HCS12 Timers

Textbook - Chapter 14

Timer usage in computer systems

- Periodically interrupt CPU to perform tasks
  - Sample temperature/pressure readings
  - Generate music samples
- Provide accurate time delays
  - Instead of software loops
- Generate pulses or periodic waveforms
  - PWM signal for motor control
  - Strobe pulse for an external device
- Determine time of an external event

Timing functions in the HCS12

- Real-time interrupt (RTI)
  - Derived from main oscillator
  - Periodically set flag and trigger interrupt
- Timer module
  - 8 channels (TC0-TC7) of output compare/input capture functions
  - Work with Port T pins
- Pulse-width modulator (PWM)
  - Generate up to 6 PWM waveforms on Port P pins

HCS12 Basic Timer Module

(Chapter 14.2)

- Based on 16-bit free-running timer (TCNT)
- 8 channels - each uses/controls one I/O pin
  - Pins in Port T (PT0, PT1, …, PT7)
  - Input capture: capture time of PTn change
  - Output compare: trigger event at designated time
- 16-bit pulse accumulator
  - Count pulses on an input pin

Main timer system block diagram.

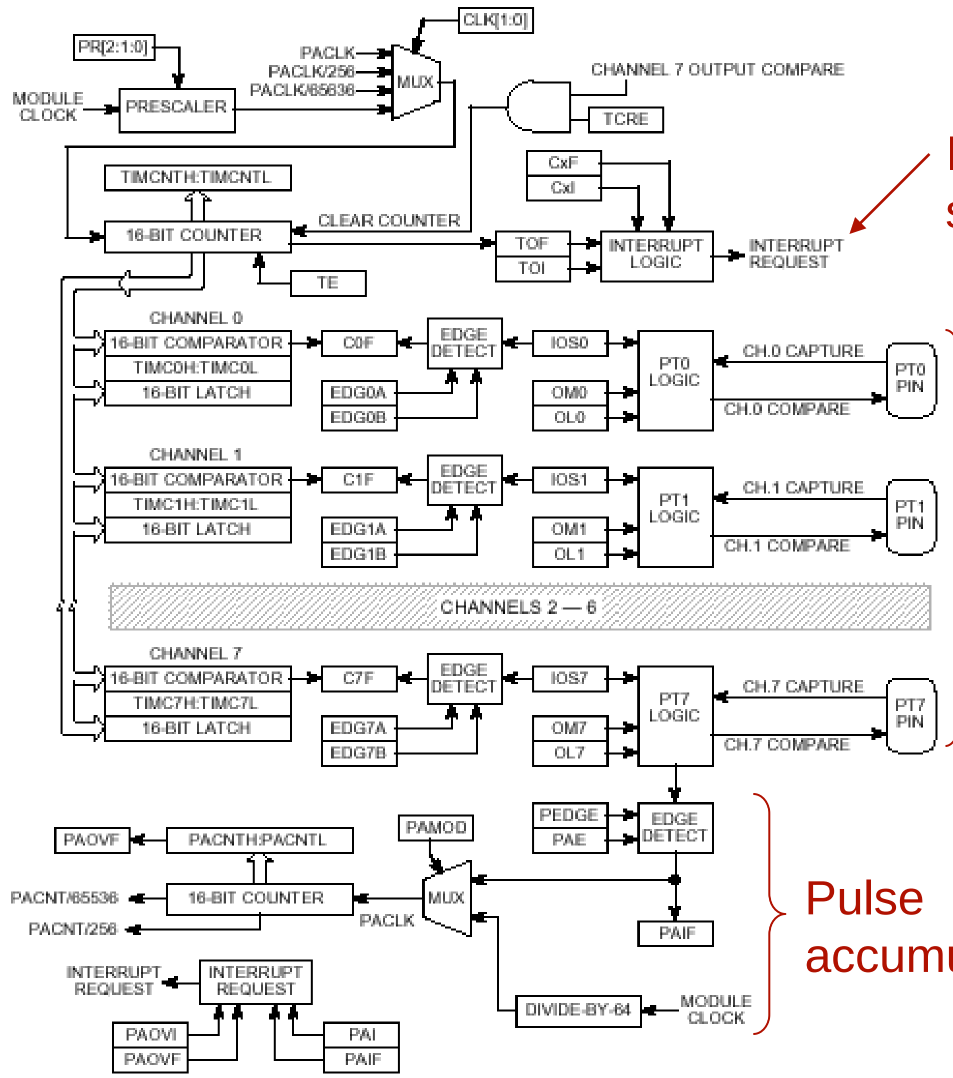
Timer count register & prescaler

8 channels of input capture & output compare

Interrupt signal

I/O Pins

Pulse accumulator



TCNT 16-bit free-running counter

- TCNT resets to 0000 and runs (increments) continuously
  - Read TCNT at any time (address $0044, $0045)
  - TCNT cannot otherwise be set
- TCNT clocked by system “bus” clock
  - Bus clock = 1/2 oscillator frequency
  - Bus clock “prescaled” by programmable divider
    - divide by factor of 1, 2, 4, 8, 16, 32, 64, 128
- Timer overflow flag sets when TCNT rolls over from $FFFF to $0000
  - Triggers timer overflow interrupt, if enabled

TCNT free-running counter

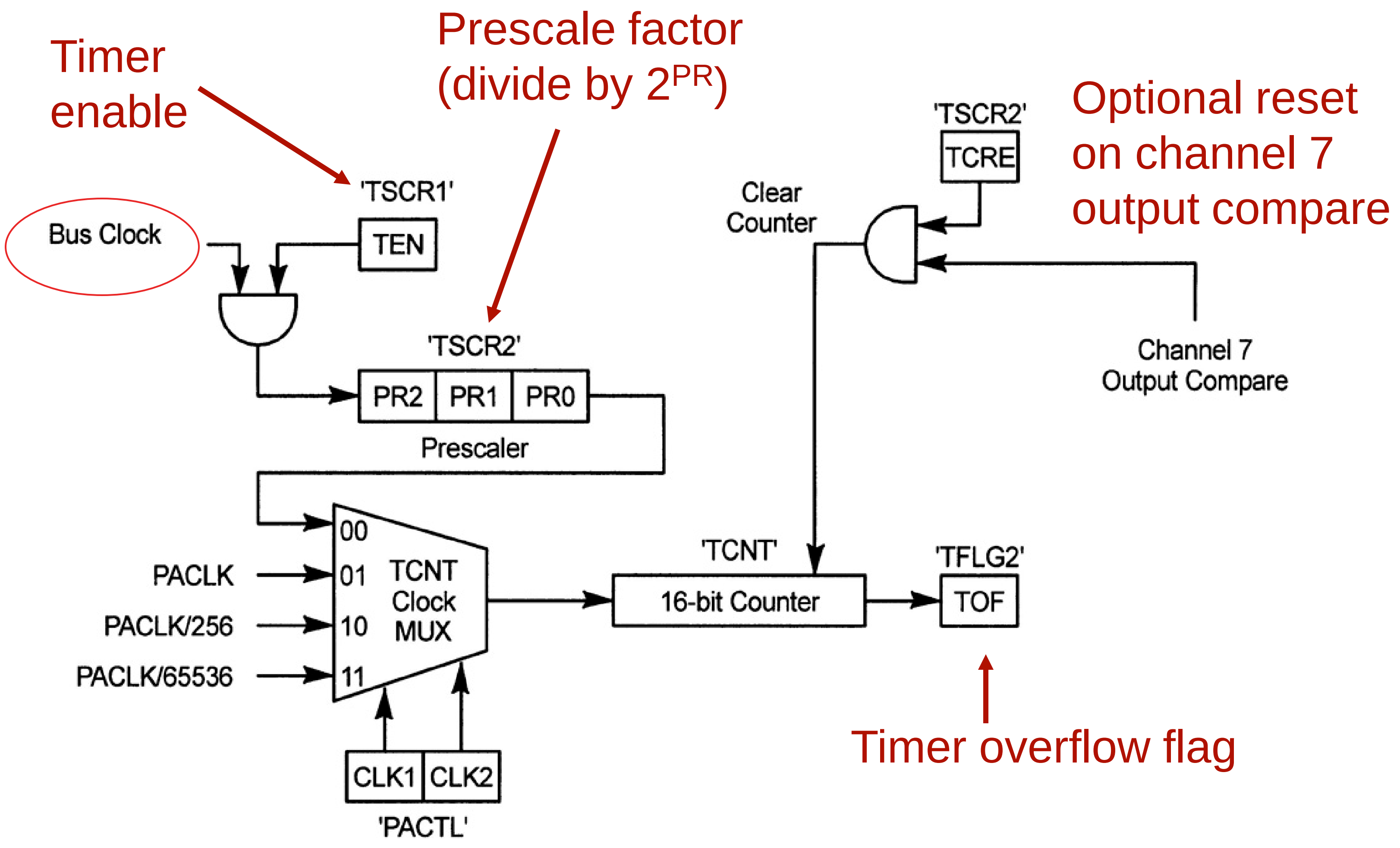


Figure 14-1 TCNT hardware.

