

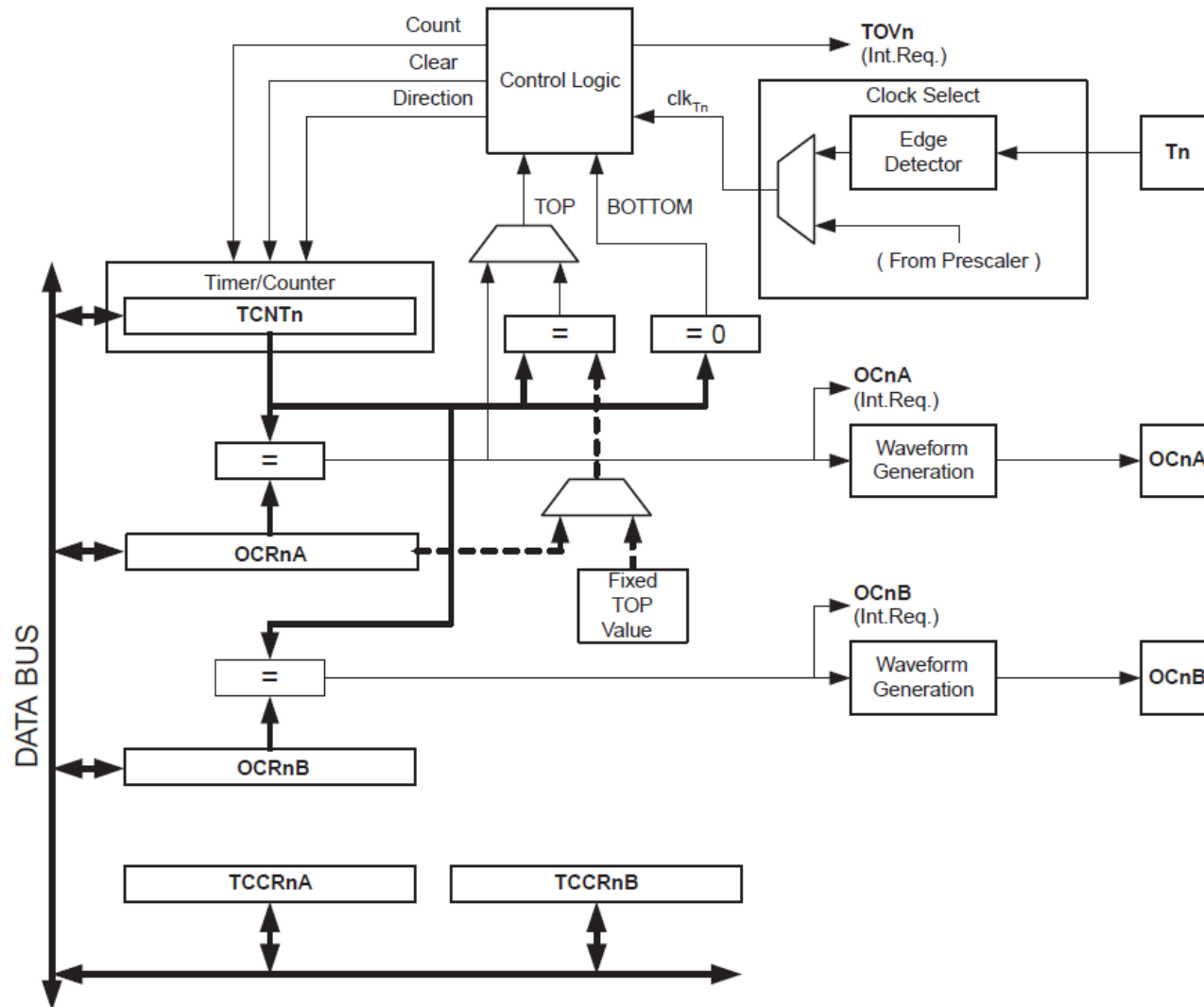
# Timers 0, 1 & 2

---

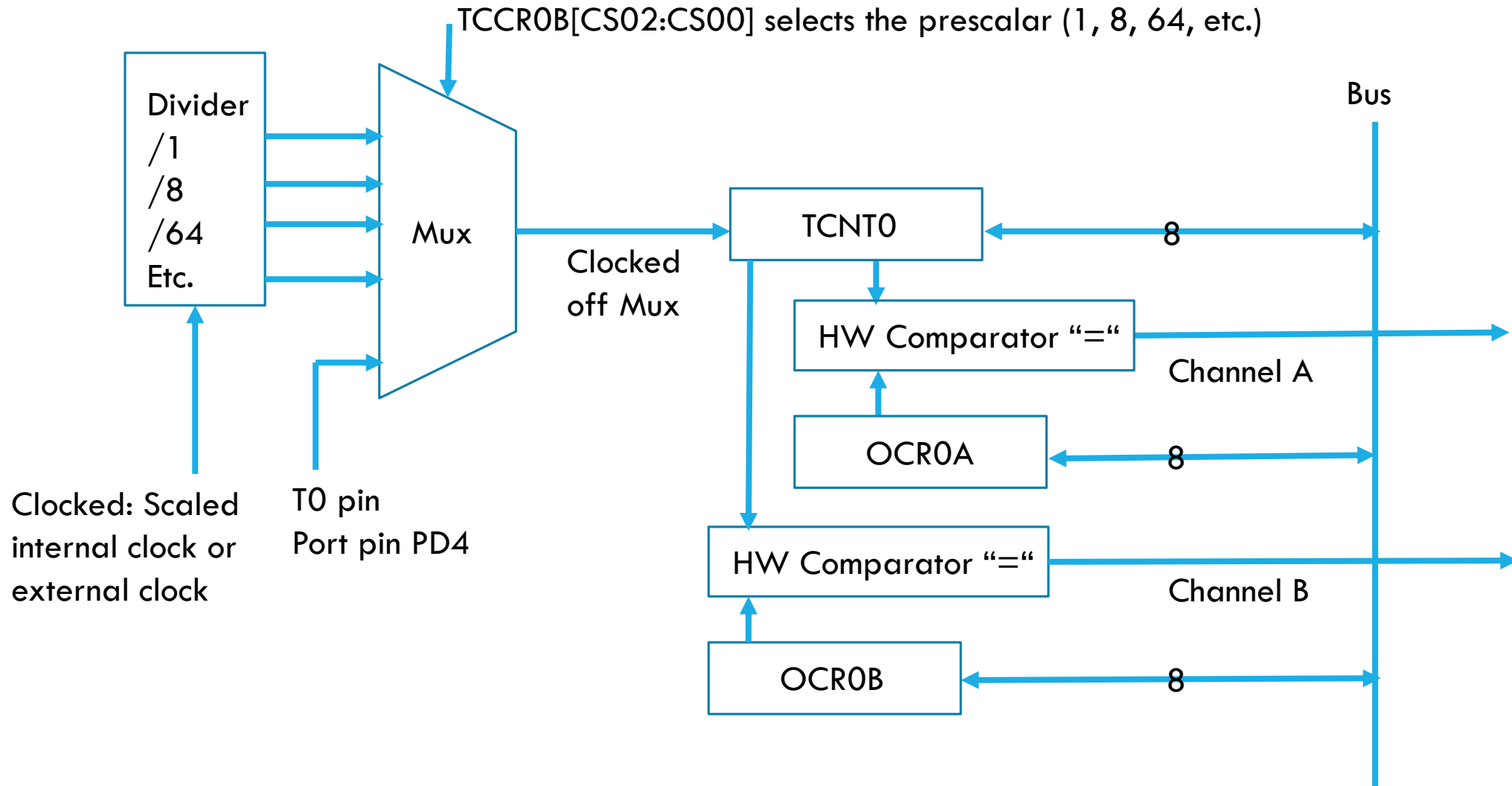
**Marten van Dijk, Syed Kamran Haider**  
Department of Electrical & Computer Engineering  
University of Connecticut  
Email: {vandijk, syed.haider}@engr.uconn.edu

