ELEC 3040/3050 Lab #7

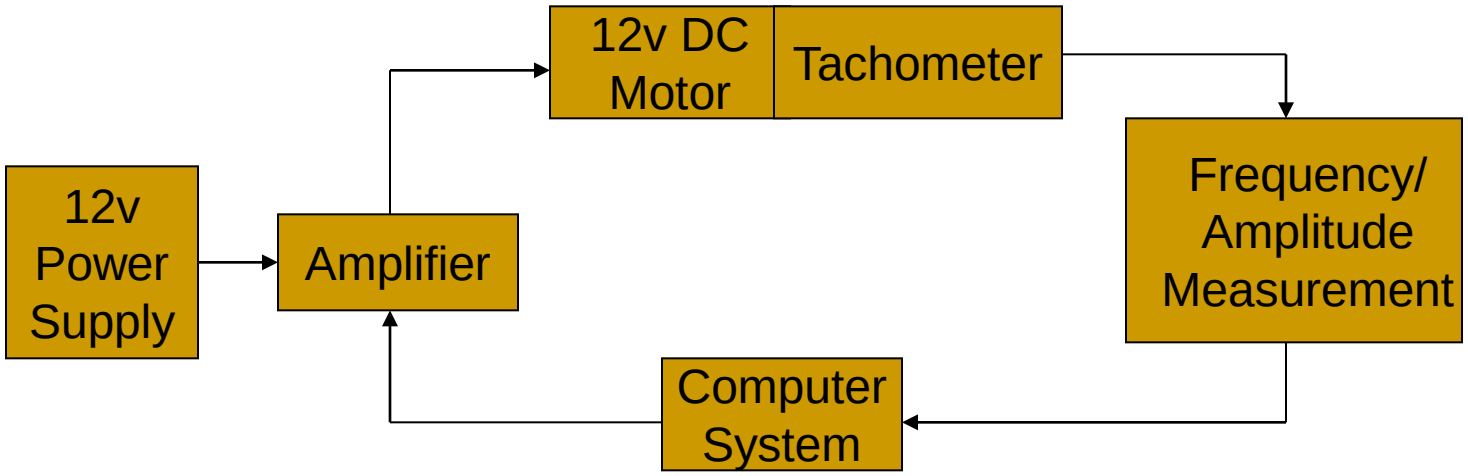
PWM Waveform Generation

Goals of this lab exercise

- Continue the design of the semester project: a speed controller for a D.C. motor
- Generate a pulse-width-modulated (PWM) waveform with switch-selectable duty cycle
  - Two approaches: using two timer functions
- The generated waveform will be amplified in a the next lab to drive a D.C. motor

Motor Speed Control Project

1. Generate a PWM waveform
2. Amplify the waveform to drive the motor
3. Measure motor speed
4. Measure motor parameters
5. Control speed with a PID controller

