

- Timers/counters
  - TC1 registers
    - TCNT1H, TCNT1L (+ TEMP)
      - Write:
        - » Write TCNT1H → write TEMP
        - » Write TCNT1L → TCNT1H=TEMP
        - » Write order: TCNT1H, TCNT1L
      - Read:
        - » Read TCNT1L → TEMP=TCNT1H
        - » Read TCNT1H → read TEMP
        - » Read order: TCNT1L, TCNT1H
      - If TEMP used in interrupts
        - » Interrupts should be masked during rd/wr operations

# AVR (2)

- Timers/counters
  - TC1 compare mode
    - OCR1A=TCNT1 continuously checked
    - If equal, action on OC1 output as defined in TCCR1A
      - TIFR.OCF1A=1 (interrupt flag)
      - CTC1=1→TCNT1=0
      - OC1 must be set as output!
    - Read/write OCR1A as TCCR1
  - TC1 capture mode
    - ICP input active edge (TCCR1B.ICES1)→ICR1A=TCNT1
      - ICF1=1
    - Read/write ICR1A as TCCR1

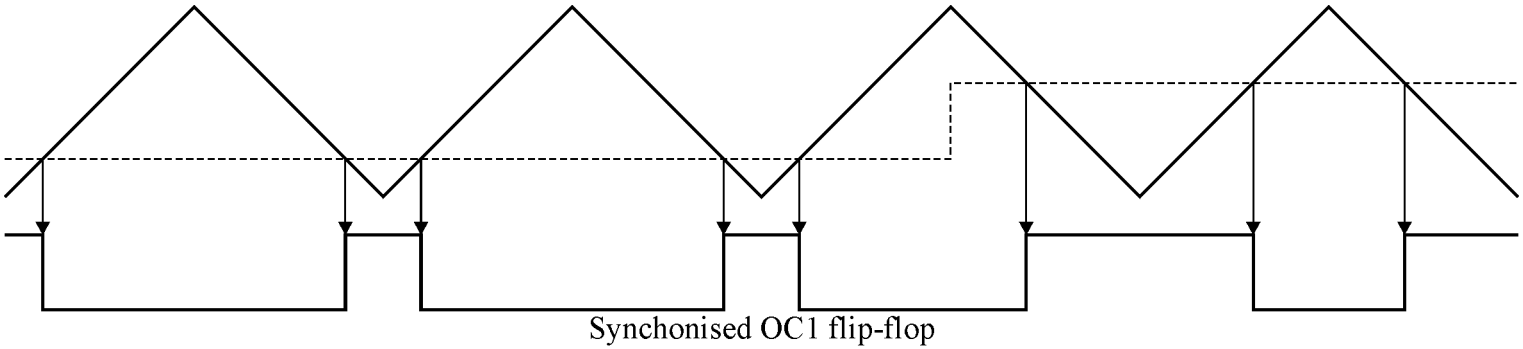
# AVR (2)

- Timers/counters
  - TC1 PWM mode
    - OCR1 register used
    - OC1=Modulator output
    - Duty cycle change without a glitch
    - TC1 cycle: 0→max, max→0
      - If TCNT1=OCR1A, OC1 set/cleared
        - » As defined in TCCR1A.COM1A$_{1..0}$ – PWM/reverse PWM

| PWM resolution | Max value | PWM frequency |
|---|---|---|
| 8 bits | $00FF (255) | $f_{TC1}/510$ |
| 9 bits | $01FF (511) | $f_{TC1}/1022$ |
| 10 bits | $03FF (1023) | $f_{TC1}/2046$ |

# AVR (2)

- Timers/counters
  - TC1 PWM mode
    - OCR1A effective write when TCNT1=max
      - No glitch pulses