# Timer 0 (Same as Timer 2)

Figure 14-1. 8-bit Timer/Counter Block Diagram



# Timer 0



# Putting It Together: Task Based Programming

```
....  
int TaskTime = 500;  
volatile int SWTaskTimer=TaskTime;  
  
ISR(TIMERO_COMPA_vect)  
{  
    if (SWTaskTimer>0) {SWTaskTimer--;}  
}  
  
// 1ms ISR for Timer 0 assuming F_CPU = 1MHz  
void InitTimer0(void)  
{  
    TCCR0A |= (1<<WGM01); //Clear on Compare A  
    OCROA = 124; //Set number of ticks for Compare A  
    TIMSK0 = 2; //Enable Timer 0 Compare A ISR  
    TCCR0B = 2; //Set Prescaler & Timer 0 starts  
}  
....
```

```
int main(void)  
{  
    ...  
    InitTimer0();  
    ...  
    sei(); // Enable global interrupt  
  
    while(1)  
    {  
        if (SWTaskTimer == 0)  
        {  
            Task();  
            SWTaskTimer == TaskTime;  
        }  
    }  
  
    return 0;  
}
```

# Example Timer 0

---

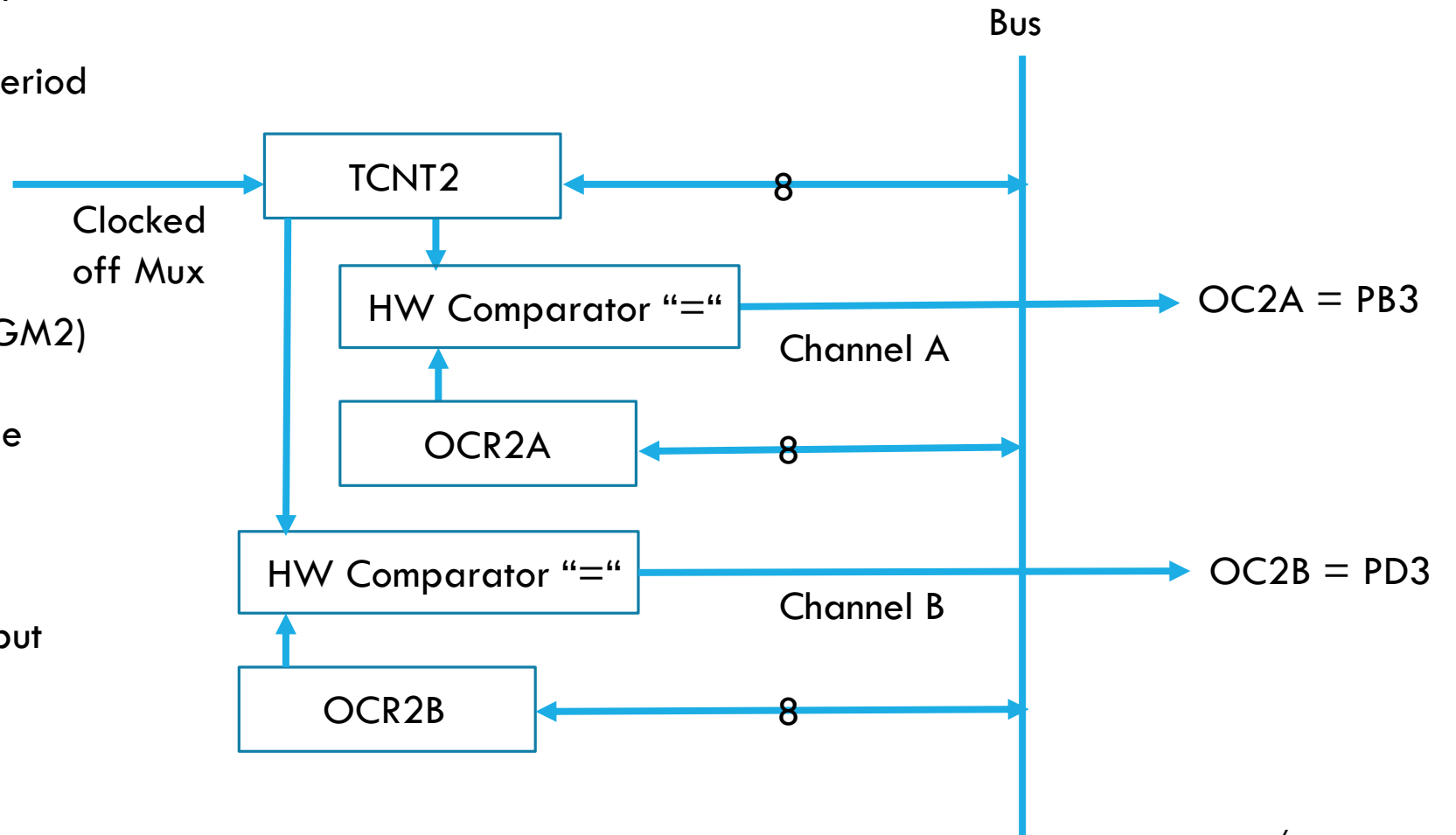
- 16MHz, 1ms ticks:

```
// 1ms ISR for Timer 0 assuming F_CPU = 16MHz
void InitTimer0(void)
{
    TCCR0A |= (1<<WGM01); //turn on clear-on-match with OCR0A
    OCR0A = 249;          //Set the compare register to 250 ticks
    TIMSK0 = (1<<OCIE0A); //Enable Timer 0 Compare A ISR
    TCCR0B = 3;           // Set Prescaler to divide by 64 & Timer 0 starts
}
....
```