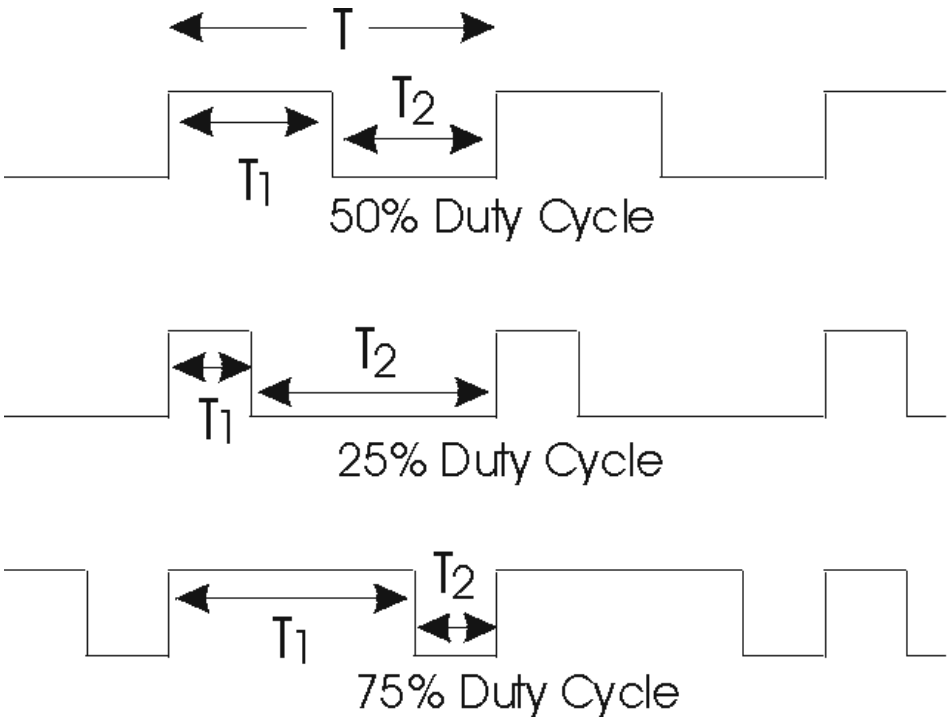


PWM Digital Waveforms

- A “pulse-width modulated” (PWM) waveform is a periodic signal comprising pulses of fixed duration
- “Modulation” refers to setting the pulse width (with period held constant) to achieve a desired effect
- PWM signals are often used to drive D.C. motors, commercial lights, etc.

PWM Waveform Parameters

T = *period* of waveform (constant)
T1 = duration of pulse
T2 = T – T1
Duty Cycle = T1/T = T1/(T1+T2)



Note: Pulses may also be active-low.

Waveform generation with the HCS12

- Pulse-width modulator
  - Program period and duty cycle registers
  - Signal produced on pin PWMx (Port P or Port T)
    - x = 0-5
- Timer module
  - Use “output compare” mode of operation
  - Write signal change times to “input capture/output compare” register
  - Generated signal on pin IOCx (Port T)
    - x = 0-7

Timer Output Compare Function

- Trigger an “event” when TCNT matches the value in register TCn (n = 0, 1, … 7)
  - TCn = timer input capture/output compare reg. n
  - “Compare Flag” CFn in TFLG1 reg. sets on match
    - Clear CFn by writing 1 to it in TFLG1
  - “Event” can be an interrupt
    - Enable by setting CnI = 1 in register TMSK1
  - “Event” can be output pin PTn set/clear/toggle
    - Defined by output mode/level select bits in registers TCTL1/TCTL2

Timer output compare hardware.