# AVR (2)

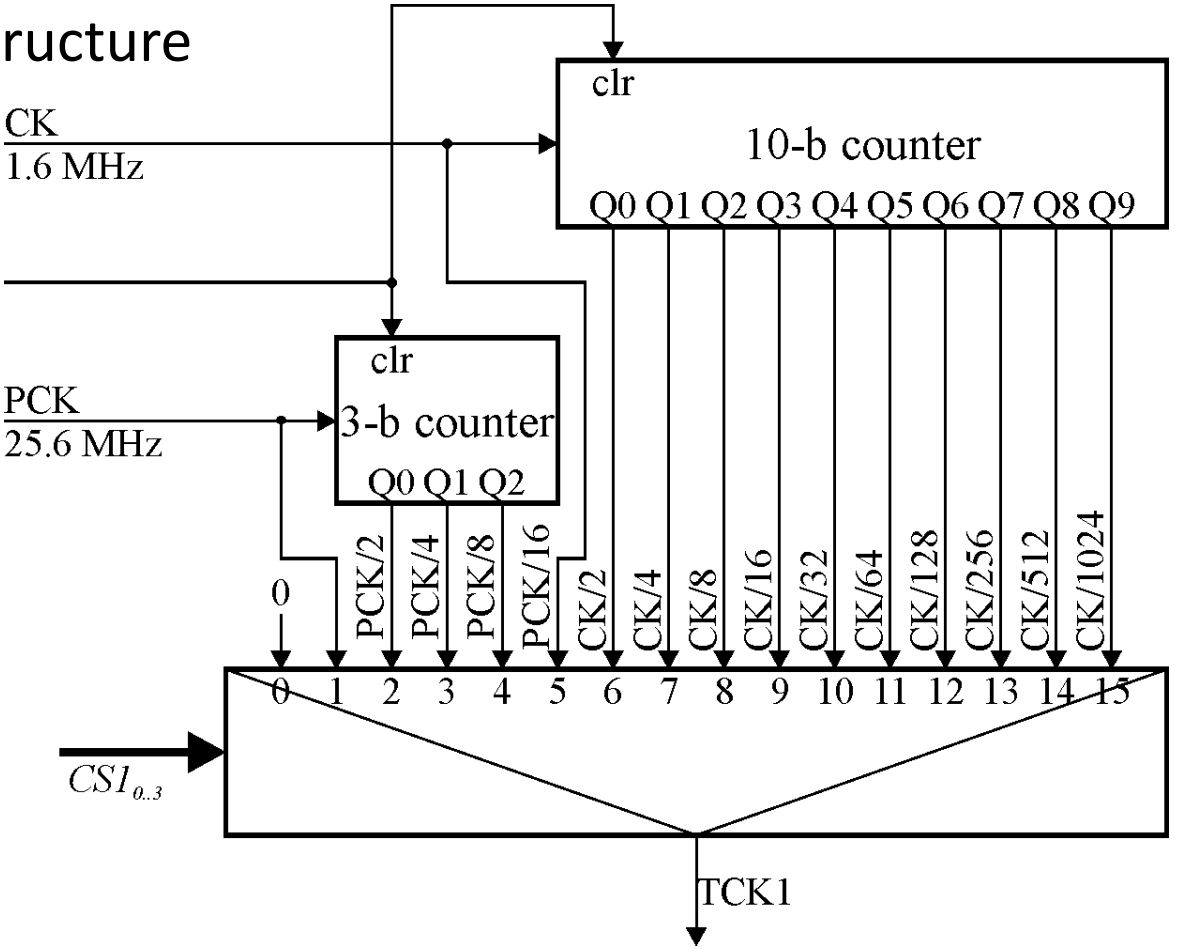
- Timers/counters
  - TC1 PWM mode
    - If OCR1A=max
      - No prescaler used → no PWM output
        - » Compared values for „up” and „down” count occur at the same time
      - Prescaler used → single pulse
        - » PWM=1 when max value reached
        - » Condition not interpreted when counting down
    - Interrupts in PWM mode
      - May work normally (TIMSK.TOIE1=1, SREG.I=1)
      - TOV=1 when TCNT1=$0000 when counting up
        - » Except 1st cycle after on