# Timer 2

Create an autonomous 200 cycle long square wave at PB3:

- $OCR2A = 99$  sets a 100 cycle half period
- $TCCR2B = 1$  sets prescalar at 1, i.e., counting is done at full rate ( $F_{CPU}$ )
- $TCCR2A = (1 \ll \text{COM2A0}) | (1 \ll \text{WGM21});$ 
  - See Timer 0 discussion:  $(1 \ll \text{WGM2})$  gives us a clear on match
  - $(1 \ll \text{COM2A0})$  is new: it tells the MCU to use channel A, i.e., it connects the comparator to the output pin  $OC2A = PB3$
- $DDRB = (1 \ll \text{PINB3})$  sets PB3 as output



# Example Timer 2

---

- 200 cycle square waveform at PB3 (a first example of PWM):

```
void InitTimer2(void)
{
    TCCR2A |= (1<<COM2A0) | (1<<WGM21); //turn on clear-on-match with OCR0A
                                           // transmit comparator result to pin OC2A

    OCR0A = 99; //Set the compare register to 100 ticks, i.e., one half period
    TCCR2B = 1; // Set Prescalar to divide by 1, i.e., full speed
    DDRB = (1<<PINB3); //Set OC2A = PB3 to output
}
....
```



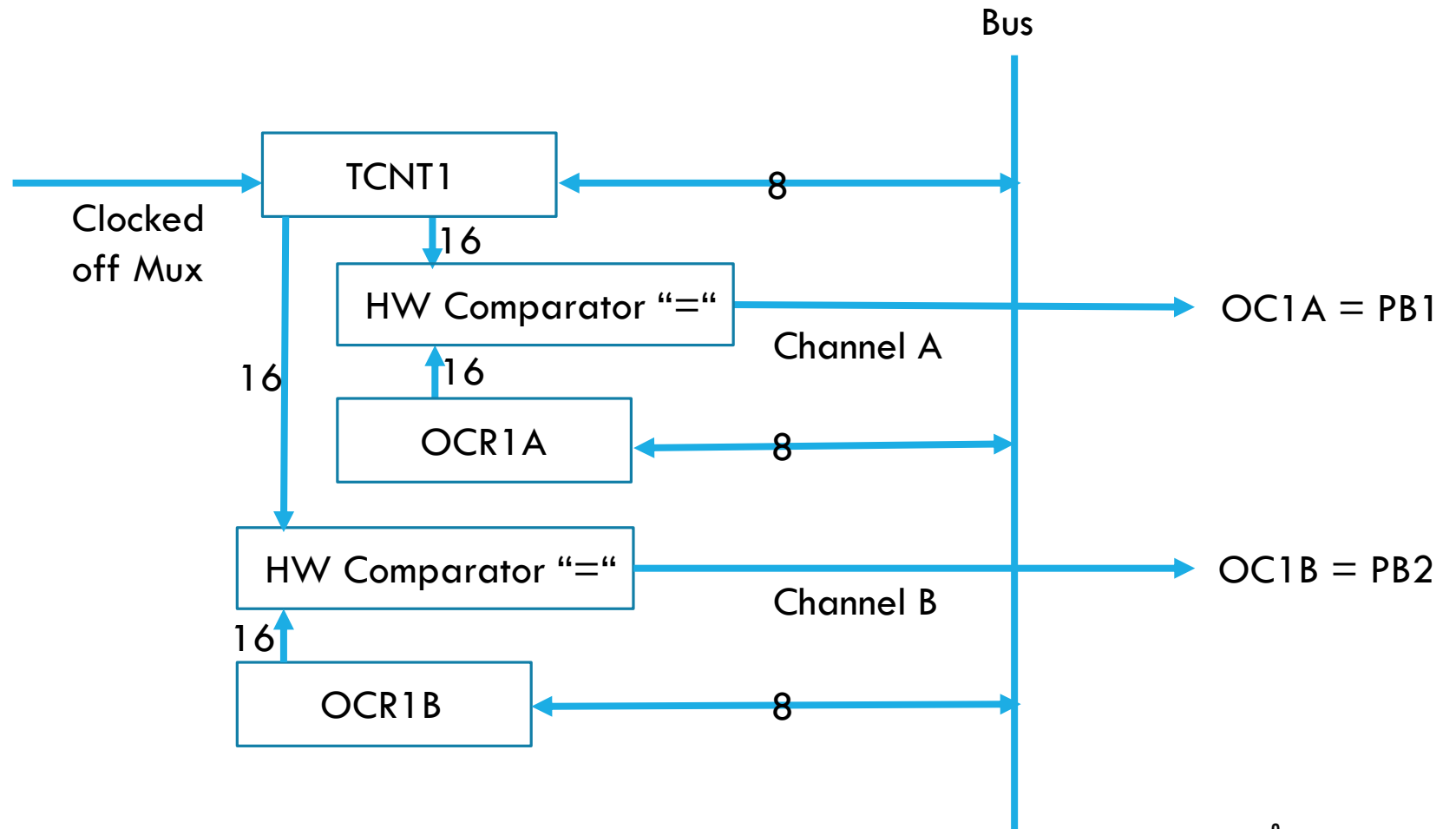


# Timer 1

TCNT1 is 16 bits, a high and low register:

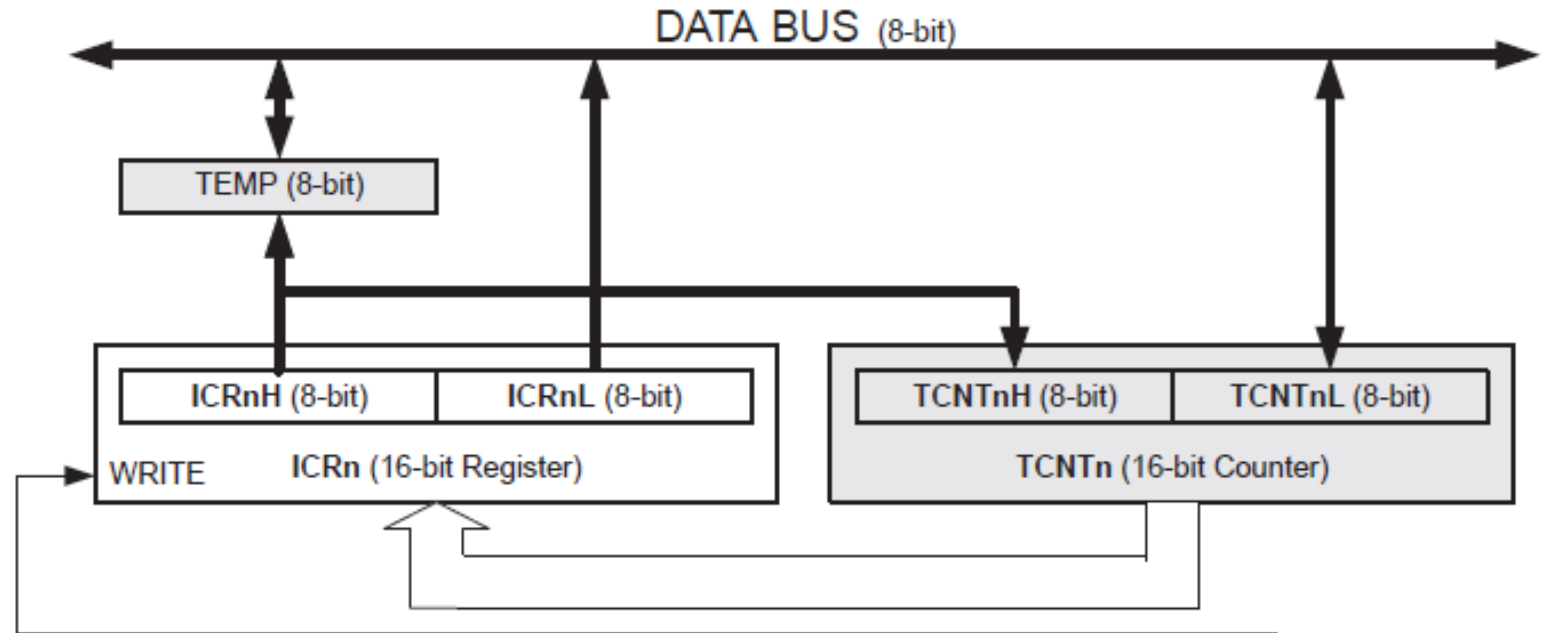
- Need 2 bus cycles to read TCNT1: it reads the lower byte and also copies the higher byte to a special location
- Stores the 8-bit high till you read it, if you do not read it, you will never read a value again
- If you read the high value first and then the low value, timer 1 is frozen
- Use the 16 bit reads built into C