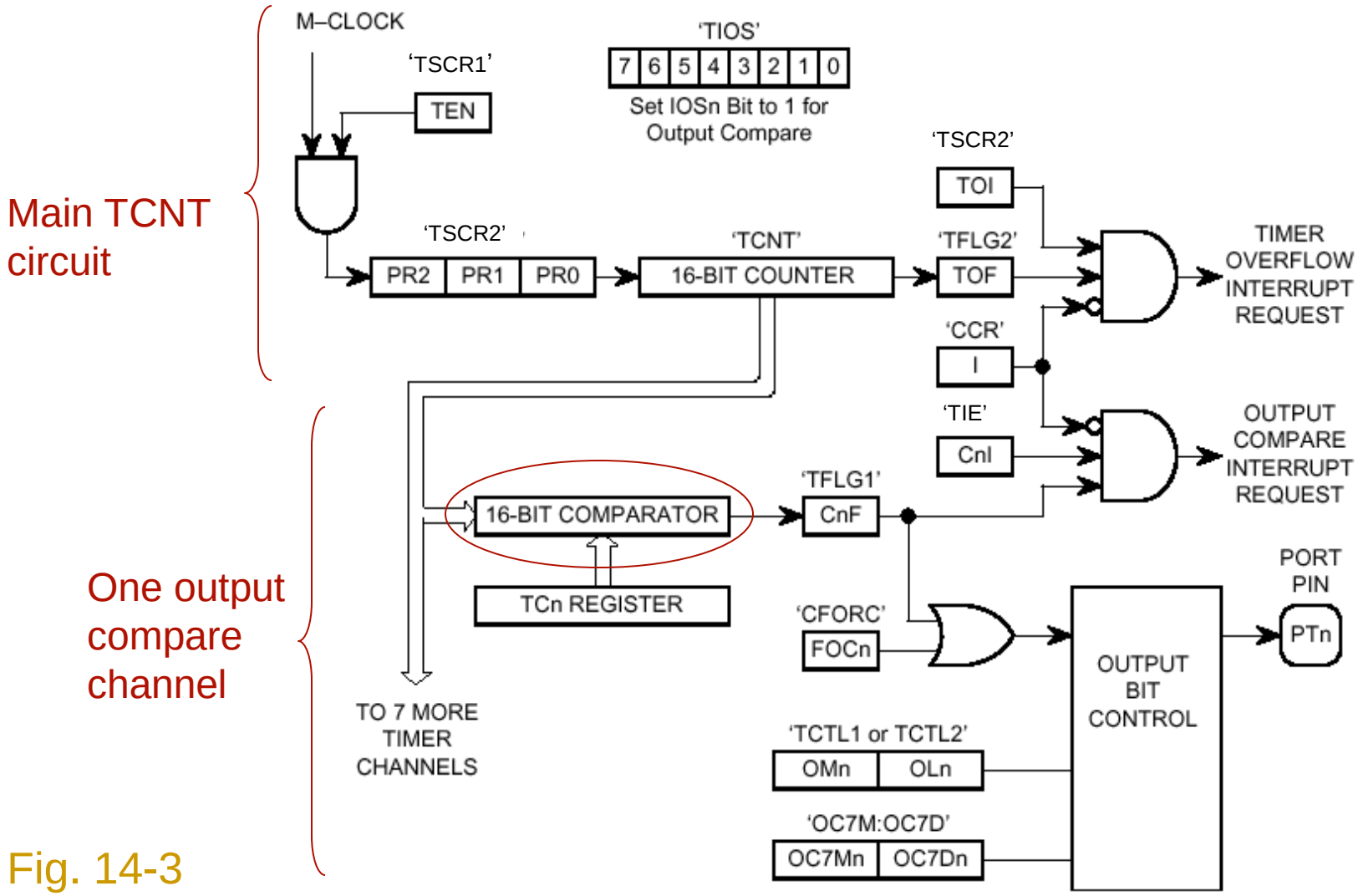


Fig. 14-3

Timer Control Registers 1 and 2 (TCTL1,TCTL2)

Pin PTn action on output compare match

	7	6	5	4	3	2	1	0
TCTL1 $0048	OM7	OL7	OM6	OL6	OM5	OL5	OM4	OL4
TCTL2 $0049	OM3	OL3	OM2	OL2	OM1	OL1	OM0	OL0

OMn:OLn(OM = Output Mode, OL = Output Level)

- 00 : disconnect timer from PTn (no pin action)
- 01 : Toggle PTn state
- 10 : Force PTn to 0
- 11 : Force PTn to 1

Example – generate 1KHz square wave

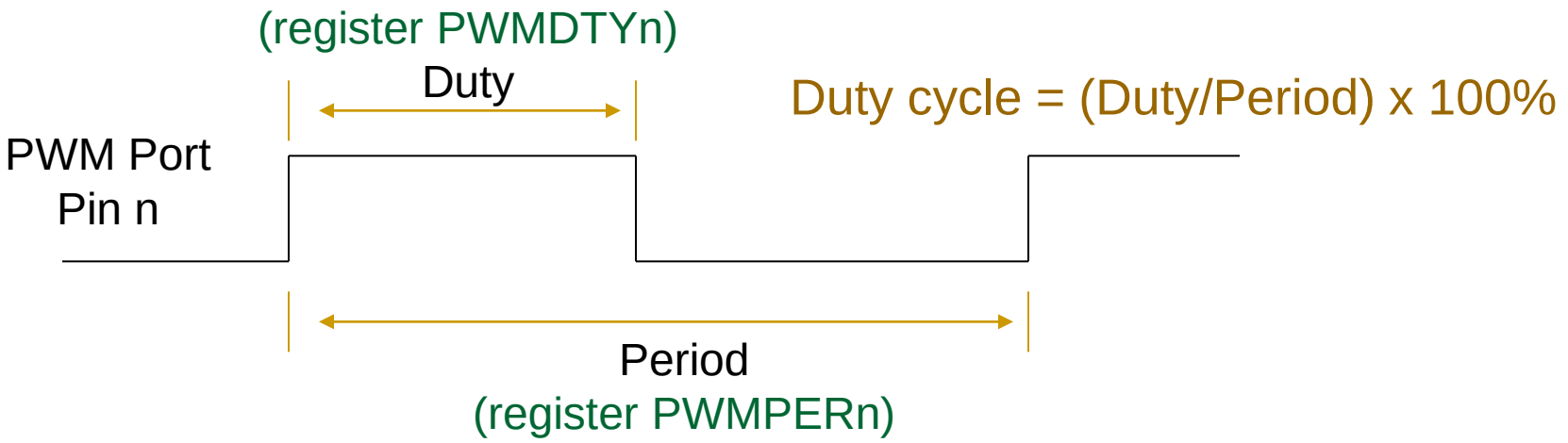
```
; f = 1KHz : T = 1ms = 8000 periods of 8 MHz clock
        bset TSCR1,%10000000   ;enable timer (TEN)
        bset TIOS,%00000010    ;enable OC chan 1
        bclr TCTL2,%00001000   ;toggle PT1 on match (OM1=0)
        bset TCTL2,%00000100   ;toggle PT1 on match (OL1=1)
        ldd  TCNT              ;current TCNT
        addd #4000             ;1/2msec from now
        std  TC1               ;write OC register 1
        movb #%00000010,TFLG1  ;reset CF1
        bset TIE,%00000010     ;CI1=1 to enable channel 1 interrupt
        cli                    ;enable CPU interrupts
        ...do some other processing
; Timer interrupt routine – entered every 1/2ms
isr:    ldd  TCNT              ;current TCNT
        addd #4000             ;next interrupt 1/2ms from now
        std  TC1               ;write OC register 1
        movb #%00000010,TFLG1  ;clear CF1
        rti                     ;return

        ORG  $FFEC    ;Timer channel 1 interrupt vector
        dc.w  isr
```

