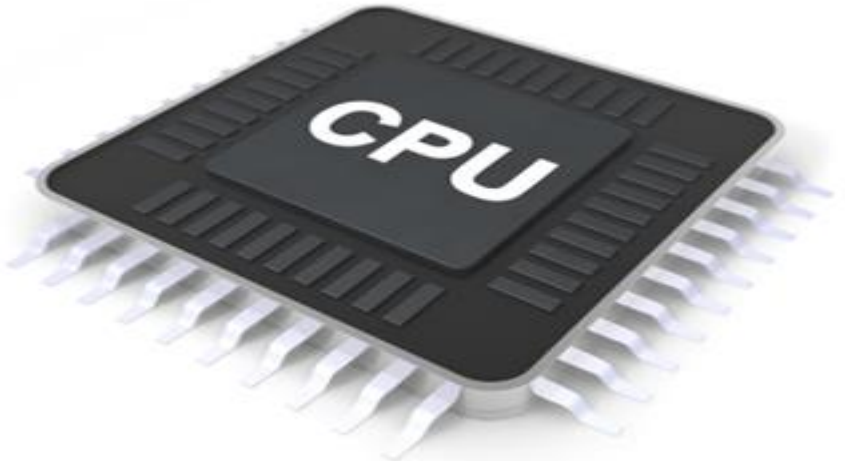


Timer Counter

ENEE 3587

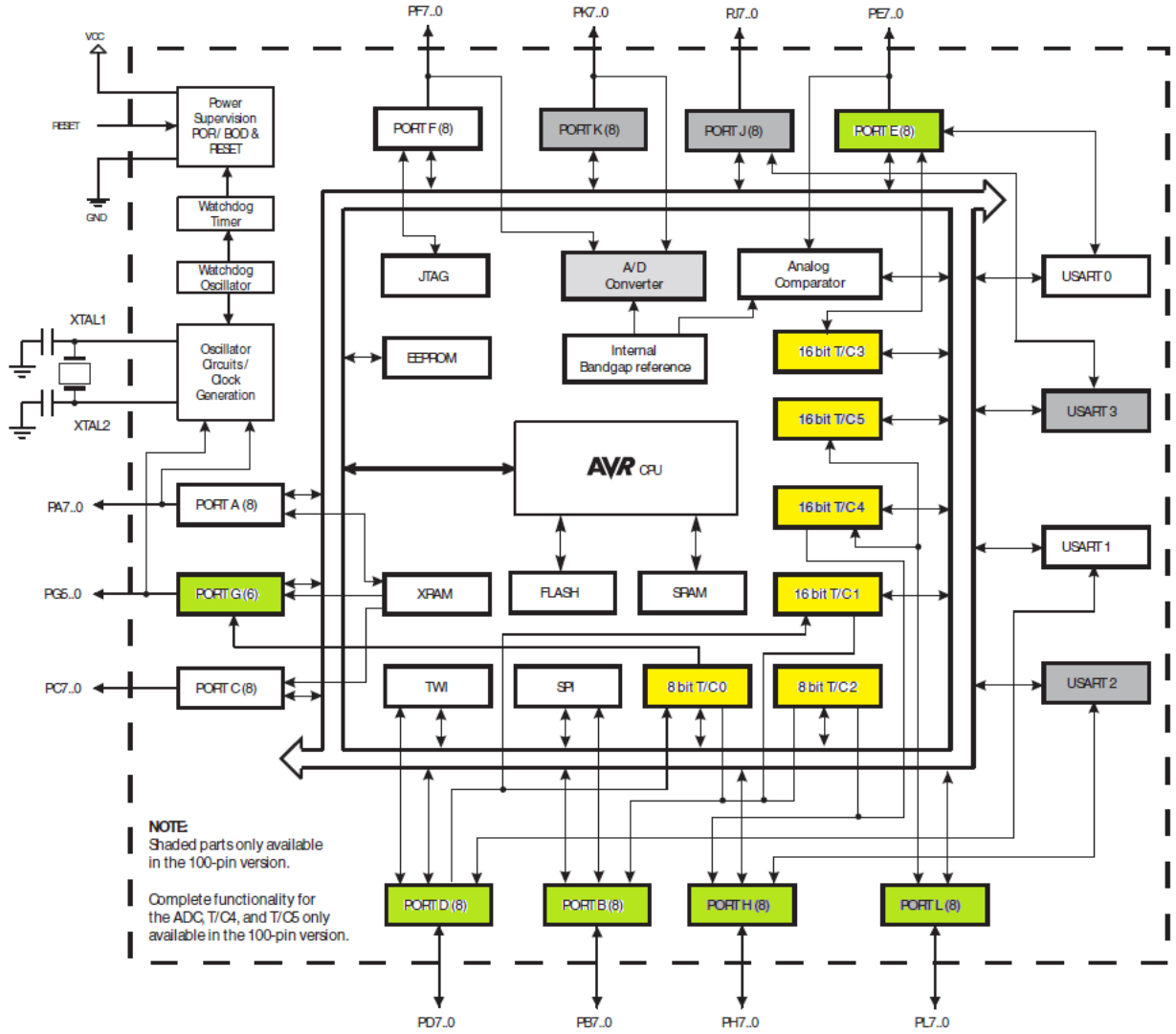
Microp Interfacing

Timer/Counter

- Counter used to time:
  - General delay functions
    - Nested for-loops: imprecise, taxing to resources
  - Input events
    - Events are rising edges, falling edges
    - Measure width of pulse, period
    - Input capture only on 16-bit timers
  - Output events
    - Create periodic signals
    - Pulse width modulation (PWM)

Timer/Counter Subsystem

- 2x 8-bit Timer/Counters:
  - TC0, TC2
  - Output Compare Mode
- 4x16-bit Timer/Counter
  - TC1,3,4,5
  - Output Compare mode
  - Input Capture mode



Pins

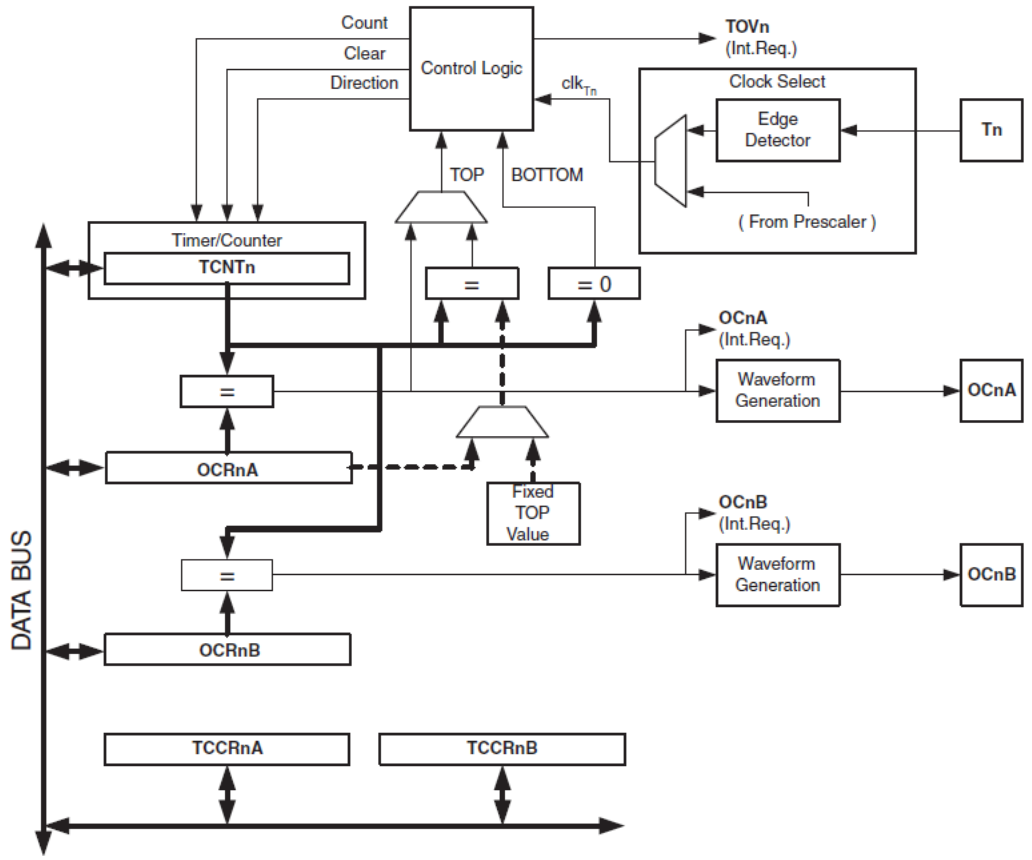
Pin Name	OC/IC	Mapped
PB7	OC0A/OC1C	Digital pin 13 PWM
PG5	OC0B	Digital pin 4 PWM
PB5	OC1A	Digital pin 11 PWM
PB6	OC1B	Digital pin 12 PWM
PB7	OC1C/OC0A	Digital pin 13 PWM
PD4	ICP1	No pin connection
PB4	OC2A	Digital pin 10 PWM
PH6	OC2B	Digital pin 9 PWM
PE3	OC3A	Digital pin 5 PWM
PE4	OC3B	Digital pin 2 PWM
PE5	OC3C	Digital pin 3 PWM
PE7	ICP3	No pin connection
PH3	OC4A	Digital pin 6 PWM
PH4	OC4B	Digital pin 7 PWM
PH5	OC4C	Digital pin 8 PWM
PL0	ICP4	Digital pin 49
PL3	OC5A	Digital pin 46 PWM
PL4	OC5B	Digital pin 45 PWM
PL5	OC5C	Digital pin 44 PWM
PL1	ICP5	Digital pin 48

PD4:6
PE6:7
are not mapped to a header on the board

8-bit Timer Counter

- **n** = 0 or 2 (Timer 0, 2)
- TCNT**n** circuit:
  - 8bit register
  - Count of clock cycles
  - Clocking sources can vary/controlled
- OC**n**A: output compare A
  - Output generated is binary
  - Output controlled by comparison
    - wait until TCNTn = OCRnA to produce output
- OC**n**B: output compare B
- TCCR**n**A: timer control registers A
- TCCR**n**A: timer control registers B