# AVR (2)

- Timers/counters
  - TC1 in ATtiny15
    - Different prescaler
    - Different structure
    - CK=internal oscillator, F≈1.6 MHz
    - PCK=PLL frequency multiplier, F≈25.6 MHz
      - Frequency stability depends on CK calibration
      - TC1 will not work for PCK, if $F_{CK}$>1.75 MHz
    - Both prescalers can be cleared (SFIOR.PSR1)
    - Two OCR1 registers
      - Duty cycle control
      - Frequency control

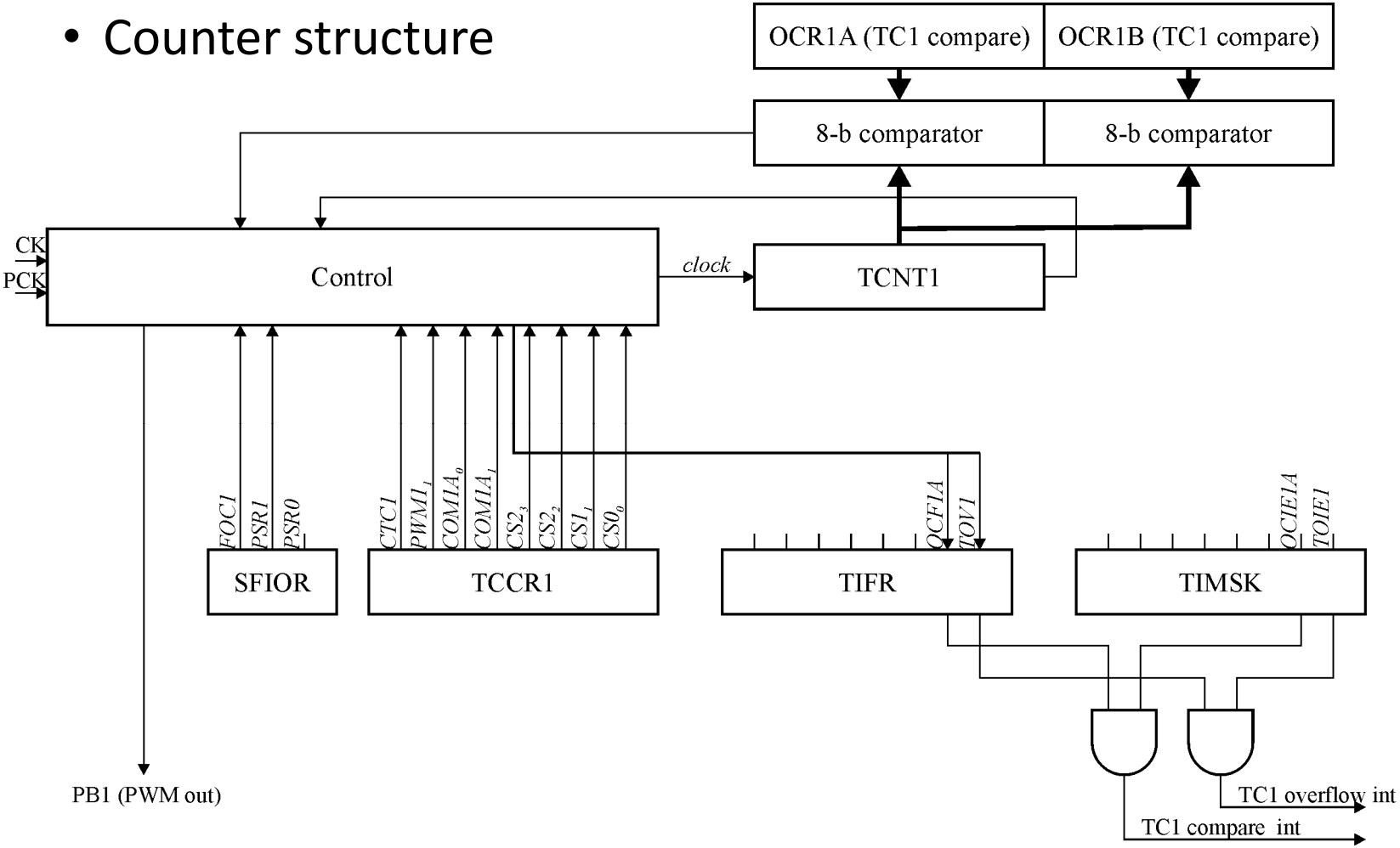
# AVR (2)