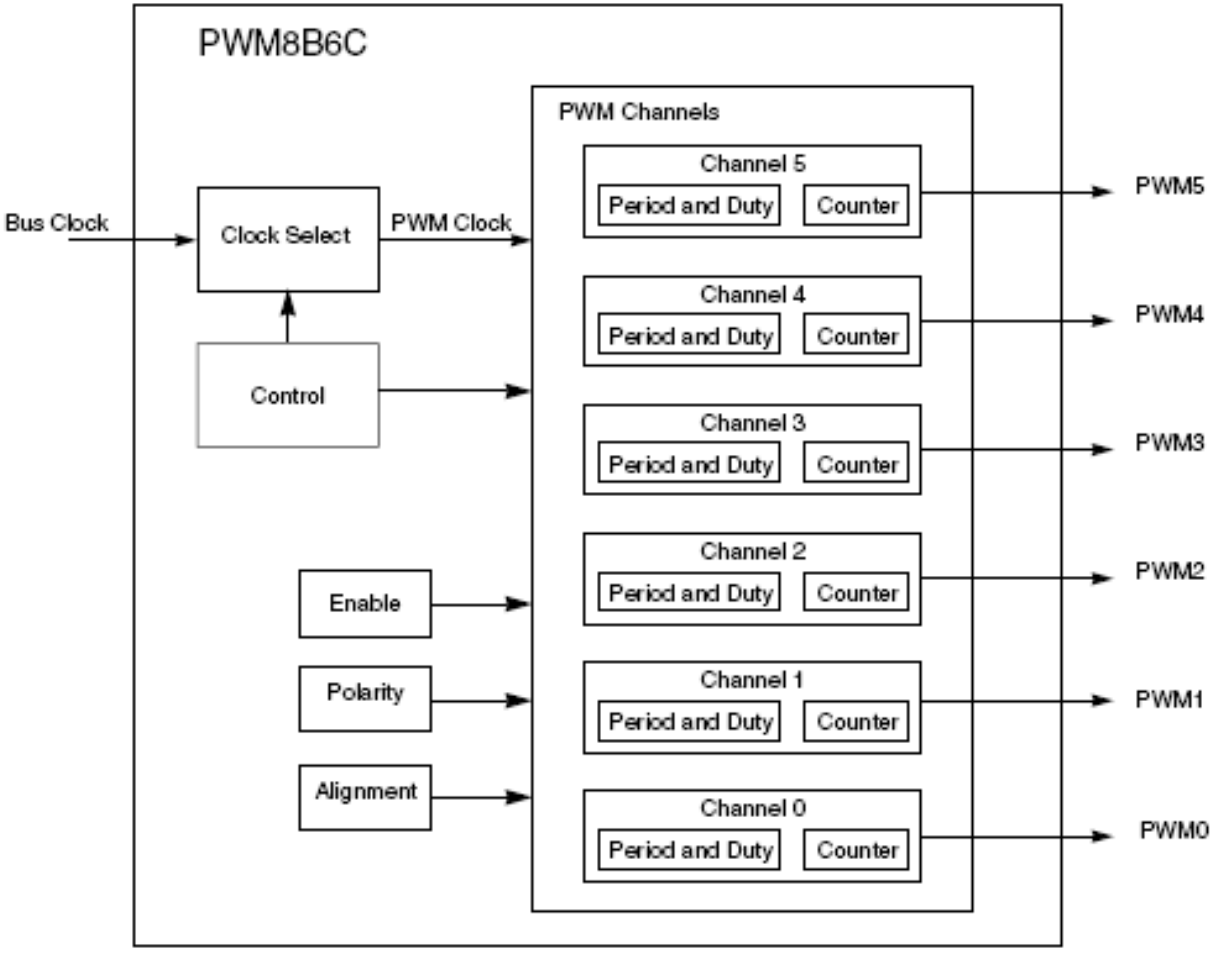
```
/* This program outputs a 250 Hz square wave on
 * Port T, bit-0 using timer channel 0 interrupts and the
 * output compare bit operations.
 ***************************************************************/
#define  HALF_P  16000     /* Number clocks per 1/2 period */
/***************************************************************/
void main(void) {
    TSCR1_TEN = 1;         /* Enable the timer */
    TIOS_IOS0 = 1;         /* Enable output compare channel 0 */
    TC0 = TCNT;            /* Set up TC0 */
  /* Now have ~ 8 ms to set up the system without an  interrupt   */
    TFLG1 = TFLG1_C0F_MASK;   /* Clear C0F flag */
    TIE_C0I = 1;           /* Enable the interrupt */
    TCTL2_OM0 = 0;         /* Set output compare action to toggle Port T, bit-0.  */
    TCTL2_OL0 = 1;
    EnableInterrupts;      /* Unmask interrupts */
    for(;;) {
        asm (wai);         /* Wait for interrupt */
  }
}
/*****************************************************************************
 * The interrupt service routine simply sets up the TC0 for the next interrupt time.
 *****************************************************************************/
void interrupt 8 OC0_isr ( void ){
      TC0 += HALF_P;                  /* Set up TC0 for next event */
      TFLG1 = TFLG1_C0F_MASK;    /* Clear C2F flag */
}
```