Figure 16-1. 8-bit Timer/Counter Block Diagram

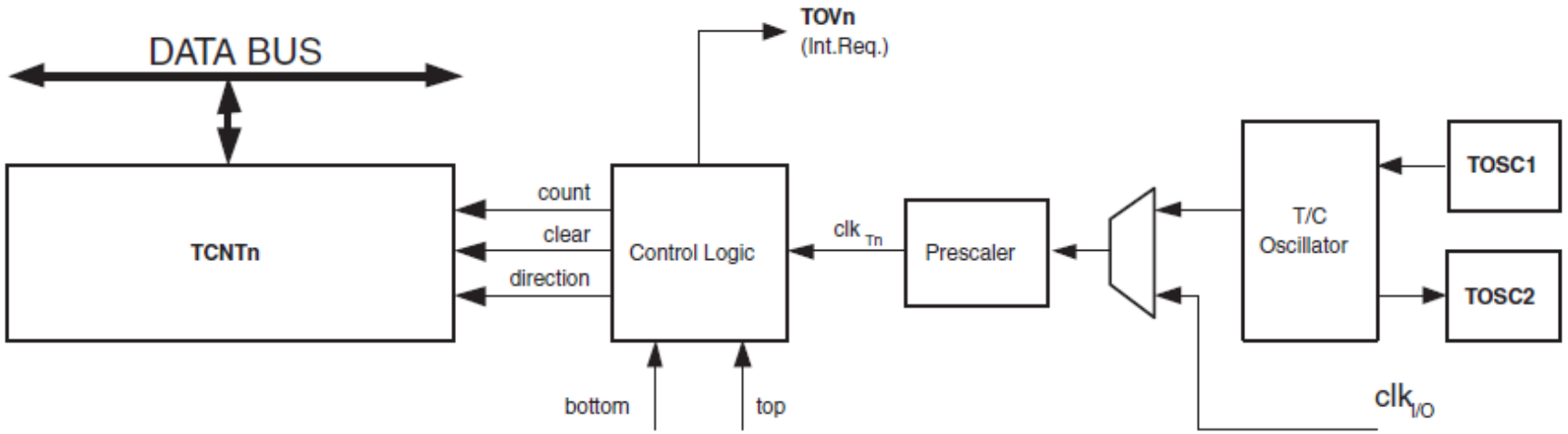


8-bit Timer Registers

- Timer **n** = 0,2
- Registers:
  - TCNT**n**: 8-bit counter register
  - TCCR**n**A: timer/counter control register A
  - TCCR**n**B: timer/counter control register B
  - ASSR: timer asynchronous status register
  - OCR**n**A: output compare register A
  - OCR**0**B: output compare register B
  - TIMSK**n**: timer counter interrupt mask register
  - TIFR**n**: timer counter interrupt flag register

TCNT Circuit

Prescaler	Clock divisor
clkT	Timer/Counter clock based on clock from T/C oscillator or clkIO
count	Increment or decrement TCNTn by 1.
direction	Selects between increment and decrement (count up or count down)
clear	Clear TCNTn
top	Signals TCNT has reached maximum value.
bottom	Signals that TCNT has reached min zero.
TOV	Overflow that TCNT has reached min/max.
TCNT	Count of cycles



8-bit Timer: TCNT0,2

Bit	7	6	5	4	3	2	1	0	
0x26 (0x46)	TCNT0[7:0]								TCNT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- 8-bit register
- Read TCNT**n** to detect current cycle count
- Write into TCNT**n** to set a count
  - Clears the compare match flag
  - Compare match flag = 1 when TCNT matches the count in OC

8-bit Timer: TCCR**n**A

7	6	5	4	3	2	1	0	
COMnA1	COMnA0	COMnB1	COMnB0	—	—	WGMn1	WGMn0	Reset value = 0x00
R/W	R/W	R/W	R/W	R	R	R/W	R/W	

- **COM<u>n</u>A1:COM<u>n</u>A0**: Compare match output mode A (<u>n</u> = timer 0 or 2)
  - Controls behavior of output
- **COM<u>n</u>B1:COM<u>n</u>A0**: Compare match output mode B (<u>n</u> = timer 0 or 2)
- **WGM<u>n</u>1:0**: Waveform generation mode bits
  - Helps control the counting sequence,
  - source for maximum (TOP) counter value,
  - type of waveform generation,
  - modes of operation

8-bit Timer: TCCR0B

7	6	5	4	3	2	1	0	
FOC0A	FOC0B	—	—	WGM02	CS02	CS01	CS00	Reset value = 0x00
W	W	R	R	R/W	R/W	R/W	R/W	

- **FOC0A**: Force output compare A
  - Force a change in output when a 1 is written to this bit (non-PWM mode)
  - No interrupt will be generated.
  - This bit is read as 0.
- **FOC0B**: Force output compare B
- **WGM02**: Waveform generation mode bit 2
- **CS02-CS00**: Clock select
  - 000: No clock source (Timer/Counter stops)
  - 001: clkI/O (no prescaling)
  - 010: clkI/O /8 (from prescaler)
  - 011: clkI/O /64 (from prescaler)
  - 100: clkI/O /256 (from prescaler)
  - 101: clkI/O /1024 (from prescaler)
  - 110: External clock source on T0 pin. Clock on falling edge
  - 111: External clock source on T0 pin. Clock on rising edge

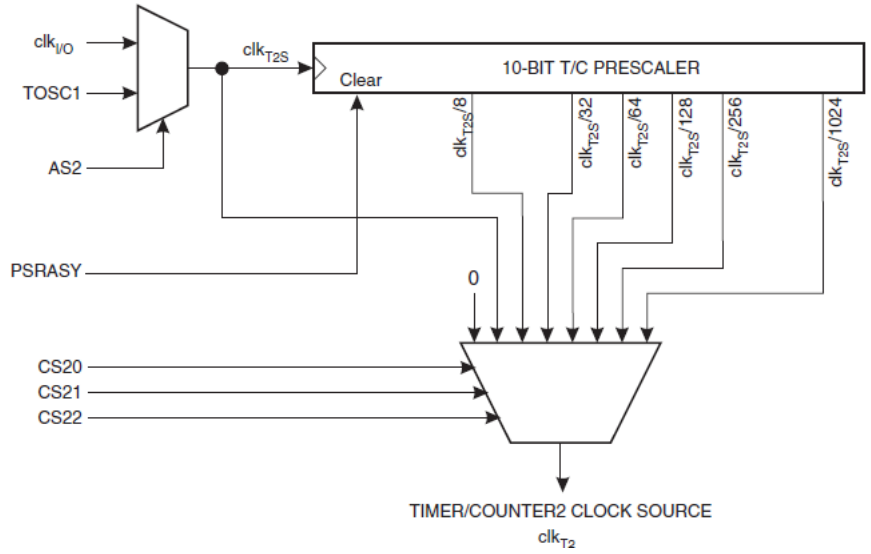
8-bit Timer: TCCR2B

7	6	5	4	3	2	1	0	
FOC0A	FOC0B	—	—	WGM02	CS02	CS01	CS00	Reset value = 0x00
W	W	R	R	R/W	R/W	R/W	R/W	

- **FOC2A**: Force output compare A
  - Force a change in output when a 1 is written to this bit (non-PWM mode)
  - No interrupt will be generated.
  - This bit is read as 0.
- **FOC2B**: Force output compare B
- **WGM22**: Waveform generation mode bit 2
- **CS22-CS20**: Clock select
  - 000: No clock source (Timer/Counter stops)
  - 001: clkT2S (no prescaling)
  - 010: clkT2S/8 (from prescaler)
  - 011: clkT2S/32 (from prescaler)
  - 100: clkT2S/64 (from prescaler)
  - 101: clkT2S/128 (from prescaler)
  - 110: clkT2S/256 (from prescaler)
  - 111: clkT2S/1024 (from prescaler)

Timer 0, 2 Clocking

- clk_{IO} is clock used to drive the general IO system
  - Same as the clk_{CPU}
- clk_{T2S} is the timer2 clock source
  - Synchronous mode: clk_{T2S} based on clk_{IO}
  - Asynchronous mode: clk_{T2S} based on timer oscillator (TOSC)
  - AS2 bit from ASSR sets (a)synchronous mode
- Prescaler: slows down the clock
  - Acts as period divisor
  - 10bit prescaler => max divisor = 1024



8-bit Timer Modes (TCCR)

- Normal operation: no output compare
  - Typically used for creating variable time delays
  - OC Mode: Generating binary output at specific counts
- CTC: Clear time on compare
  - Typically used for creating periodic time delays/functions
- PWM Mode: Pulse-width modulation
  - Fast PWM: high frequency PWM
  - Phase-correct PWM: high resolution PWM

Mode	WGM2	WGM1	WGM0	Timer/Counter mode of operation	TOP	Update of OCRx at	TOV flag set on(1)(2)
0	0	0	0	Normal	0xFF	immediate	MAX
1	0	0	1	PWM, phase correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRnA	immediate	MAX
3	0	1	1	Fast PWM	0xFF	TOP	MAX
4	1	0	0	Reserved	—	—	—
5	1	0	1	PWM, phase correct	OCRnA	TOP	BOTTOM
6	1	1	0	Reserved	—	—	—
7	1	1	1	Fast PWM	OCRnA	BOTTOM	TOP

Notes: 1. MAX = 0xFF.
2. BOTTOM = 0x00.

8-bit TCCR**n**A : COM**nx**1:COM**nx**2

- n = 0 or 2
- x = A or B

Compare match output behavior in **non-PWM** mode (8-bit timer)

COMnx1	COMnx0	Description
0	0	Normal PORT operation, OCnx flip-flop disconnected from OCnx pin
0	1	Toggle OCnx signal on compare match
1	0	Clear OCnx signal on compare match
1	1	Set OCnx signal on compare match

Compare match output behavior in **Fast-PWM** mode (**WGMn2:0 = 011**, TOP = 0xFF)

COMnx1	COMnx0	Description
0	0	Normal PORT operation, OCnx flip-flop disconnected from OCnx pin
0[1]	1[1]	Normal PORT operation, OCnA flip-flop disconnected from OCnA pin
1	0	Clear OCnx signal on compare match, set OCnx at BOTTOM (non-inverting mode)
1	1	Set OCnx signal on compare match, clear OCnx at BOTTOM (inverting mode)

Note: 1. This option is not available for OCnB (n = 0 or 2) module.

Compare match output behavior in **Fast-PWM** mode (**WGMn2:0 = 111**, TOP = OCRnA)

COMnx1	COMnx0	Description
0	0	Normal PORT operation, OCnx flip-flop disconnected from OCnx pin
0[1]	1[1]	Toggle OCnA on compare match
1	0	Set OCnx at BOTTOM (non-inverting mode)
1	1	Clear OCnx at BOTTOM (inverting mode)

Note: 1. This option is not available for OCnB (n = 0 or 2) module.