Timer overflow hardware

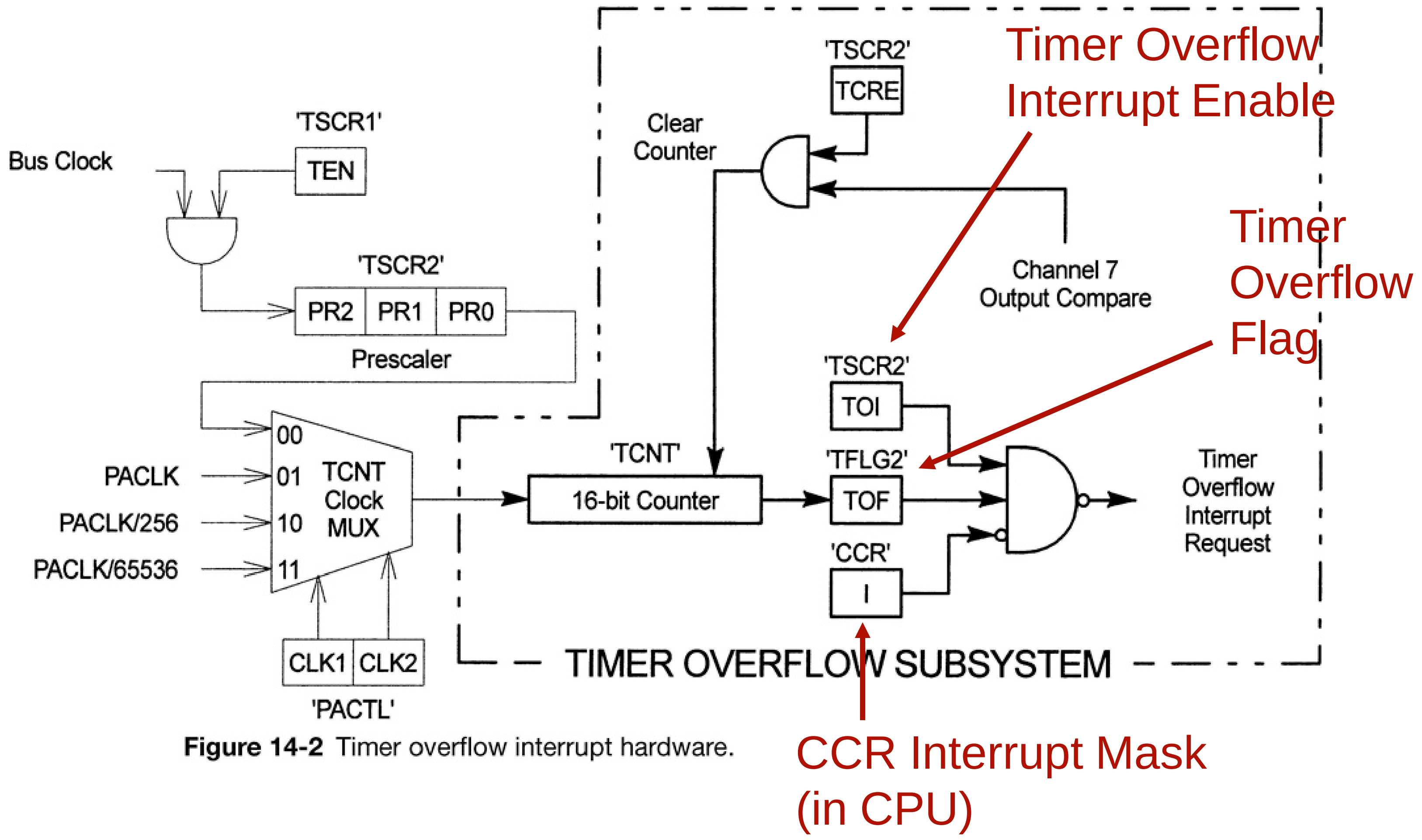


Figure 14-2 Timer overflow interrupt hardware.