Not drawn: TCNT1 can be captured by register ICR1 clocked of a mux .... see next slide



# TCNT1 Can Be Captured by ICR1

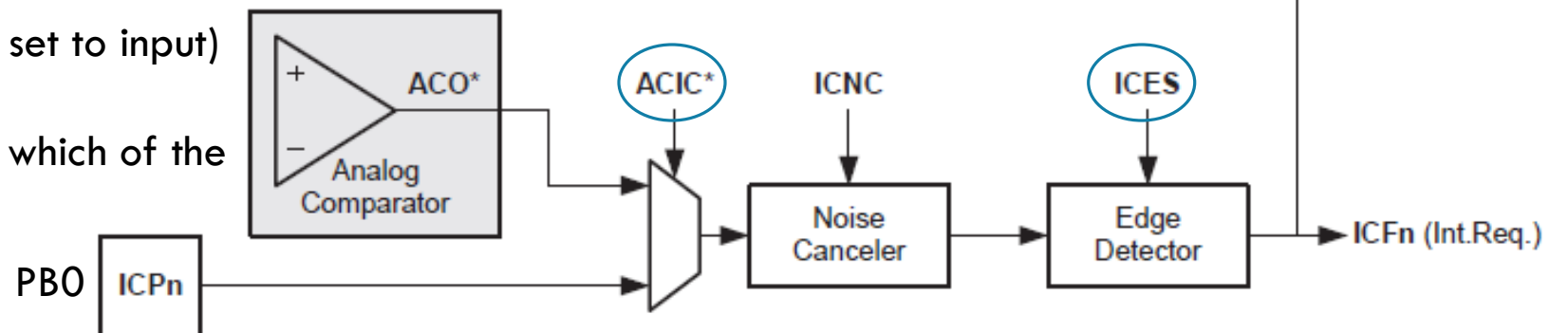
Figure 15-3. Input Capture Unit Block Diagram



Edge change indicates when to capture.