Compare match output behavior in **Phase-Correct-PWM** mode[1] (8-bit timer)

COMnx1	COMnx0	Description
0	0	Normal PORT operation, OCnx flip-flop disconnected from OCnx pin
0[2]	1[2]	WGMn2=0: normal PORT operation, OCnA flip-flop disconnected from OCnA pin
		WGMn2=1: Toggle OCnA on compare match
1	0	Clear OCnx flip-flop on compare match during up-counting
		Set OCnx flip-flop on compare match during down-counting
1	1	Set OCnx flip-flop on compare match during up-counting
		Clear OCnx flip-flop on compare match during down-counting

Notes: 1. A special case occurs when OCRnx equals TOP and COMnx1 is set to 1. In this case, the compare match is ignored, but the set or clear is done at the TOP.
2. This option is not available for OCnB (n = 0 or 2) module.

8-bit Timer: ASSR

7	6	5	4	3	2	1	0	
—	EXCLK	AS2	TCN2UB	OCR2AUB	OCR2BUB	TCR2AUB	TCR2BUB	Reset value = 0x00
R	R/W	R/W	R	R	R	R	R	

- **EXCLK**: Enable external clock input
  - 0: Crystal oscillator (32,768 Hz) is selected.
  - 1: Asynchronous clock (from TOSC1 pin) instead of the crystal oscillator is selected.
- **AS2**: Asynchronous Timer/Counter2
  - 0: Timer/Counter2 is clocked from the I/O clock, clkI/O.
  - 1: Timer/Counter2 is clocked from the TOSC1 pin (can be a crystal oscillator or an external clock).
- **TCN2UB**: Timer/Counter2 update busy
  - 0: TCNT2 is ready to be updated with a new value.
  - 1: Timer/Counter2 operates asynchronously and TCNT2 is written.
- **OCR2xUB**: Output-compare register 2 update busy (x = OC A or B)
  - 0: OCR2x is ready to be updated with a new value. (x = A or B)
  - 1: Timer/Counter2 operates asynchronously and OCR2A is written but hasn't received the new value from the temporary storage register.
- **TCR2xUB**: Timer/Counter control register 2 update busy (x = OC A or B)
  - 0: TCCR2x is ready to be updated with a new value.
  - 1: When Timer/Counter2 operates asynchronously and TCCR2A is written, this bit becomes set.

8-bit Timer: Output Compare Register

- Timer 0: OCR0A, OCR0B
- Timer 2: OCR2A, OCR2B
- OCR**nx** are 8-bit registers
  - **n** = 0, 2 for timer 0, 2
  - **x** = output channel A, B
  - There are 2 outputs in each 8-bit timer
- OCR: Registers used for compare match
  - When TCNT = OCR => perform an event
- “Events” depend on the mode: normal, CTC, PWM

8-bit Timer: TIFR0,2

7	6	5	4	3	2	1	0	
—	—	—	—	—	OCFnB	OCFnA	TOVn	Reset value = 0x00
R	R	R	R	R	R	R	R	

- **OCF<u>n</u>B**: Timer/Counter n output compare match B flag (**<u>n</u>**=0 or 2)
  - 0 = Compare match between TCNT<u>n</u> and OCR<u>n</u>B did not occur.
  - 1 = Compare match between TCNT<u>n</u> and OCR<u>n</u>B has occurred.
- **OCF<u>n</u>A**: Timer/Counter n output compare match A flag
  - 0 = Compare match between TCNT<u>n</u> and OCR<u>n</u>A did not occur.
  - 1 = Compare match between TCNT<u>n</u> and OCR<u>n</u>A has occurred.
- **TOV<u>n</u>**: Timer/Counter n overflow interrupt enable
  - 0 = TCNT<u>n</u> hasn't overflown since this flag was last cleared.
  - 1 = TCNT<u>n</u> has overflown since this flag was last cleared.

- To clear flags write a 1 into the flag

8-bit Timer: TIMSK0,2

7	6	5	4	3	2	1	0	
—	—	—	—	—	OCIEnB	OCIEnA	TOIEn	Reset value = 0x00
R	R	R	R	R	R/W	R/W	R/W	

- **OCIE<u>n</u>B**: Timer/Counter n output compare match B interrupt enable (**<u>n</u>** = 0 or 2)
  - 0 = Disable the OCnB match interrupt
  - 1 = Enable the OCnB match interrupt
- **OCIE<u>n</u>A**: Timer/Counter n output compare match A interrupt enable (**<u>n</u>** = 0 or 2)
  - 0 = Disable the OCnA match interrupt
  - 1 = Enable the OCnA match interrupt
- **TOIE<u>n</u>**: Timer/Counter n overflow interrupt enable (**<u>n</u>** = 0 or 2)
  - 0 = Disable Timer/Counter n overflow interrupt
  - 1 = Enable Timer/Counter n overflow interrupt

8-bit Timer: Polling vs Interrupts

- Timer flags registers (TIFR) is used to detect events:
  - When OCR = TCNT
  - When TCNT overflows
  - Use “polling” method to confirm an event
  - Flag is cleared by writing a 1 into TIFR
  - Fast but resource “hungry”
- Timer interrupt mask register (TIMSK) is used enable interrupts
  - 2 interrupts:
    - Detect when OCR = TCNT
    - Detect when TCNT overflows
  - Hardware “function calls” when an event occurs
  - Slow but frees resources

8-bit vs 16-bit Timers

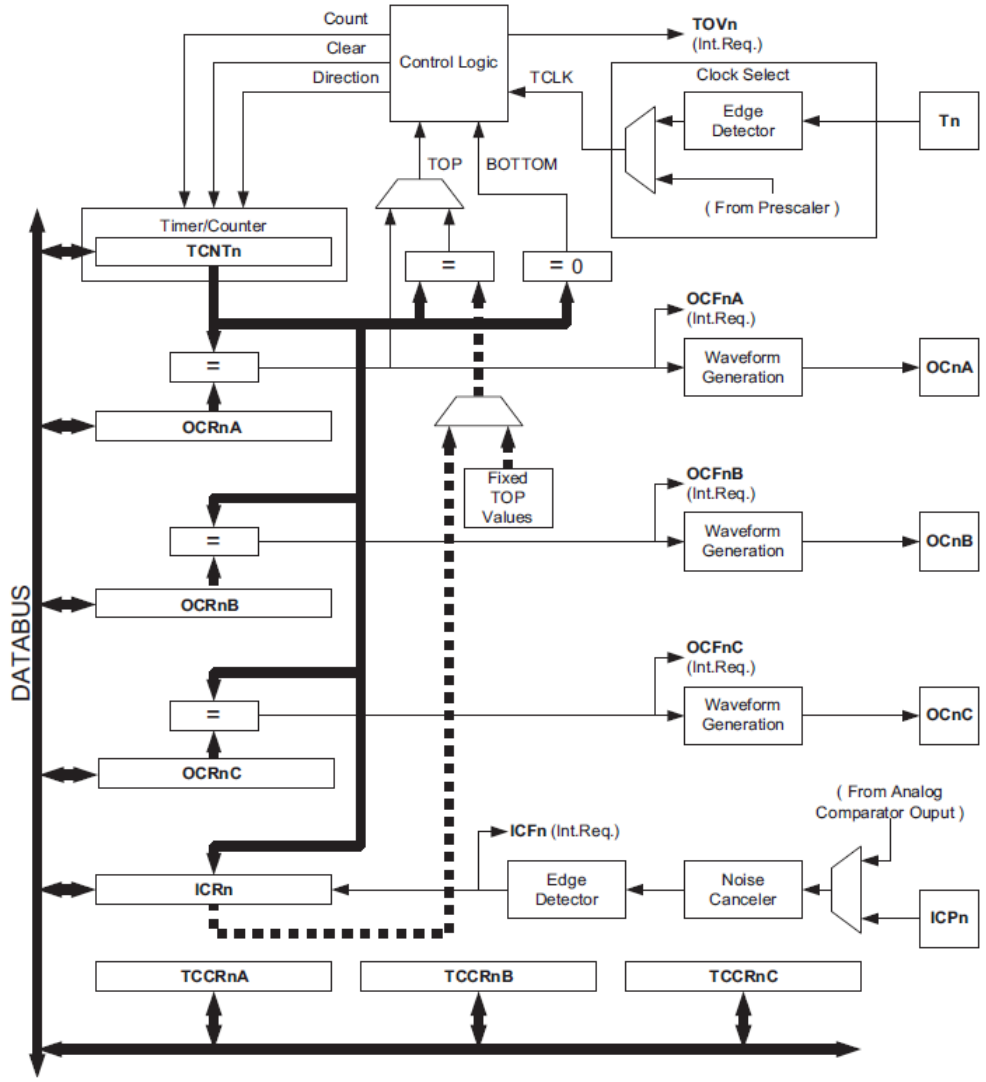
- 2x8-bit Timer 0,1
- 8-bit counter
- 2 output compares per timer
  - OCA, OCB
- PWM per timer
  - Fast, Phase correct

- 4x16-bit Timers 1,3,4,5
- 16-bit counter: higher delay times
- 3 output compares per timer
  - OCA, OCB, OCC
- PWM per timer
  - fast, phase correct, freq&phase correct
- 1 input capture per timer
  - ICP

16-bit Timer Registers

- Timer **n** =1,3,4,5
- Registers:
  - TCNT**n**H/TCNT**n**L: High and Low 16-bit counter registers (8-bit each)
  - TCCR**n**A: timer/counter control register A
  - TCCR**n**B: timer/counter control register B
  - TCCR**n**C: timer/counter control register C
  - OCR**n**AH, OCR**n**AL: High and Low output compare register A
  - OCR**n**BH, OCR**n**BL: High and Low output compare register B
  - OCR**n**CH, OCR**n**CL: High and Low output compare register C
  - ICR**n**H, ICR**n**L: High and Low input capture register
  - TIMSK**n**: timer counter interrupt mask register
  - TIFR**n**: timer counter interrupt flag register

16-bit Timer Circuit



16-bit Timer: TCNT**n**H, TCNT**n**L

- 2x8-bit registers

Bit	7	6	5	4	3	2	1	0	
(0x85)	TCNT1[15:8]								TCNT1H
(0x84)	TCNT1[7:0]								TCNT1L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
(0x95)	TCNT3[15:8]								TCNT3H
(0x94)	TCNT3[7:0]								TCNT3L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
(0xA5)	TCNT4[15:8]								TCNT4H
(0xA4)	TCNT4[7:0]								TCNT4L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
(0x125)	TCNT5[15:8]								TCNT5H
(0x124)	TCNT5[7:0]								TCNT5L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

16-bit : TCCR1A, TCCR3A, TCCR4A, TCCR5A,

7	6	5	4	3	2	1	0	
COMnA1	COMnA0	COMnB1	COMnB0	COMnC1	COMnC0	WGMn1	WGMn0	Reset value = 0x00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