- Timers/counters
  - TC1 in ATtiny15
    - Prescaler structure



# AVR (2)

- Timers/counters
  - TC1 in ATtiny15
    - Counter structure



# AVR (2)

- UART
  - Built-in baud rate generator
  - High transmission rates for low quartz frequencies
  - 8/9-bits data word length
  - Noise canceller
  - Receiver buffer overflow detection
  - Frame error detection
  - Start bit error detection
  - 3 interrupts:
    - Transmission end
    - Transmitter buffer empty
    - Receiver buffer full

# AVR (2)

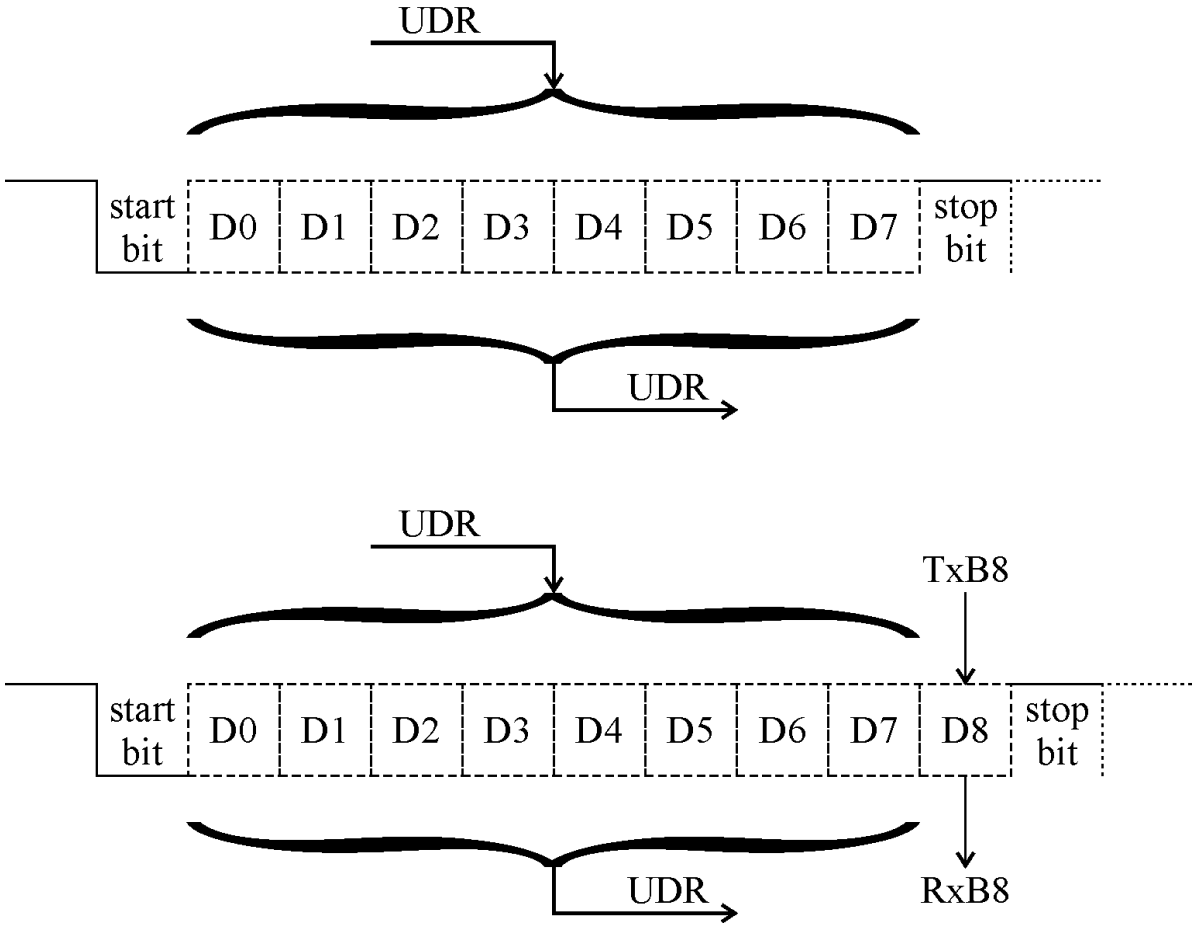
- UART
  - Registers
    - UDR (*UART data register*)
      - Rd/wr separate registers
    - USR (*UART status register*)
      - RxC – receive complete (receiver ready)
      - TxC – transmit complete (transmitter ready)
      - UDRE – UDR empty
      - FE – frame error (e.g., stop bit=0)
      - OR – overrun (receiver buffer overwritten)