Pulse-Width Modulator (PWM)

- Generate up to 6 PWM waveforms
- No further program actions required after PWM module has been initialized
- Program 8 or 16-bit period and duty cycle

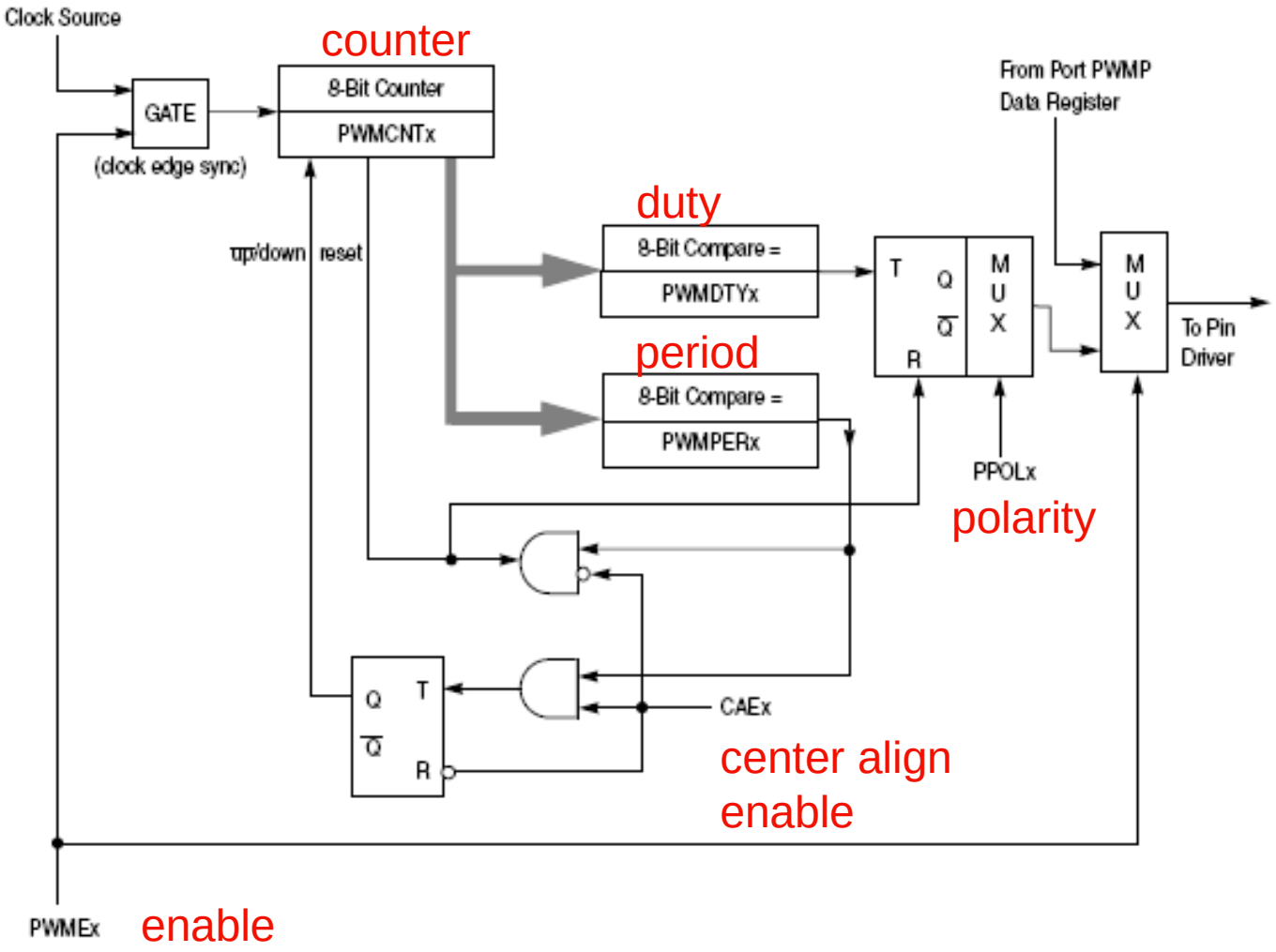


PWM block diagram



To pins in Ports P or T

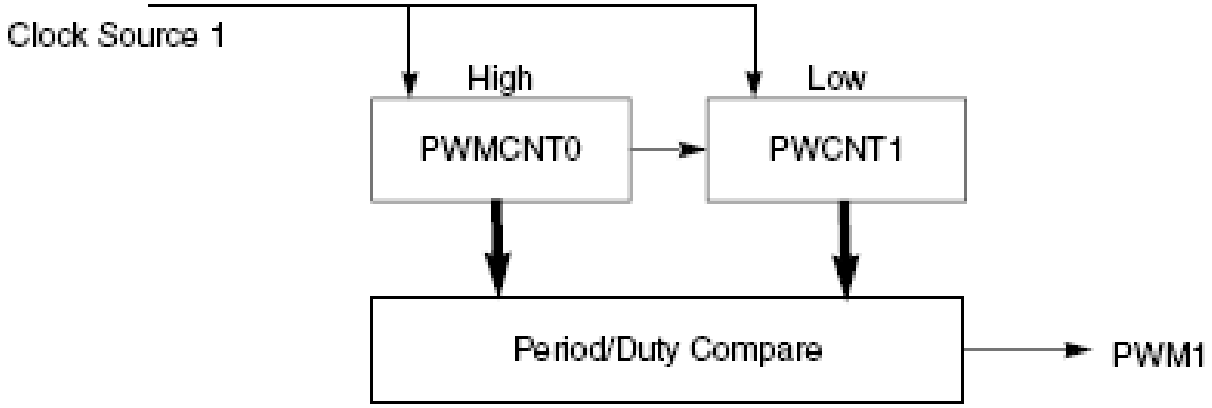
PWM timer channel



PWM period, duty & counter registers

- One set per channel (channel #'s = 0,1,2,3,4,5)
- PWMPERn – 8-bit period for channel n
  - Register addresses $00F2 - $00F7
- PWMDTYn – 8-bit duty for channel n
  - Register addresses $00F8 - $00FD
- PWMCNTn – 8-bit counter for channel n (read-only)
  - Counter driven by selected clock source
  - PWM signal changes when this matches period or duty registers
  - Register addresses $00EC - $00F1