- **COM<u>n</u>A1:0**: Compare output mode for channel A (**<u>n</u>** = 1, 3, 4, or 5)
- **COM<u>n</u>B1:0**: Compare output mode for channel B
- **COM<u>n</u>C1:0**: Compare output mode for channel C
  - Controls the compare match behavior.
- **WGMn1:0**: Waveform generation mode bits
  - Helps control the counting sequence,
  - source for maximum (TOP) counter value,
  - type of waveform generation,
  - modes of operation

16-bit: TCCR1B, TCCR3B, TCCR4B, TCCR5B

7	6	5	4	3	2	1	0	
ICNCn	ICESn	—	WGMn3	WGMn2	CSn2	CSn1	CSn0	Reset value = 0x00
R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	

- **ICNC<u>n</u>**: Input-capture noise canceler (**<u>n</u>** = 1,3,4,5)
  - 0 = Disable input-capture noise canceler.
  - 1 = Enable input-capture noise canceler.
- **ICES<u>n</u>**: Input-capture edge select
  - 0 = Falling edge triggers input capture.
  - 1 = Rising edge triggers input capture.
  - (If ICRn is used as TOP value, then the ICPn is disconnected and input capture is disabled).
- **WGM<u>n</u>3:2**: Waveform generation mode bits 3 and 2
- **CS<u>n</u>2-CS<u>n</u>0**: Clock select
  - 000 = No clock source (Timer/Counter stopped)
  - 001 = clkI/O (no prescaling)
  - 010 = clkI/O /8 (from prescaler)
  - 011 = clkI/O /64 (from prescaler)
  - 100 = clkI/O /256 (from prescaler)
  - 101 = clkI/O /1024 (from prescaler)
  - 110 = External clock source on Tn pin. Clock on falling edge
  - 111 = External clock source on Tn pin. Clock on rising edge

16-bit Timer Modes

Mode	WGMn3	WGMn2	WGMn1	WGMn0	Timer/Counter mode of operation	TOP	Update of OCRx at	TOV flag set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, phase correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, phase correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, phase correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCRnA	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, phase and frequency correct	ICRn	BOTTOM	BOTTOM
9	1	0	0	1	PWM, phase and frequency correct	OCRnA	BOTTOM	BOTTOM
10	1	0	1	0	PWM, phase correct	ICRn	TOP	BOTTOM
11	1	0	1	1	PWM, phase correct	OCRnA	TOP	BOTTOM
12	1	1	0	0	CTC	ICRn	Immediate	MAX
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICRn	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCRnA	BOTTOM	TOP

16-bit TCCR**n**A : COM**nx**1,COM**nx**2

- **n** = 1,3,4,5
- **x** = A,B,C

Compare match output behavior in **non-PWM** mode (16-bit timer)

COMnx1	COMnx0	Description
0	0	Normal PORT operation, OCnA/OCnB/OCnC disconnected
0	1	Toggle OCnA/OCnB/OCnC on compare match
1	0	Clear OCnA/OCnB/OCnC (pull low) on compare match
1	1	Set OCnA/OCnB/OCnC (pull high) on compare match

Compare match output behavior in **Fast-PWM** mode (16-bit timer)

COMnx1	COMnx0	Description
0	0	Normal PORT operation, OCnA/OCnB/OCnC disconnected
0	1	WGMn 3:0 = 14 or 15: toggle OCnA on compare match, OCnB and OCnC disconnected (normal PORT operation). For all other PWM WGMn settings, normal PORT operation, OCnA/OCnB/OCnC disconnected.
1	0	Clear OCnA/OCnB/OCnC (pull low) on compare match, set OCnA/OCnB/OCnC at BOTTOM (non-inverting mode)
1	1	Set OCnA/OCnB/OCnC (pull high) on compare match, clear OCnA/OCnB/OCnC at BOTTOM (inverting mode)

Compare output mode, **Phase Correct** and **Phase-and-Frequency-Correct PWM** mode (16-bit)

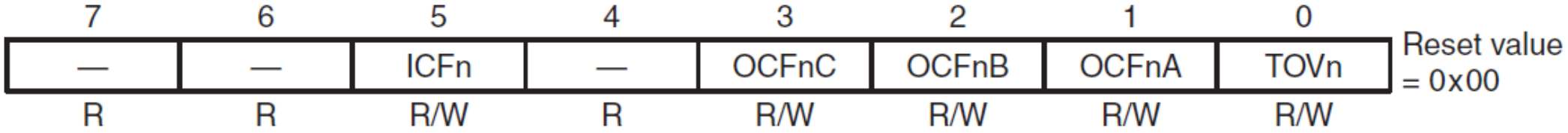
COMnx1	COMnx0	Description
0	0	Normal PORT operation, OCnA/OCnB/OCnC disconnected
0	1	WGMn 3:0 = 9 or 11: toggle OCnA on compare match, OCnB and OCnC disconnected (normal PORT operation). For all other PWM WGMn settings, normal PORT operation, OCnA/OCnB/OCnC disconnected.
1	0	Clear OCnA/OCnB/OCnC (pull low) on compare match when up-counting, set OCnA/OCnB/OCnC on compare match when down-counting
1	1	Set OCnA/OCnB/OCnC (pull high) on compare match when up-counting. Clear OCnA/OCnB/OCnC on compare match when down-counting.

16-bit : TCCR1C, TCCR3C TCCR4C, TCCR5C

7	6	5	4	3	2	1	0	
FOCnA	FOCnB	FOCnC	—	—	—	—	—	Reset value = 0x00
R/W	R/W	R/W	R	R	R	R	R	

- **FOC<u>n</u>A**: Force output compare for channel A (**<u>n</u>**= 1,3,4,5)
- **FOC<u>n</u>B**: Force output compare for channel B
- **FOC<u>n</u>C**: Force output compare for channel C
  - These bits are active only when WGMn3:0 bits specify a non-PWM mode.
  - FOC**<u>n</u>**A/FOCnB/FOCnC = 1 => OCnA/OCnB/OCnC output is changed

16-bit: TIFR**n**



- **ICF<u>n</u>**: Timer/Counter n, input-capture flag (**<u>n</u>** = 1,3,4,5)
  - 0 = No capture event occurred on the ICPn pin.
  - 1 = A capture event has occurred on the ICPn pin.
- **OCF<u>n</u>C**: Timer/Counter n output compare C match flag
  - 0 = TCNTn has not matched the OCRnC register since this flag was last cleared.
  - 1 = TCNTn has matched the OCRnC register.
- **OCF<u>n</u>B**: Timer/Counter n output compare B match flag
  - 0 = TCNTn has not matched the OCRnB register since this flag was last cleared.
  - 1 = TCNTn has matched the OCRnB register.
- **OCF<u>n</u>A**: Timer/Counter n output compare A match flag
  - 0 = TCNTn has not matched the OCRnA register since this flag was last cleared.
  - 1 = TCNTn has matched the OCRnA register.
- **TOV<u>n</u>**: Timer/Counter n, overflow interrupt flag
  - 0 = TCNTn has not overflowed since this flag was last cleared.
  - 1 = TCNTn has overflowed.

16-bit: TIMSK1,3,4,5

7	6	5	4	3	2	1	0	
—	—	ICIEn	—	OCIEnC	OCIEnB	OCEnA	TOIEn	Reset value = 0x00
R	R	R/W	R	R/W	R/W	R/W	R/W	

- **ICIE<u>n</u>**: Timer/Counter n, input-capture interrupt enable (**<u>n</u>** = 1,3,4,5)
  - 0 = Disable Timer/Counter n input-capture interrupt
  - 1 = Enable Timer/Counter n input-capture interrupt
- **OCIE<u>n</u>C**: Timer/Counter n output compare C match interrupt enable
  - 0 = Disable Timer/Counter n output compare C match interrupt
  - 1 = Enable Timer/Counter n output compare C match interrupt
- **OCIE<u>n</u>B**: Timer/Counter n output compare B match interrupt enable
  - 0 = Disable Timer/Counter n output compare B match interrupt
  - 1 = Enable Timer/Counter n output compare B match interrupt
- **OCIE<u>n</u>A**: Timer/Counter n output compare A match interrupt enable
  - 0 = Disable Timer/Counter n output compare A match interrupt
  - 1 = Enable Timer/Counter n output compare A match interrupt
- **TOIE<u>n</u>**: Timer/Counter n, overflow interrupt enable
  - 0 = Disable Timer/Counter n overflow interrupt
  - 1 = Enable Timer/Counter n overflow interrupt

Applications

- Creating delays/delay functions
- Generating binary (periodic) signals
- Detecting binary (periodic) signals

8bit vs 16bit Counter

- Time is measured using clock cycles
  - 8-bit counter: $count_{\max} = 255$ cycles
  - 16-bit counter: $count_{\max} = 65535$ cycles

$$count = counter_{\text{end}} - counter_{\text{start}}$$

$$time = \frac{count}{freq}$$

  - Pre-scaler: divisor used to slow down clock further:

$$time = \left(\frac{count}{freq}\right) prescaler$$

  - Timer/counter overflow: when timer counts 1 over max => resets to zero

$$count = count_{\text{end}} + overflow * count_{\max} - count_{\text{start}}$$

Application: Delays

- Given a time delay, clock frequency: determine count, prescaler

$$count = \frac{delay * freq}{prescaler}$$

  - For 8-bit timer: count < 256; for 16-bit timer count < 65536
  - 16-bit timers give the best range
- Setup timer for normal mode
- To detect delay time use flags or interrupts
  - Flags: use polling method
  - Interrupts: use interrupt service routine (ISR function)
- Delay time can be detected by overflow method or OC method
  - Overflow: set TCNT = -count
  - OC : set OCR = TCNT + count
- Remember to clear flags

Delay Example

- Given f_{clk} = 16MHz: create 100ms delay
  - 100ms * 16MHz = 1600K = 1.6M
  - Prescaler = 1024, 256, 64, 8 => count = 1562.5, 6250, 25K, ~~200K~~
    - Count > 255 => Must use 16bit Timer 1,3,4,5
  - Avoid 0.5, e.g. 64 => count = 1.6M/64 = 25K
  - Using overflow method: TCNT = -25K
    - Same as 65536 – 25K = 40536
  - Using OC method: OCR = TCNT + 25K
    - Don’t worry about OV

Delay Example Code

- 100ms delay using overflow method:

```
TCCR1A = 0;          // configure Timer 1 to normal mode with clock source
TCCR1B = 0x03;       // set to clkI/O/64
TCNT1 = -25000;      // loads 40536 into TCNT1 so that it overflows in 25000 cycles
TIFR1 = 1;           // clear TOV flag
while(!(TIFR1 & 1));      // polling method: wait for TOV flag to set.
TIFR1 = 1;           // clear TOV flag
```