# AVR (2)

- UART
  - Registers
    - UCR (*UART control register*)
      - RxCIE – RxC interrupt enable
      - TxCIE – TxC interrupt enable
      - UDRIE – UDRE interrupt enable
      - RxEN – receiver enable
      - TxEN – transmitter enable
      - CHR9 – 8/9-b data word length
      - RxB8 – received bit 8
      - TxB8 – transmitted bit 8

# AVR (2)

- UART
  - Data format



# AVR (2)

- UART
  - Registers
    - UBRR (UART Baud Rate Register)
      - $R = \frac{f_{CK}}{16(UBRR+1)}$
    - Recommended clock frequencies
      - 11.059, 7.3728, (4.608), 3.6864, 1.8432 MHz
        - » Most popular data rates exactly generated

# AVR (2)

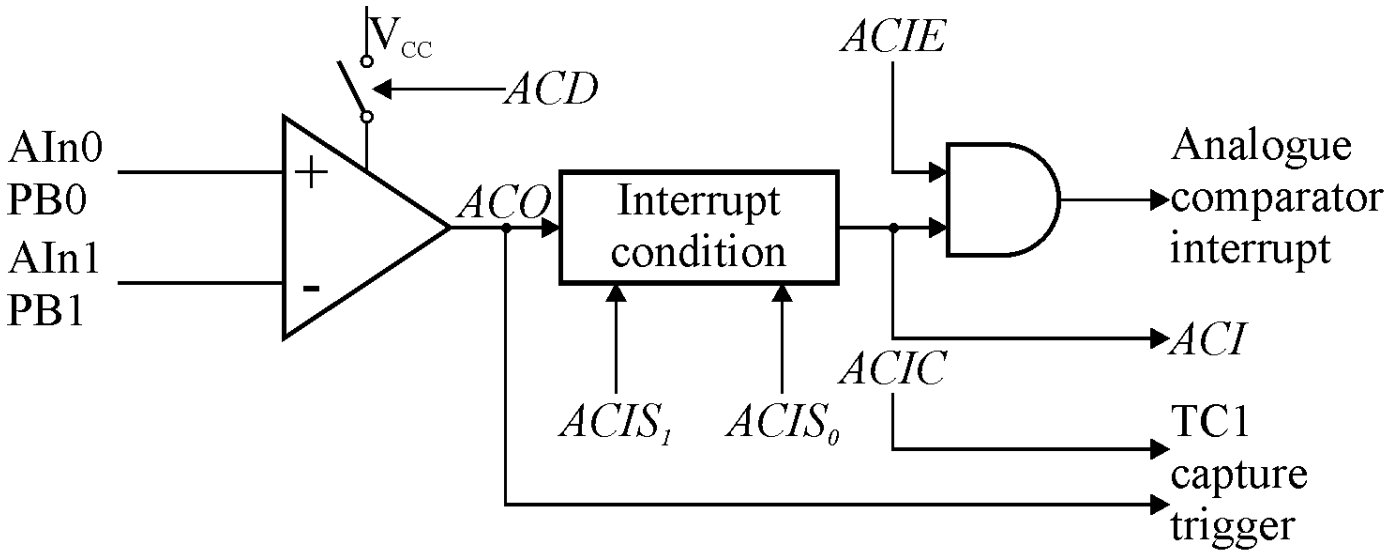
- UART
  - Receiver noise canceller
    - 16 samples per bit
    - Start bit
      - RxD=1→0 → start bit
      - Compare samples 8, 9 and 10
      - If at least 2 samples are „1”, start bit cancelled
    - Other bits
      - Compare samples 8, 9 and 10
      - Bit value = majority of samples
        - » Not necesserily consecutive
        - » E.g., 101 → 1