- Caused by an edge change on PBO (if set to input)
- **Analog Comparator**

Select bit ACIC in ACSR register indicates which of the two is chosen



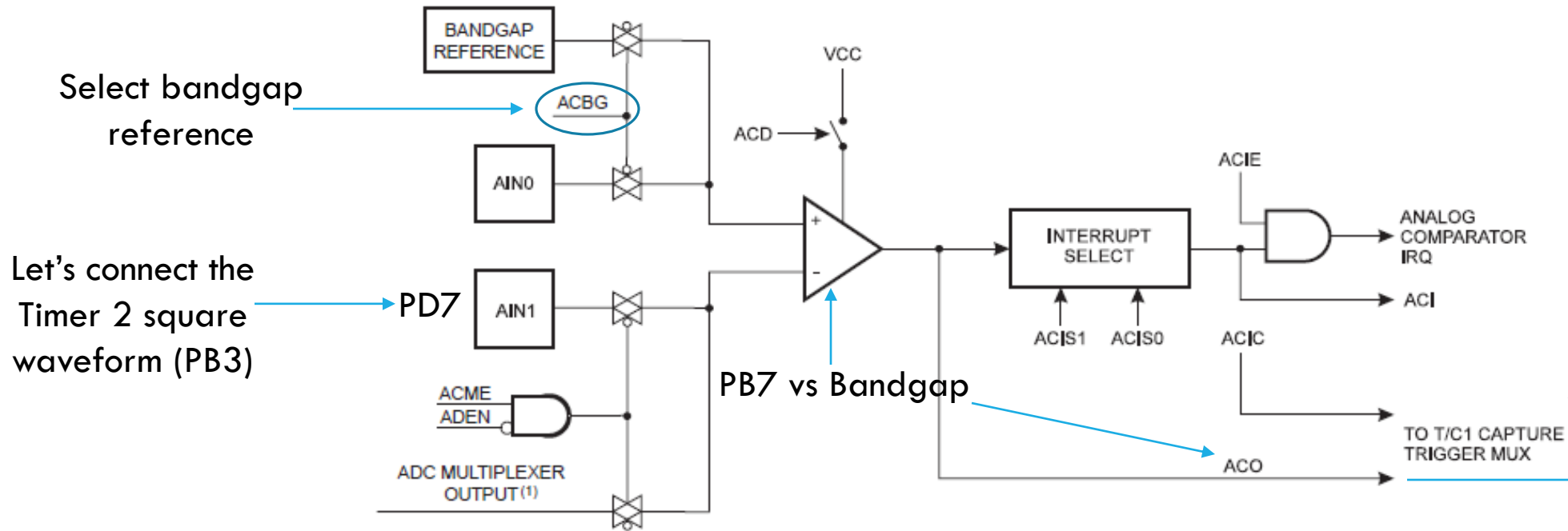
# Register Description Timer 1

---

- Control register TCCR1A:
  - Positions 7&6 → COM1A (COM1A0 and COM1A1)
  - Positions 5&4 → COM1B (COM1B0 and COM1B1)
  - Positions 1&0 → WGM11 and WGM10 (waveform)
- Control register TCCR1B:
  - Positions 3&4 → WGM13 and WGM12 (waveform continued)
  - Positions 0,1,2 → Prescalar as before
  - Position ICES1=6 → Sets input capture edge select:
    - 1 = rising
    - 0 = falling
  - Position ICNC1=7 → Sets input capture noise canceler
    - This requires 2 measurements in a row making sure one transmission actually occurred
- Control register TIMSK1:
  - Positions 0,1,2 → As before
    - TOIE1 (timer overflow interrupt enable)
    - OCIE1A and OCIE1B (on compare and match interrupt enable)
  - Positions ICIE1=5 → Interrupt capture interrupt enable

# Analog Comparator Output

Figure 22-1. Analog Comparator Block Diagram<sup>(2)</sup>



- Notes:
1. See Table 22-1 on page 247.
  2. Refer to Figure 1-1 on page 2 and Table 13-9 on page 88 for Analog Comparator pin placement.

Want to capture at rising edge ICES1 set to 1; Since PB3 generates a waveform on a cycle by cycle basis, prescaler set to 1 for full speed

# PIN Assignment

**Table 13-9.** Port D Pins Alternate Functions

Port Pin	Alternate Function
PD7	AIN1 (Analog Comparator Negative Input) PCINT23 (Pin Change Interrupt 23)
PD6	AIN0 (Analog Comparator Positive Input) OC0A (Timer/Counter0 Output Compare Match A Output) PCINT22 (Pin Change Interrupt 22)
PD5	T1 (Timer/Counter 1 External Counter Input) OC0B (Timer/Counter0 Output Compare Match B Output) PCINT21 (Pin Change Interrupt 21)
PD4	XCK (USART External Clock Input/Output) T0 (Timer/Counter 0 External Counter Input) PCINT20 (Pin Change Interrupt 20)
PD3	INT1 (External Interrupt 1 Input) OC2B (Timer/Counter2 Output Compare Match B Output) PCINT19 (Pin Change Interrupt 19)
PD2	INT0 (External Interrupt 0 Input) PCINT18 (Pin Change Interrupt 18)
PD1	TXD (USART Output Pin) PCINT17 (Pin Change Interrupt 17)
PD0	RXD (USART Input Pin) PCINT16 (Pin Change Interrupt 16)