- 100ms delay using OC method:

```
TCCR1A = 0;          // configure Timer 1 to normal mode with clock source
TCCR1B = 0x03;       // set to clkI/O/64
OCR1A = TCNT1 + 25000;    // use A channel. Set compare in 25000 cycles
TIFR1 = 2;                // clear OCA match flag
while(!(TIFR1 & 2));      // polling method: wait for OCF1A flag to set.
TIFR1 = 2;                // clear OCF1A flag
```

Delays Overhead

- Overhead: Code executed at the beginning and end of the delay
- In the previous example: 17 cycles before while loop + 2 cycles after
  - ~ 1.19 us
    - Can add up for continuous/periodic counting applications
- Example: 1ms delay executed 1000 times

```
for (int i=0; i<1000; i++)
{       TCCR1A = 0;    // configure Timer 1 to normal mode with clock source
        TCCR1B = 0x03;                // set to clkI/O/64
        OCR1A = TCNT1 + 25000;        //in 25000 cycles
        TIFR1 = 2;                    // clear OCA match flag
        while(!(TIFR1 & 2));  // polling method: wait for OCF1A flag to set.
        TIFR1 = 2;
}
```

  - Overhead = 25008 cycles ~1.56ms

Periodic/Continuous Counting

- Given a time delay, clock frequency: determine count, prescaler
- Setup timer for CTC mode
  - Use OCR**n**A for count
  - Can also use ICR**n** for count
- To detect delay time use flags or interrupts
- Delay time can be detected by OC method
  - OCF**n**A (0x02) or ICF**n** (0x20)
- Remember to clear flags

Continuous Count/Delay Example

- Given fclk = 16MHz: create kx1ms delay
  - 1ms * 16MHz = 16K = 16000
  - 8-bit counter: Prescaler = 1024,256,64,8 => count = 15.625, 62.5, 250, ~~2000~~
  - 16-bit counter: Prescaler = 1024,256,64,8 => count = 15.625, 62.5, 250, 2000
- Code: Overhead 4 cycles + 7 cycles

```
TCCR1A = 0x00;          // configure Timer 1 to CTC mode with clock
TCCR1B = 0x0A;          // set prescaler to 8 (WGM1:0 = 01)
TCNT1 = 0;              // TCNT1 counts up from 0
OCR1A = 2000-1;         // CTC TOP = OCR1A.
TIFR1 = 0x02;           // clear OCF1A flag
while(k) {              // set K for whatever k*1ms
        while (!(TIFR1 & 2));       // wait for 1 ms.
        TIFR1 = 0x02;               // clear OCF1A flag
        k--;                        // decrement count
}
```

Digital Waveform Generation

- Given:
  - waveform characteristic: frequency, duty cycle
    - Frequency is optional
  - IO clock: f_{CLK}
- Determine:
  - HIcount = cycles HI, LOcount = cycles LO, P = period count
  - Find appropriate prescaler
  - Use overflow counter when prescaler is insufficient

$$P = \frac{1}{f_{signal}}\left(\frac{f_{CLK}}{prescaler}\right)$$

- Setup Timer for normal mode
- Setup PORT pins for output
- Set output event upon match: OC goes HI, LO, toggle
  - Wait LOcount before going HI
  - Wait HIcount before going LO
- Use OC flags (polling) or interrupts
  - Clear flags

Example Periodic Waveform

- Use OC to generate a 50Hz square wave, 75% duty, f_{clk} = 16 MHz.
  - P = (1/50)*16M = 320K
  - Prescaler = 1024, 256, 64, 8
  - P/prescaler = 312.5, 1250, 5K, 40K
  - HICount = 0.75*P/prescaler = 234.375, 937.5, 3750, 30K
  - LOcount = 0.25*P/prescaler = 78.125, 312.5, 1250, 10K
- Solution:
  1. Use 16-bit timer 1: OC1A
  2. Setup timer, port
  3. Force output to go HI
  4. Set output to toggle
  5. Wait HIcount then go LO
  6. Wait LOcount then go HI
  7. Repeat 5-6

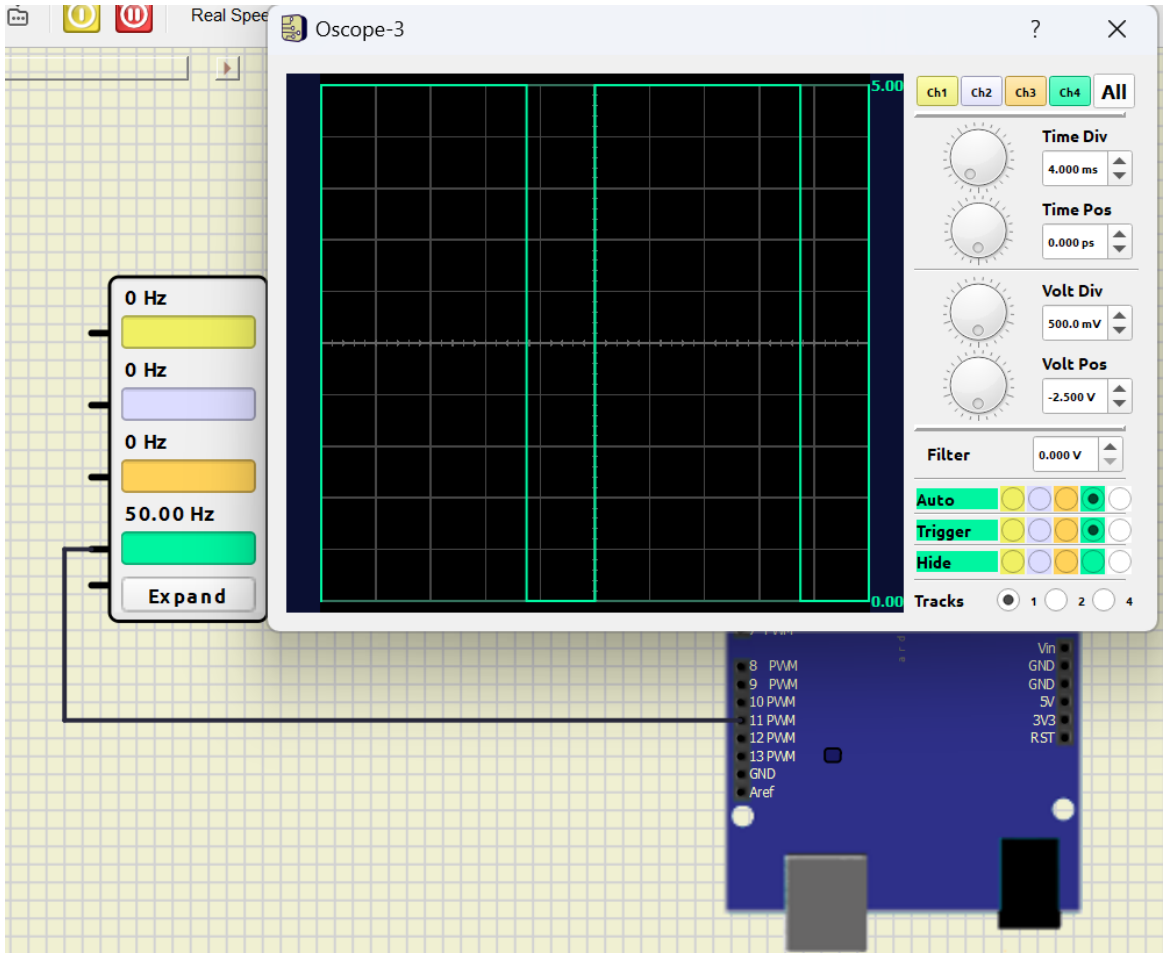
Code: Periodic Waveform Generation

```
#define HIcnt 30000
#define LOcnt 10000

DDRB |= 0x20;          // configure OC1A pin for output
TCCR1A = 0xC0;         // OC1A pin pull high on compare match
TCCR1B = 0x02;         // normal mode, prescaler 8
TCCR1C |= 0x80;        // force OC1A pin to high
TCCR1A = 0x40;         // select toggle as the OC1A pin action on compare match
while(1)               // infinite loop
{       TIFR1 = 0x02;  // clear the OCF1A flag
        OCR1A = TCNT1 + HIcnt;        // set HI count
        while (!(TIFR1 & 2));         // wait HI count, then toggle
        TIFR1 = 0x02;                 // clear the OCF1A flag
        OCR1A = TCNT1 + LOcnt;        // set LO count
        while (!(TIFR1 & 2));         // wait LO count, then toggle
}
```

Simulation IDE: 50Hz, 75%, OC

- Period = 5*4ms = 20ms
- Freq = 1/P = 50Hz
- Duty = 15ms/20ms = 75%
  - Uncheck **Auto** and adjust **Time Div** to 1ms and measure LO time.



Periodic 50% Duty Waveform Gen

- 50% duty cycle => HI/LO count are the same
- Use CTC for waveform generation
  - Avoid overhead
- Process:
  - Determine Period count, HI/LO counts (50%), prescaler
  - Setup timer for CTC mode
  - Setup PORT pins
  - Setup event = toggle
  - Clear TCNT
  - OCR = count - 1 (TCNT starts counting at 0)
  - Infinite loop