PWM using 16-bit counters



- Two count, period & duty registers concatenated
  - Use 16-bit writes for updates
- Clock/polarity defined by lower-numbered channel
- Output pin from higher-numbered channel
- Enable via bit CON01 in PWMCTL register
  - Similar for channels 2/3 and 4/5 (CON23, CON45)

PWM Concatenate Control Register
(PWMCTL – address $00e5)

7	6	5	4	3	2	1	0
0	CON45	CON23	CON01	PSWAI	PFRZ	0	0

CON45, CON23, CON01:

0 = channels are separate 8-bit PWMs

1 = channels combine to be 16-bit PWMs

- Output pins are PWM5, PWM3, PWM1
- Counter/period/duty registers 4:5, 2:3, 0:1
  - High byte in even # register
- Clock select/polarity/enable set in odd # registers

PWM Enable Register
(PWME – address $00e0)

7	6	5	4	3	2	1	0
0	0	PWME5	PWME4	PWME3	PWME2	PWME1	PWME0

PWMEn: PWM enable n

1 = PWM channel n

- signal on pin PWMn, beginning next clock
- if CON45=1, PWME4 has no effect
  (likewise for CON23, CON01)

0 = PWM channel n disabled

PWM Polarity Register
(PWMPOL – address $00e1)

7	6	5	4	3	2	1	0
0	0	PPOL5	PPOL4	PPOL3	PPOL2	PPOL1	PPOL0

PPOLn:polarity of signal on output pin n

1 =high at beginning of period,
low when duty count reached

0 =low at beginning of period,
high when duty count reached

PWM Center Align Enable Register
(PWMCAE – address $00e4)

7	6	5	4	3	2	1	0
0	0	CAE5	CAE4	CAE3	CAE2	CAE1	CAE0

CAEn: center align enable for channel n

0 = channel n operates in left-aligned mode

1 = channel n operates in center-aligned mode

(period = 2x value in PWMPERn)