# AVR (2)

- Analogue comparator
  - Much cheaper than a full ADC
    - Analogue and digital parts interfere each other
    - IC design becomes complex
  - In AT90S2313
    - ACO=1, if $U_{AIN0} > U_{AIN1}$
    - ACO can trigger TC1 capture
    - Can generate an interrupt
      - ACO=1, ACO=0, ACO=NOT ACO
    - Can be software-turned off

# AVR (2)

- Analogue comparator



# AVR (2)

- Analogue comparator
  - ACSR – control/status register
    - ACD – 1→analogue comparator disable
      - ACSR.ACIE should be 0 to prevent unintended interrupt
    - ACO – analogue comparator output (copy of ACO out)
    - ACI – comparator analogue interrupt flag
    - ACIE – comparator analogue interrupt enable
    - ACIC – TC1 capture triggerred by analogue comparator
    - $ACIS_{1..0}$ – interrupt request condition
      - ACO=NOT ACO
      - ACO falling edge
      - ACO rising edge

# AVR (2)

- Interrupts
  - 10 Sources
    - External
      - INT1, INT0
    - Internal – timers/counters
      - TC0 overflow
      - TC1 capture
      - TC1 overflow
      - TC1 compare
    - Internal – UART
      - RxC
      - TxC
      - UDRE
    - Internal – analogue comparator

# AVR (2)

- Interrupts
  - External interrupts
    - Flags software unavailable
      - Set automatically
      - Cleared when procedure starts
    - Software interrupt call
      - PD2 (INT0), PD3 (INT1) configured as outputs

# AVR (2)

- Interrupts
  - Registers – external interrupts
    - SREG.I
      - Global interrupt enable
    - GIMSK (General Interrupt Mask Register)
      - INT1 – external int 1 enable
      - INT0 – external int 0 enable
    - GIFR (General Interrupt Flag Register)
      - INTF1 – external int 1 flag
      - INTF0 – external int  0 flag
    - MCUCR
      - How INT0, INT1 external interrupts are signalled
        - » Low level/rising edge/falling edge

# AVR (2)

- Interrupts
  - Registers – T/C's
    - TIMSK (T/C Interrupt Mask Register)
      - TOIE1 – TC1 Overflow Int Enable
      - OCIE1A – TC1 Output Compare Match Int Enable
      - TICIE1 – TC1 Input Capture Int Enable
      - TOIE0 – TC0 Overflow Int Enable
    - TIFR (T/C Interrupt Flag Register)
      - TOV1 – TC1 Overflow Flag
      - OCF1A – TC1 Output Compare Flag
      - ICF1 – TC1 Input Capture Flag
      - TOV0 – TC0 Overflow Flag