Example: OC periodic 50% waveform gen

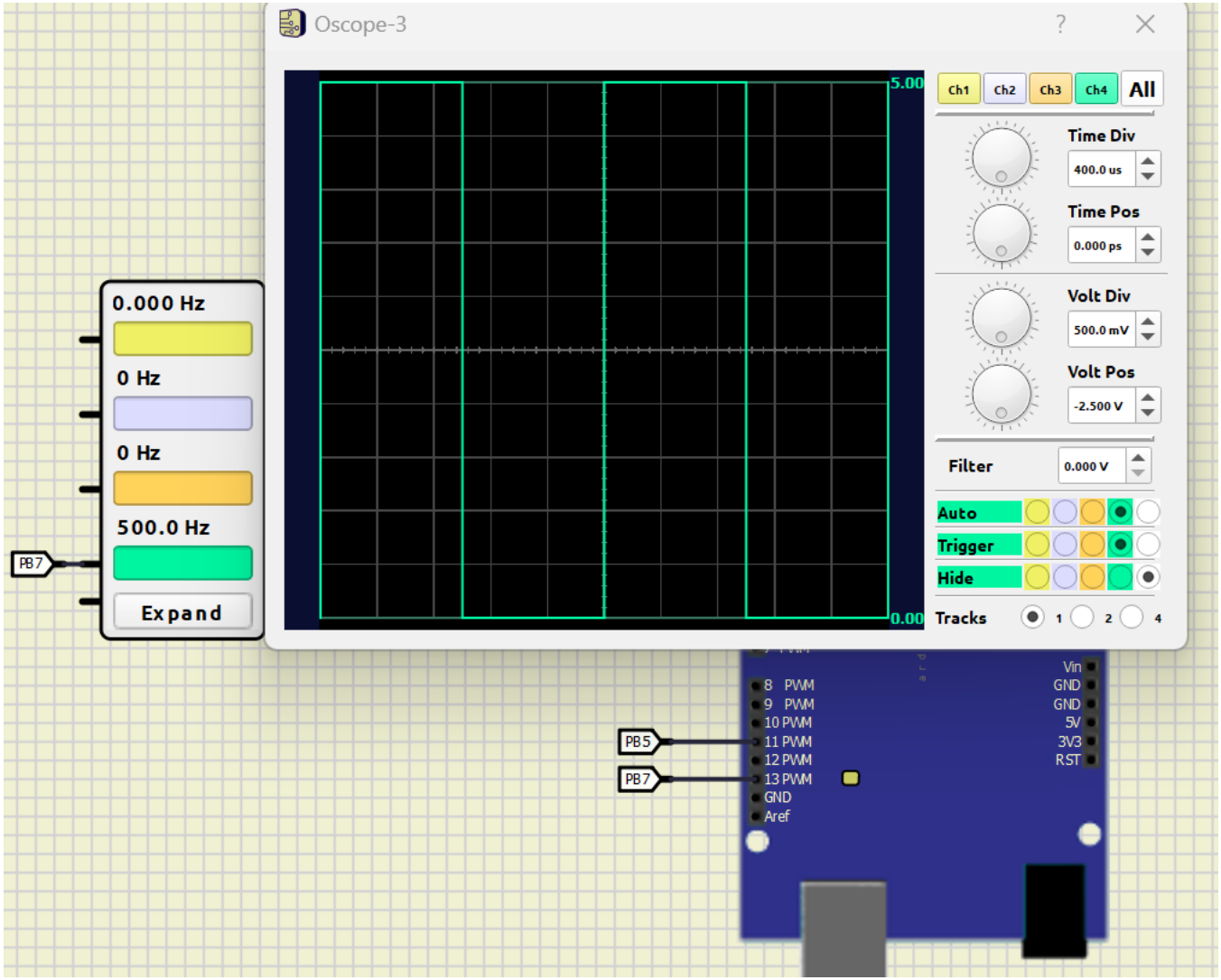
- Generate 500 Hz square wave, 50% duty, f_{clk} =16 MHz.
  - P = 16M/500 = 32K
  - Prescaler = 1024, 256, 64, 8 =>
  - P/prescaler = 31.25, 125, 500, 4K
  - HI/LO count = 15.625, 62.5, 250, 2K
- Setup:
  - Use Timer 0 (8-bit)
  - Use OC0A = PORTB pin 7
  - Set CTC mode
  - Set output to toggle
    - No force output needed (HI = LO)
  - OCR0A = 250-1
  - TCNT0 overflows => 250 cycles (LO/HI)

Code: OC periodic 50% waveform gen

```
DDRB |= 0x80;       // configure the OC0A pin for output
TCCR0A = 0x42;      // OC0A pin to toggle on compare match
TCCR0B = 0x03;      // CTC mode, prescaler 64
TCNT0 = 0;          // force TCNT0 to count up from 0
OCR0A = 249;        // start the first compare operation
while(1);           // wait for waveform to be generated
```

Simulation IDE: 500Hz, 50%, CTC

- Period = 5*400us = 2ms
- Freq = 1/P = 500Hz
- Duty = 1ms/2ms
  - Adjust **Time Div** to 200us



IC to Measure Signal Characteristics

- IC can only measure binary signal input
  - Square like signals
- Only available in 16-bit Timers 4,5
  - Timer1,3: ICP pins not connected
- Signal characteristics:
  - Period
  - Frequency
  - Duty cycle
- Measurements are in the form of counts
  - Can be converted to real values
  - Problem: when counts overflows

Measuring Period

- Period count w/ OV: $P = TCNT_{end} + OV * TCNT_{max} - TCNT_{start}$
  - $TCNT_{start}$: TCNT at 1st rising edge
  - $TCNT_{end}$: TCNT at 2nd rising edge
  - Also works for 2 falling edges
  - OV: number of overflow
  - $TCNT_{max}$: max count (65536 or 256)
  - Period count no OV: $P = TCNT_{end} - TCNT_{start}$
- To convert period to seconds:

$$prescaler * \frac{P}{f_{CLK}}$$

  - P: period count
  - f_{CLK}: IO clock

Measuring Frequency

- Method 1:
  - Implement a period count then use formula to calculate frequency

$$f = \left(\frac{f_{clk}}{P * prescaler} \right)$$

- Method 2: count rising edges (or falling edges) in 1 sec
  - Uses OC to time 1s
    - More accurate when using interrupts
  - Can use more than 1s for greater accuracy

IC Measuring Duty Cycle

- Duty cycle:

$$d = \frac{T_2 - T_1}{T_3 - T_1}$$

  - T_1: count for 1st rising edge
  - T_2: count for 1st falling edge
  - T_3: count for 2nd rising edge
  - wo OV: T_1, T_2, T_3 are based on instant TCNT reading
- Can be complicated by OV:

$$T_i = TCNT_i + OV * TCNT_{max}$$

Example: IC (no OV)

- A signal has a frequency between 10Hz and 10KHz. Measure its period. f_{CLK}=16MHz.
  - Max period count = 16M/10 = 1600K
  - Min period count = 16M/10K = 16K
  - Prescaler = 1024, 256, 64, 8
  - max period = 1562.5, 6250, 25K, ~~200K~~
  - min period = 1.5625, 6.250, 25, 200
- Setup:
  - Timer 4(16bit), ICP4: port L pin 0
  - Normal mode, input capture
  - Event = positive edges
  - ICR4 will contain cycle at which event occurs

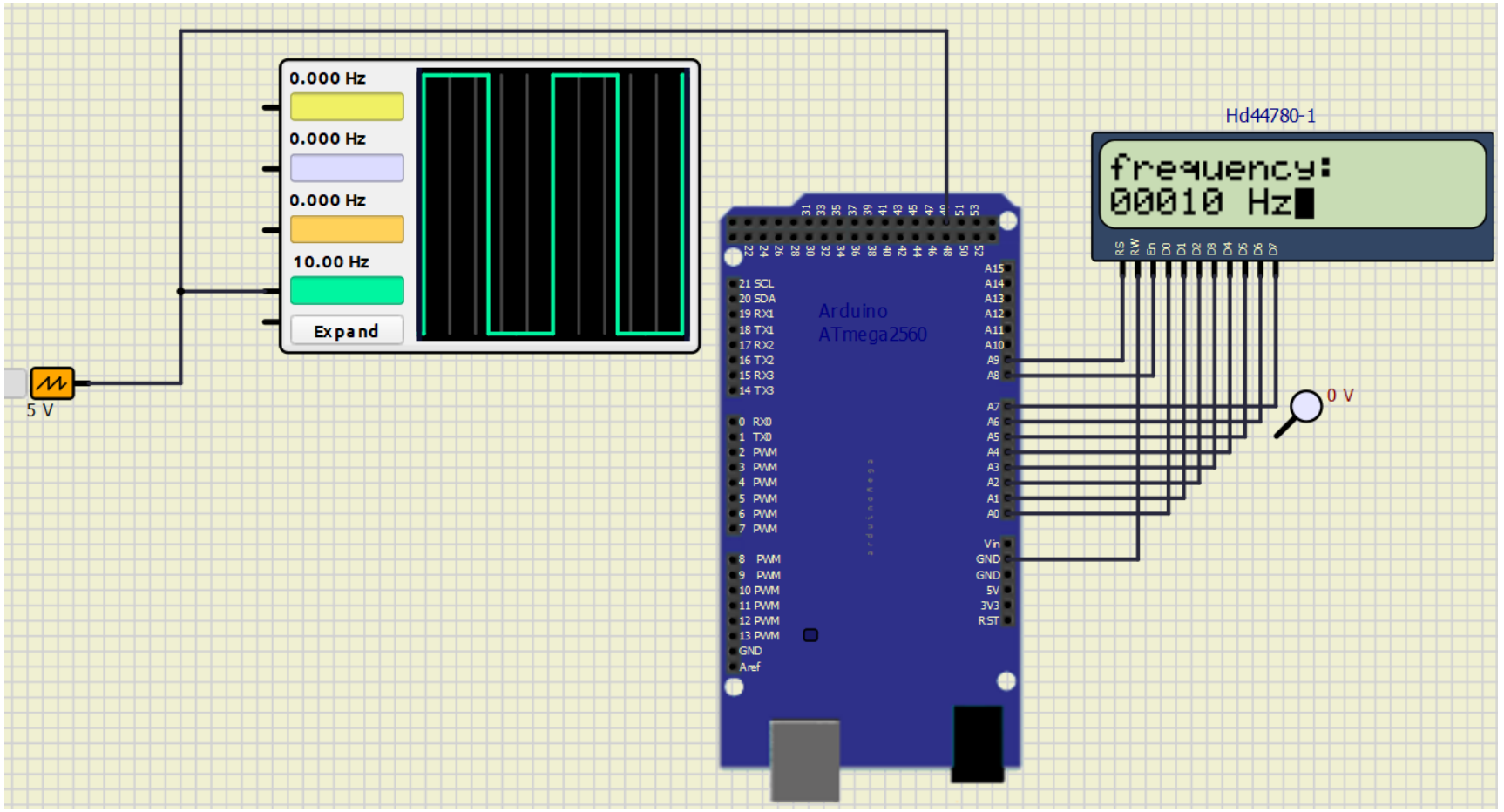
Code: IC (no OV)

```
unsigned int period;

TCCR4A = 0;          // Timer 4 to normal mode
DDRL &= 0xFE;        // ICP4 input (PL0)
TIFR4 = 0x2F;        // clear all flags related to Timer 4
TCCR4B = 0x43;       // capture rising edge, use prescaler = 64
TCNT4 = 0;           // count up from 0
while(!(TIFR4 & 0x20));   // wait for 1st rising edge
period = ICR4;
TIFR4 = 0x21;             // clear ICF4 and OVF
while(!(TIFR4 & 0x20));   // wait for 2nd rising edge
TCCR4B = 0;               // stop Timer 4
period = ICR4 - period;   // period count
while(1);
```

Simulation IDE: IC

- Use wave generator to simulate signals of different frequencies
  - Selections from square, triangular, sinusoid signals
  - Control frequency
  - Control duty cycle