# Register Description Timer 1

## 22.3.2 ACSR – Analog Comparator Control and Status Register

Bit	7	6	5	4	3	2	1	0	
0x30 (0x50)	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	ACSR
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	N/A	0	0	0	0	0	

- Let's program a capture interrupt using the analog comparator
- Need to set register ACSR:
  - Positions 0&1 → Interrupt on toggle rise or fall
  - Positions 2&3 → Capture and Comparator interrupt enable
    - We want to enable the capture interrupt (not the comparator interrupt)
  - Position 4 → Comparator interrupt flag:
    - Is set when this interrupt happens, and
    - clears when a corresponding ISR executes the final (atomic) RETI instruction
  - Position 5 → Records ACO raw comparator output:
    - in real time nanosec by nanosec
    - digital output of the analog comparator (signal ACO)
  - Position 6 → Connects positive input to a bandgap reference (a temperature independent voltage reference circuit)
    - If 1, then (see description datasheet) fixed bandgap reference is used as input to the analog comparator (usually do not want this)
  - Position 7 → When switched to 1 the analog comparator is turned off

# Example Timer 1

```
void InitTimer1(void)
{
    //Set up timer1 for full speed and capture an edge on analog comparator pin D.7

    //Set capture to positive edge; Full counting rate (prescaler set to 1)
    TCCR1B = (1<<ICES1) + 1;

    // Turn on timer1 interrupt-on-capture
    TIMSK1 = (1<<ICIE1);

    // Set analog comp to connect to timer capture input and turn on the band gap reference on the positive input
    ACSR = (1<<ACBG) | (1<<ACIC);
    // Comparator negative input is AIN1 = D.7
    DDRD = 0;

}
....
```

# Full Picture Timers

---

- 3 initialization codes for Timer 0, 1, 2; Timer 0 implements a 1ms software counter
- Timer 2 (at full speed) generates a square waveform, period 200 cycles
- Waveform drives ACO
- Rising edges ACO causes capture interrupts for Timer 1 (at full speed)
- Both timers run at full speed and should be synchronized:

```
ISR (TIMER1_CAPT_vect)
{
    // read timer1 input capture register
    T1capture = ICR1 ;
    // compute time between captures
    period = T1capture - lastT1capture;
    lastT1capture = T1capture ;
}
```

```
ISR (TIMER0_COMPA_vect)
{
    //Decrement the time if not already zero
    if (time1 > 0) --time1;
}
```



# Full Picture Timers

```
int main(void)
{
  initialize();
  while(1)
  {
    // task1 prints the period
    if (time1 == 0){time1 = t1;    task1 ();}

    // poll for ACO 0->1 transition (ACSR.5)
    // as fast as possible and record Timer1
    ACObit = ACSR & (1 << ACO) ;
    if ((ACObit != 0) && (lastACObit == 0))
    {
      T1poll = TCNT1 ;
      periodPoll = T1poll - lastT1poll;
      lastT1poll = T1poll ;
    }
    lastACObit = ACObit ;
  }
}
```

Connect PD3 and PB7 !

Print the polled period from ACSR.5 which records ACO  
Print the polled captured period from ICR1 (done in capture ISR)

What is the difference?

Measurement from polled period is off by 10% → Next lab

# Labs 3b & 3c

---

- We will remove `delay_ms()` from the LCD goto and write data commands
- The assumption is that the task that calls these commands
  - is issued every  $x$  ms with  $x$  much larger than
  - the combined waiting time over all `delay_ms` in the LCD commands within the task.
- This implies that this task will not be called while LCD commands are being executed, hence, no multi-threading and our simple solution (without priority queues etc.) should work
- By making the LCD commands non-blocking, other tasks in the main while loop continue without interruption! In a future lab we plan to demonstrate this.