# AVR (2)

- Interrupts
  - Registers – UART
    - USR (*UART status register*)
      - RxC – receive complete (receiver ready)
      - TxC – transmit complete (transmitter ready)
      - UDRE – UDR empty
    - UCR (*UART control register*)
      - RxCIE – RxC interrupt enable
      - TxCIE – TxC interrupt enable
      - UDRIE – UDRE interrupt enable
  - Registers – analogue comparator
    - ACSR
      - ACI – comparator analogue interrupt flag
      - ACIE – comparator analogue interrupt enable

# AVR (2)

- Interrupts
  - Interrupt acceptance (usually 8 clock cycles)
    - Interrupt flag set (then 4 clock cycles)
    - SREG.I=0
    - Return address → stack (2 clock cycles)
    - Call an interrupt service procedure (2 clock cycles)
    - Interrupt flag cleared
  - Multilevel interrupt system
    - SREG.I=1 in an interrupt service procedure
  - RETI (usually 4 clock cycles)
    - Return address taken off the stack (2 clock cycles)
    - SREG.I=1

# AVR (2)

- Interrupts
  - Interrupt vectors

| Number | Address | Source |
|---|---|---|
| 1 | $0000 | Reset |
| 2 | $0001 | INT0 |
| 3 | $0002 | INT1 |
| 4 | $0003 | TC1 capture |
| 5 | $0004 | TC1 compare |
| 6 | $0005 | TC1 overflow |
| 7 | $0006 | TC0 overflow |
| 8 | $0007 | UART Rx |
| 9 | $0008 | UART UDRE |
| 10 | $0009 | UART Tx |
| 11 | $000A | Analogue comparator |

# AVR (2)

- Interrupts
  - Interrupt vectors – example program fragment

```
$0000     rjmp Reset
$0001     rjmp ExtInt0
$0002     rjmp ExtInt1
$0003     rjmp TC1_Capt
$0004     rjmp TC1_Comp
$0005     rjmp TC1_Ovl
```

# AVR (2)

- Low power modes
  - *Idle* mode
    - CPU stops
    - Peripherials operating
    - Interrupt → wake, CPU starts immediately
  - *Power Down* mode
    - CPU stops
    - Peripherials stop
    - Watchdog, external interrupts operating (if enabled)
    - Reset, watchdog, external interrupt (level-triggerred)→ wake

# AVR (2)

- Low power modes
  - MCUCR register
    - SE – 1=Sleep Enable (sleep after `sleep` command)
      - Should be enabled immediately before `sleep` command
    - SM – Sleep Mode:
      - 0: Idle
      - 1: Power Down

# AVR (2)

- Commands
  - Arithmetical/logical commands
    - Addition/substraction
      - Add; Rd+=Rs
      - Adc; Rd+=Rs+C
      - Adiw; RRh:RRl+=c63 (16-b + constant)
      - Sub; Rd-=Rs
      - Subi; Rh-=c255
      - Sbiw; RRh:RRl-=c63 (16-b + constant)
      - Sbc; Rd-=Rs-C
      - Sbci; Rd-=Rh-c255-C
      - Inc; Rd+=1
      - Dec; Rd-=1

# AVR (2)

- Commands
  - Arithmetical/logical commands
    - Logical commands
      - And; Rd&&=Rs
      - Andi; Rh&&=c255
      - Or; Rd||=Rs
      - Ori; Rh||=c255
      - Eor; Rd^=Rs
      - Com; Rd=$ff-Rd
      - Neg; Rd=$00-Rd
      - Sbr; Rh||=c255
      - Cbr; Rh&&=c255
      - Ser; Rh=$ff
      - Tst; Rd&&=Rd
      - Clr; Rd^=Rd