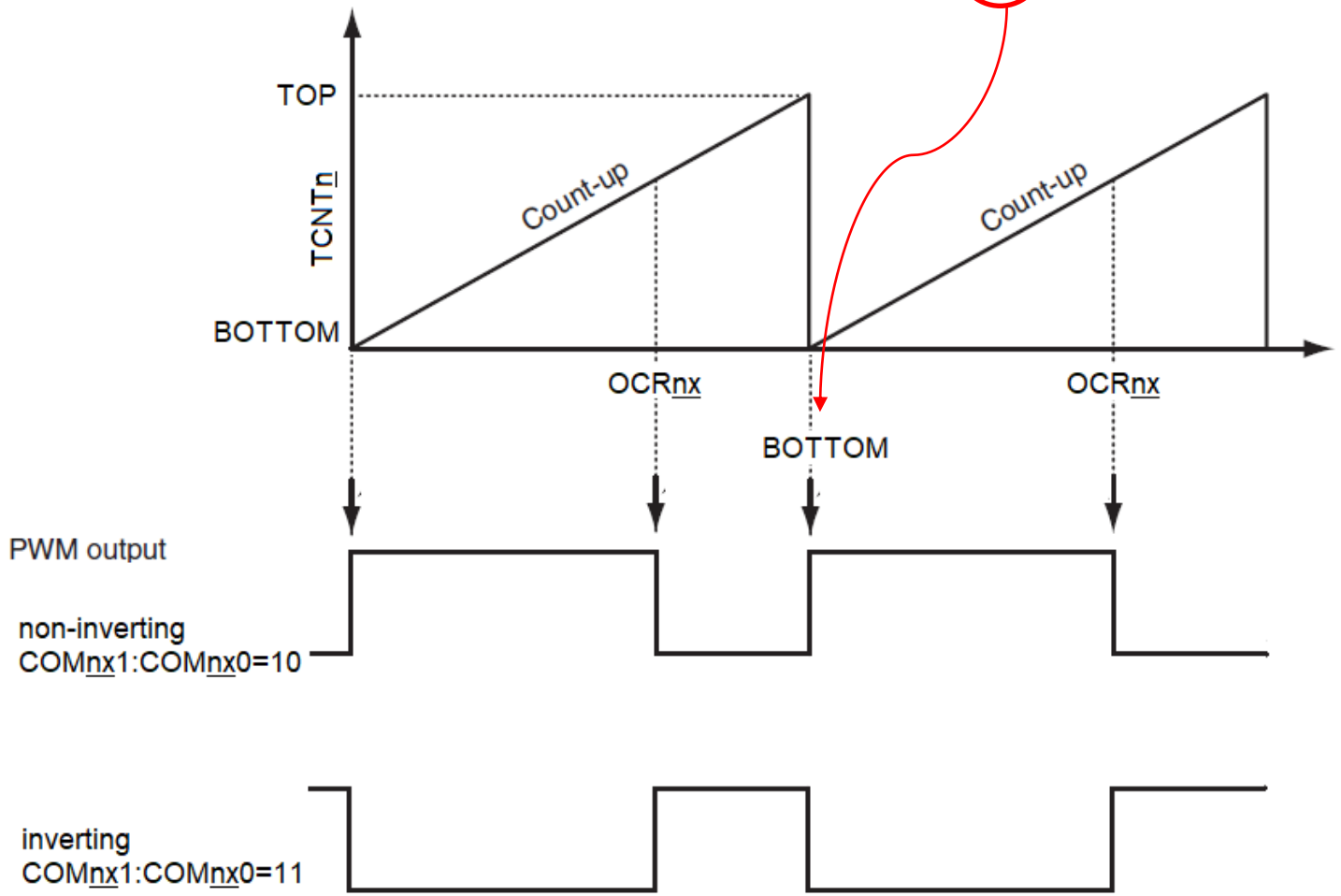


PWM

- Pulse width modulation:
  - Uses the width (duty cycle) to communicate information, or
  - Control devices
- Easy setup:
  - No need to constantly monitor of flags, drive output
- PWM types:
  - Fast PWM
    - aka single slope PWM
  - Phase correct (PC) PWM
    - aka dual slope PWM
  - Phase and frequency correct (PFC) PWM
    - Dual slope PWM

Fast PWM: Single Slope

$$f_{fastPWM} = \frac{f_{CLK}}{prescaler * (TOP + 1)}$$



Fast PWM

- TCNTn counts BOTTOM to TOP, repeats
  - Output: OCnA, OCnB, OCnC
- 8-bit fast PWM: Timer n: n=0,2
  - BOTTOM: 0
  - TOP: 0xFF, OCRnA
- 16-bit fast PWM: Timer n: n=1,3,4,5
  - BOTTOM: 0
  - TOP: 0x00FF, 0x01FF, 0x3FF, ICRn, OCRnA
- Can't use OCnA when TOP=OCRnA
- Can't use ICn when TOP=ICRn

Fast PWM Setup: Duty Cycle Only (No Freq)

- Many applications require a duty cycle only
  - E.g. PWM to control LED brightness
- Calculate duty cycle count: $d_{count} = (TOP + 1) * d\%$
  - Choose TOP so that count is an integer value
  - +1 is needed because count starts at 0
- Easiest solution: OCRnA = TOP, OCRnB = d_{count}
  - OCRnA = 99, 999, or 999
    - Can't use OCnA for output
    - Helps avoid fraction in multiplication
  - 8-bit: WGMn2:WGMn0 = 7
  - 16-bit WGMn3:WGMn0 = 0xF
- Inverting typically not needed in this application
- Prescaler = 1

Example: Fast PWM Cycle Only

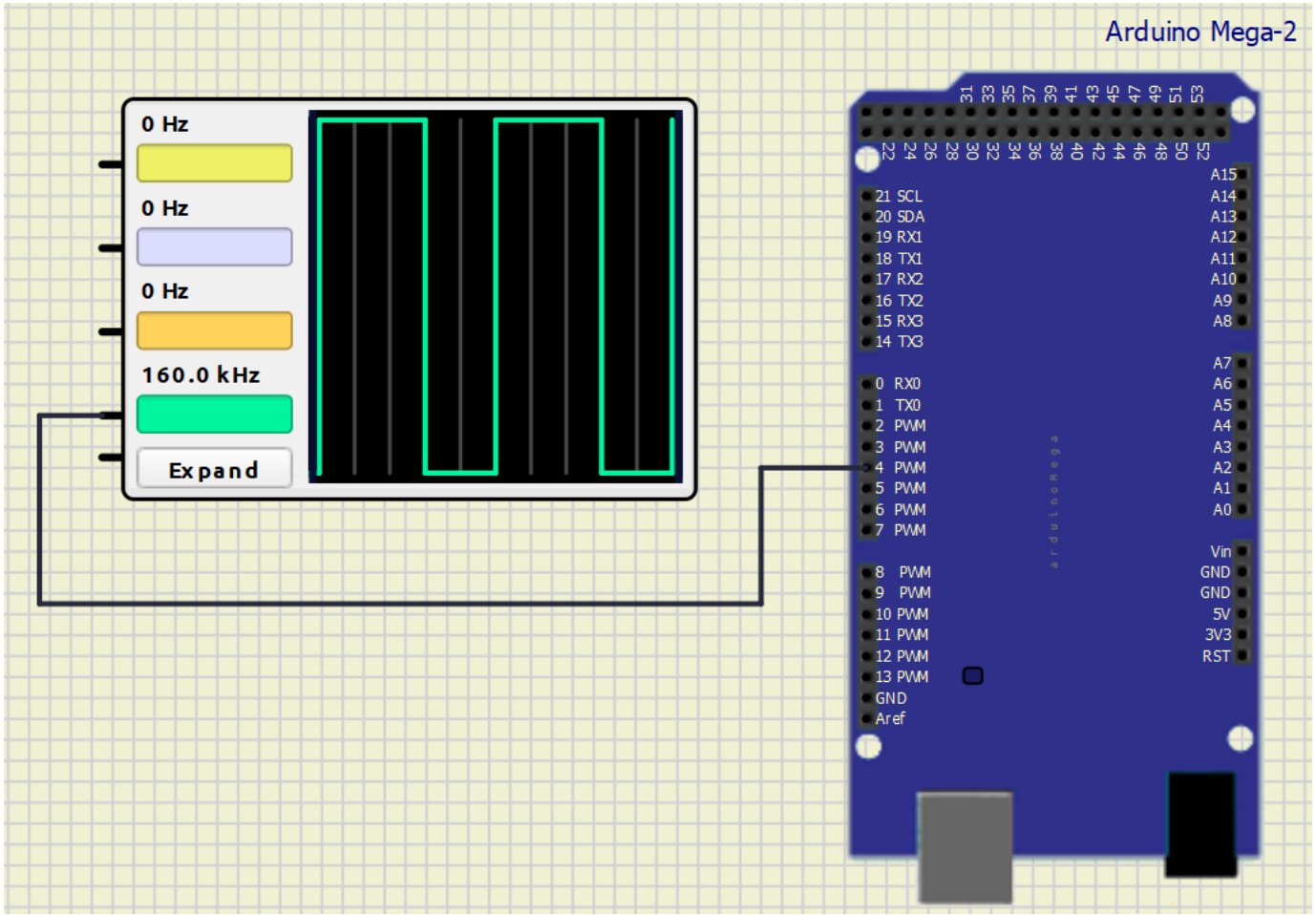
- Generate a PWM waveform with 60% duty. Assume f_{clk} = 16 MHz.
  - d = 0.6*TOP = 153.6, 307.2, 614.4 for TOP=0xFF, 0x1FF, 0x3FF
  - Use Time0 (8-bit)
  - TOP = OCR0A = 99
    - Can't use OC0A for PWM output
  - Fast PWM
    - WGM02:WGM00 = 111
  - fast PWM output using channel OC0B
    - PortG, pin 5
  - OCR0B = 0.6*(99+1)=60
  - Clear when TCNT0=OCR0B
    - COM01:COM00 = 10
  - Prescaler = 1

Code: Fast PWM Duty Cycle Only

```
DDRG |= 0x20;              // configure PTG5 = OC0B pin for output
TCCR0A = 0b00100011;       // COM0A1 COM0A0 COM0B1 COM0B0 -      -     WGM01 WGM02
TCCR0B = 0b00001001;       // FOC0A  FOC0B  -      -      WGM02  CS02  CS01  CS00
TCNT0 = 0;                 // force TCNT0 to count up from 0
OCR0A = 99;                // set TOP = 99 (MAX=100)
OCR0B = 60;                // set duty cycle to 60%
while(1);
```

- What is the frequency of the generated waveform?