PWM waveforms

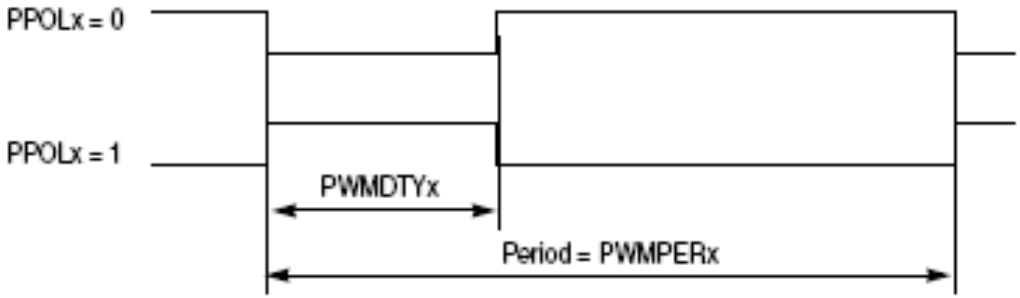


Figure 12-36. PWM Left Aligned Output Waveform

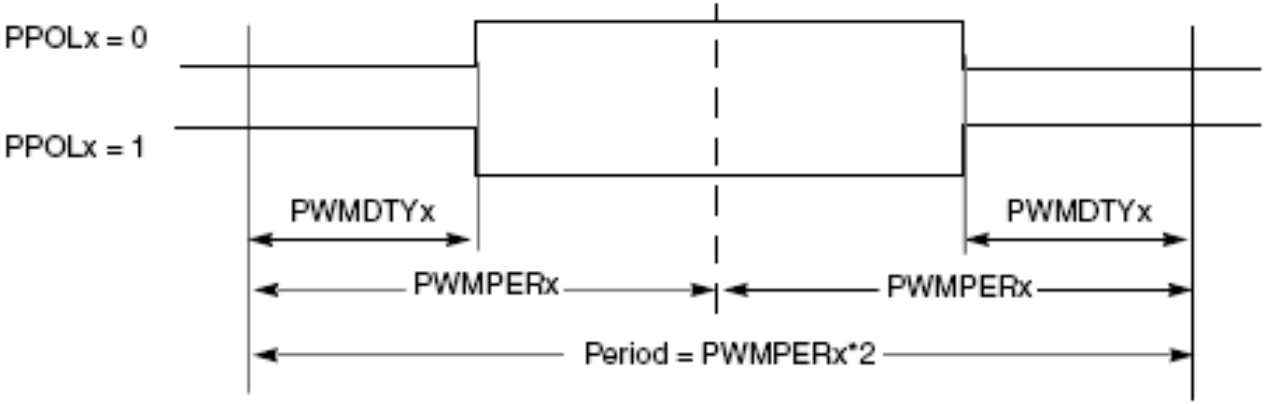


Figure 12-38. PWM Center Aligned Output Waveform

PWM clock sources

- Four clock sources derived from system bus clock: A, B, SA, SB
  - PWM prescale clock register (PWMPRCLK) controls divider stages for clocks A and B
    - Divide bus clock frequency by 1,2,4,8,16,32,64,128
  - Clocks SA/SB are scaled versions of A/B
    - Divide clock A/B frequency by 1…512
  - Select clock source in PWM clock select reg. (PWMCLK)
    - A or SA for pins 0,1,4,5
    - B or SB for pins 2,3

PWM Clock Source Select Register
(PWMCLK – address $00e2)

7	6	5	4	3	2	1	0
0	0	PCLK5	PCLK4	PCLK3	PCLK2	PCLK1	PCLK0

PCLKn: select clock source for channel n

(2 available for each channel)

Channels 0,1,4,5:

0 = select clock A

1 = select clock SA

Channels 2,3:

0 = select clock B

1 = select clock SB

PWM Prescale Clock Select Register
(PWMPRCLK – address $00e3)

7	6	5	4	3	2	1	0
0	PCKB2	PCKB1	PCKB0	0	PCKA2	PCKA1	PCKA0

PCKA2-0/PCKB2-0:prescalers for clocks A & B

000 – bus clock (default)
001 – bus clock/2
010 – bus clock/4
011 – bus clock/8
100 – bus clock/16
101 – bus clock/32
110 – bus clock/64
111 – bus clock/128

PWM Scale Registers

- Scaling value for clocks A and B to produce SA and SB clocks, respectively
  - Freq. of clock SA = freq. of clock A / (2 x scale)
  - Freq. of clock SB = freq. of clock B / (2 x scale)
- Scale value from 8-bit clock scale registers:
  - PWMSCLA ($00e8) for SA
  - PWMSCLB ($00e9) for SB
  - Scale value 0 treated as 512

Example – 20KHz PWM signal with 20% duty cycle on channel 1

- Assume bus clock = 8MHz (T = .125us)
  - Period = 8MHz/20KHz = 400
  - Duty = 400 x 20% = 80
    - 400 exceeds max value of period register (255), so prescale bus clock by 2
    - Period = 400/2 = 200, Duty = 80/2 = 40
    - Use clock A for pin 1
    - Select polarity = 1 (high during Duty)

Continued on next slide

Example – 20KHz PWM signal with 20% duty cycle on channel 1 (continued)

- Register values:
  - PWMCLK = 0x00 (PCLK1 = 0 to select clock A)**
  - PWMPRCLK = 0x01 (PCKA = 1 for prescale = 2)
  - PWMSCLA = *don't care*, since using A and not SA
  - PWMPOL = 0x02 (PPOL1 = 1 to start high)
  - PWMCAE = 0x00 (CAE1 = 0 for left-aligned)**
  - PWMPER1 = 200 (scaled period)
  - PWMDTY1 = 40 (scaled duty)
  - PWME = 0x02 (PWME1 = 1 to enable channel 1)

** default values do not need to be written

PWM output pins

- Default: PWM outputs in Port P
  - PWMk = bit k of Port P
- PWM output pins can be redirected to Port T
  - Select via Port T Module Routing Register (MODRR, address $0247)
  - MODRR bit k controls bit k of Port T
    - k=0 to connect pin to Port T timer module (default)
    - k=1 to connect pin to PWM module (PWMk)
- Dragonfly Plus:
  - PP5/KPW5 only Port P connection available
  - Switch S2 on the module must be in the RUN position to disconnect from a pull-down resistor on PP5

Lab Procedure

- Program 1: Generate a PWM waveform using the dedicated PWM module
  - First, generate a waveform with one duty cycle value
  - Then, verify that you can generate waveforms with each of the 10 specified duty cycles, as selected by the keypad
    - Measure and record the 10 duty cycle values
    - Plot measured duty cycle vs. switch setting
- Program 2: Repeat using the timer output compare function