# AVR (2)

- Commands
  - Arithmetical/logical commands
    - Integer multiplication
      - Mul; (unsigned) R1:R0=Rd×Rs
      - Muls; (signed) R1:R0=Rhd×Rhs
      - Mulsu; (signed×unsigned) R1:R0=Rhd×Rhs
    - Fractional multiplication
      - Fmul; R1:R0=(Rd×Rs)<<1
      - Fmuls; R1:R0=(Rd×Rs)<<1
      - Fmulsu; R1:R0=(Rd×Rs)<<1

# AVR (2)

- Commands
  - Bit commands
    - Set/clear bit in IO port
      - Sbi; P(b)=1
      - Cbi; P(b)=0
    - Shift/rotate
      - Lsl, lsr; logical shift
      - Rol, ror; rotate with carry
      - Asr; arithmetical shift
    - others
      - Swap; $Rd_{4..7} \leftrightarrow Rd_{3..0}$
      - Bset, bclr; SREG(b)=1, 0
      - Bst; T=Rs(b)
      - Bld; Rd(b)=T

# AVR (2)

- Commands
  - Bit commands
    - Flags set/clear
      - sec, clc; C=1, 0
      - sen, cln; N=1, 0
      - Sez, clz; Z=1, 0
      - Sei, cli; I=1, 0
      - Ses, cls; S=1, 0
      - Sev, clv; V=1, 0
      - Set, clt; T=1, 0
      - Seh, clh; H=1, 0

# AVR (2)

- Commands
  - Data transfer
    - Register-register
      - Mov; Rd=Rs
      - Movw; Rd:Rd+1=Rs:Rs+1 (16-b copy)
    - I/O
      - In; Rd=P
      - Out; P=Rs
    - Stack
      - Push; (SPL)=Rs; SPL-=1
      - Pop; SPL+=1; Rd=(SPL)

# AVR (2)

- Commands
  - Data transfer
    - Load
      - Ldi; Rh=c255
      - Ld; intermediate address (+postincr, predecr)
      - Ldd; intermediate + displacement (c63)
      - Lds; direct address
    - Store
      - si; intermediate address (+postincr, predecr)
      - std; intermediate + displacement (c63)
      - sts; direct address

# AVR (2)

- Commands
  - Data transfer
    - Program memory transfer
      - Lpm; intermediate (+ postincr.)
        - » Rd=(Z), Rd=(Z++)
      - Elpm; extended intermediate
        - » Rd=(rampz:z), Rd=(rampz:z++)
      - Spm; intermediate
        - » (Z)=R1:R0

# AVR (2)

- Commands
  - Jumps/calls
    - Jumps
      - Rjmp; relative, PC+=c1024+1
      - Ijmp; intermediate, PC=Z
      - Eijmp; PC=EIND.Z
      - Jmp; PC=adr4M
    - Calls (as jumps)
      - Rcall
      - Icall
      - Eicall
      - Call
    - Returns
      - Ret
      - reti

# AVR (2)

- Commands
  - Jumps/calls
    - Compare register
      - Cpse; compare & skip on equal
      - Cp; compare registers
      - Cpc; compare registers with carry
      - Cpi; register and constant
    - Skip; conditionally skip next command
      - Sbrc; skip if register bit cleared
      - Sbrs; skip if register bit set
      - Sbic; skip if port bit cleared
      - Sbrc; skip if port bit set

# AVR (2)

- Commands
  - Jumps/calls
    - Conditional jumps
      - Brbs, brbc; if flag in SREG set/cleared
      - Breq, brne; if equal/not equal
      - Brcs, brcc; if carry set/cleared
      - brsh, brlo; if ≥/< (unsigned)
      - Brmi, brpl; if negative/positive
      - brge, brlt; if ≥/< (signed)
      - Brhs, brhc; if half carry set/cleared
      - Brts, btrs; if T set/cleared
      - Brvs, brvc; if V set/cleared
      - Brie, brid; if interrupts enabled/disabled

# AVR (2)

- Commands
  - CPU control
    - Sleep
    - Wdr; watchdog reset
    - Break; used by debuggers only
    - Nop;