Simulation IDE



Fast PWM Setup: Frequency & Duty Cycle

- Given $f_{fastPWM}$ and duty %, calculate TOP and duty cycle count:

$$TOP = \frac{f_{CLK}}{prescaler * f_{fastPWM}} - 1$$

$$d_{count} = (TOP + 1) * d\%$$

- Select prescaler that will produce integer values
- OCRnA = TOP, OCRnB = d_{count}
  - Can't use OCnA for output
  - Helps avoid fraction in multiplication
  - 8-bit: WGMn2:WGMn0 = 7
  - 16-bit WGMn3:WGMn0 = 0xF

Example: Fast PWM Frequency & Duty Cycle

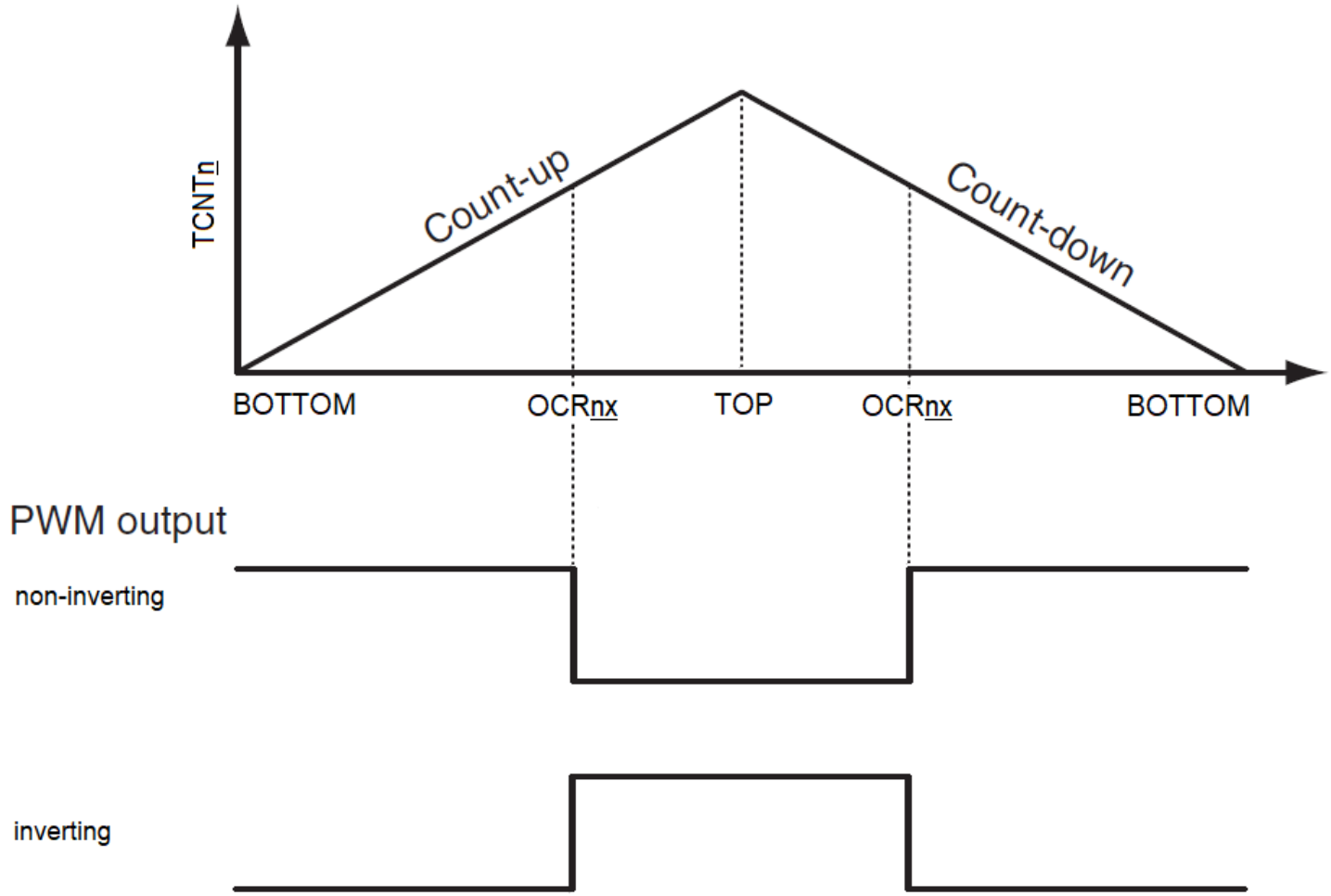
- Generate a 2 kHz square waveform, 40% duty cycle. Assume f_{clk} = 16 MHz.
- Fast PWM Solution:
  - Prescaler = 1, 8, 64, 256, 1024
    - TOP = 7999, 999, 124, 30.25, 6.812
    - dcount = 3200, 400, 50, 12.5 , 3.125
  - Timer0:
    - OCR0A = 124
    - OCR0B = 50
    - Output = OC0B
      - PORTG pin5
    - Prescaler = 64: CS02, CS01, CS00 = 0,1,1
    - WGM02, WGM01, WGM00 = 1,1,1

Code: Fast PWM Frequency & Duty Cycle

```
DDRG |= 0x20;              // configure PTG5 = OC0B pin for output
TCCR0A = 0b00100011;       // COM0A1 COM0A0 COM0B1 COM0B0 -      -      WGM01 WGM02
TCCR0B = 0b00001011;       // FOC0A  FOC0B  -      -      WGM02  CS02  CS01  CS00
TCNT0 = 0;                 // force TCNT0 to count up from 0
OCR0A = 124;               // set TOP = 124 (MAX=125)
OCR0B = 50;                // set duty cycle to 40% (60/125)
while(1);
```

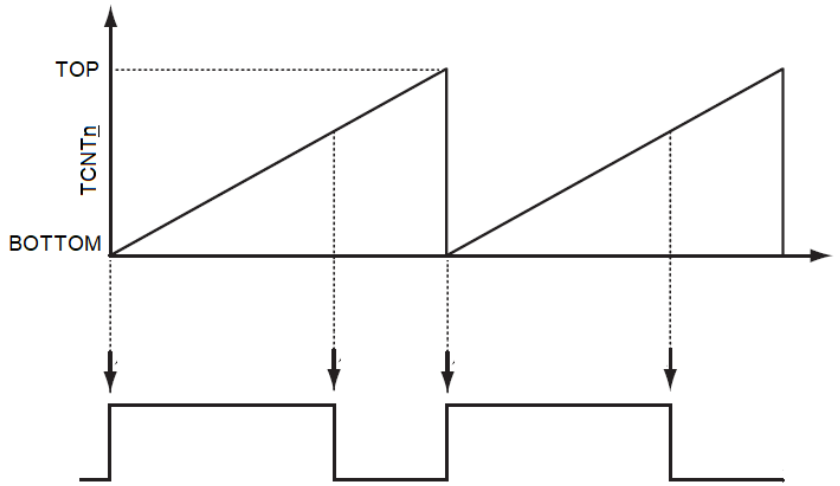
Phase Correct PWM

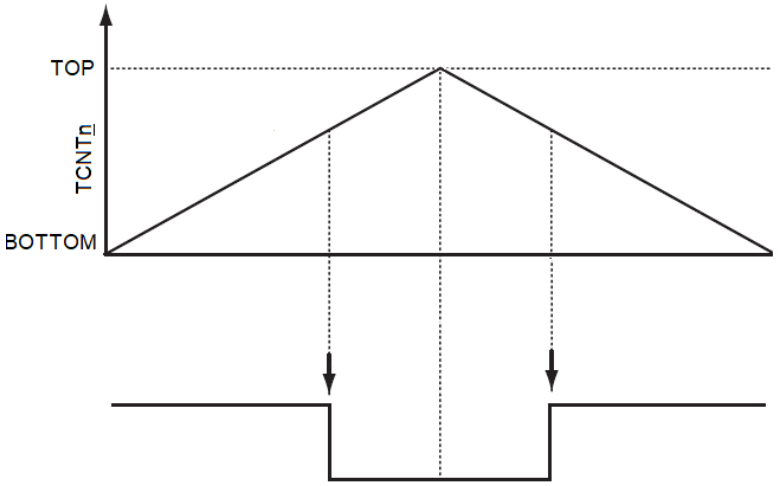
$$f_{PCPWM} = \frac{f_{CLK}}{2 * prescaler * TOP}$$



PC vs Fast

- Fast PWM
  - Uses 1 count to generate the signals
  - has higher frequency
  - Output controlled by OCR and BOTTOM (TOP+1)
- PC PWM
  - aka centered PWM
  - More accurate duty cycle
    - Uses 2 counts to generate the signal
    - Output controlled by OCR only
    - Doesn't need (TOP+1)
  - Has less harmonics





PC PWM

- TCNTn counts BOTTOM to TOP to BOTTOM, repeats
  - Output: OCnA, OCnB, OCnC
- 8-bit PC PWM: Timer n: n=0,2
  - BOTTOM: 0
  - TOP: 0xFF, OCRnA
- 16-bit PC PWM: Timer n: n=1,3,4,5
  - BOTTOM: 0
  - TOP: 0x00FF, 0x01FF, 0x3FF, ICRn, OCRnA
- Can't use OCnA when TOP=OCRnA
- Can't use ICn when TOP=ICRn

Examples PC PWM

- Generate 5kHz centered PWM waveform with 75% duty cycle, f_{clk}=16 MHz.
- PC PWM solution:
  - TOP = fclk/(2*prescaler)
  - Prescaler = 1, 8, 64, 256,
    - TOP = 1600, 200, 26, 6.25,
    - dcount = 1200, 150, 18.75, 4.6875,
  - Timer0:
    - OCR0A = 200
    - OCR0B = 150
    - Output = OC0B
      - PORTG pin2
    - Prescaler = 8: CS02, CS01, CS00 = 0,1,0
    - WGM02, WGM01, WGM00 = 1,0,1

Code PC PWM

```
DDRG |= 0x20;              // configure PTG5 = OC0B pin for output
TCCR0A = 0b00100001;       // COM0A1 COM0A0 COM0B1 COM0B0 -      -      WGM01 WGM02
                           // ouput=OC0B, WGM3:0=101 for PC PWM
TCCR0B = 0b00001010;       // FOC0A  FOC0B  -      -      WGM02  CS02  CS01  CS00
                           // CS02:1=010 prescaler=8
TCNT0 = 0;                 // force TCNT0 to count up from 0
OCR0A = 200;               // set TOP = 200 (MAX=200)
OCR0B = 150;               // set duty cycle to 75% (150/200)
while(1);
```

Phase and Frequency Correct (PFC)PWM

- Dual slope PWM
  - Similar to PC PWM with some difference
- PFC: available only in 16-bit Timers (1,3,4,5)
  - PC available in 8/16-bit
- 2 modes of PFC compared to 5 modes of PC
  - PFC: TOP = OCnA, ICRn
  - PC: TOP = 0xFF, 0x1FF, 0x3FF, OCnA, ICRn
- PFC: changing phase will only take effect at end of cycle
  - Ie when TCNTn= BOTTOM
  - This will maintain preserve frequency
  - PC: changing phases effective at half cycle (TCNTn=TOP)
    - Can make the first waveform appear to have the wrong frequency