Timer registers

- Timer Count Register (TCNT)
  - Read with 16-bit load (LDX, LDY, LDD)
  - *Don’t use 8-bit loads (LDAA, LDAB)*
- Timer System Control Register 1 (TSCR1)
  - Enables bus clock to increment TCNT
- Timer System Control Register 2 (TSCR2)
  - Enables interrupt on TCNT overflow
- Timer Interrupt Flag Register 2 (TFLG2)
  - Flag indicates TCNT overflow

Timer System Control Register 1 (TSCR1)

TSCR1 address: $0046

7	6	5	4	3	2	1	0
TEN	TSWAI	TSFRZ	TFFCA	0	0	0	0

Timer fast flag clear all

Timer stops in background mode
  0 = continue running (default)

Timer stops while in wait
  0 = continue running (default)

Timer enable
  0 disables TNCT (default)
  1 enables the timer

Timer System Control Register 2 (TSCR2)

TSCR2 address: $004D

7	6	5	4	3	2	1	0
TOI	0	TPU	TDRB	TCRE	PR2	PR1	PR0

Timer Overflow Interrupt Enable

1 =enable interrupt when TOF set
0 = inhibit interrupt

Timer Pull-up Resistor Enable

Timer Drive Reduction

Timer Counter Reset Enable

1 = reset to 0000 on output compare on channel 7
0 = TNCT runs free

Prescale Factor

000	1
001	2
010	4
011	8
100	16
101	32
110	64
111	128

Timer Interrupt Flag 2 (TFLG2)

TFLG2 address: $004F

7	6	5	4	3	2	1	0
TOF	0	0	0	0	0	0	0

Timer Overflow Flag

-Set when TCNT overflows from FFFF to 0000

(interrupt if TOI bit set in TMSK2)

-Reset by writing 1 to TOF bit

Ex. bset TFLG2,%10000000

Timer overflow interrupt vector at $FFDE:$FFDF

Example 1: Create a 1 msec pulse on PA0

```
; Force PA0=1, wait 1 msec, then force PA0=0
; Assume M = 8MHz clock: T=.125µs
; 1 msec/.125µs = 8000 counts
    movb #%10000000,TFLG2     ;clear TOF
    movb #%10000000,TSCR1     ;enable timer (TEN)
    bset PORTA,#1             ;start pulse
    ldd  TCNT                 ;current TCNT
    addd #8000                ;1msec from now
S1: cpd  TCNT                 ;target TCNT yet?
    bne  S1                   ;wait for match
    bclr PORTA,#1             ;end pulse
```

;More accurate than “do nothing loops”.

Example 2 – Delay 1 second by checking TCNT overflows (long delay)

```
; Delay 1 second
; Assume 8MHz bus clock => T=.125μs
; Overflow every (65536 counts)x(.125μs)=8.192ms
; 1 second/8.192ms ≈ 122 overflows
    movb #%10000000,TFLG2           ;clear TOF
    movb #%10000000,TSCR1           ;enable timer (TEN)
    movb #122,count                 ;count 122 O/Fs
S1: brclr TFLG2,%10000000,S1        ;wait until TOF=1
    movb #%10000000,TFLG2           ;clear TOF
    dec  count
    bne  S1                         ;repeat
```

Example 3 – 1 sec delay with interrupts

```
; From before, 1 second = 122 timer overflows
        clr  count
        movb #%10000000,TFLG2    ;clear TOF
        bset  TSCR1,%10000000    ;enable timer (TEN)
        bset  TMSK2,%10000000    ;enable interrupt (TOI)
        cli                       ;enable CPU interrupts
start:  wai                       ;wait for interrupt
        ldaa count                ;count changed by isr
        cmpa #122                 ;122 = 1 second
        bne  start                ;repeat until 122

; Timer interrupt routine – entered 122 times/second
isr:  inc  count                ;count interrupts
      movb #%10000000,TFLG2    ;clear TOF
      rti                       ;return to main prog

      ORG  $FFDE   ;Timer overflow interrupt vector
      dc.w  isr
```

Timer Output Compare Function (Chapter 14.4)

- 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

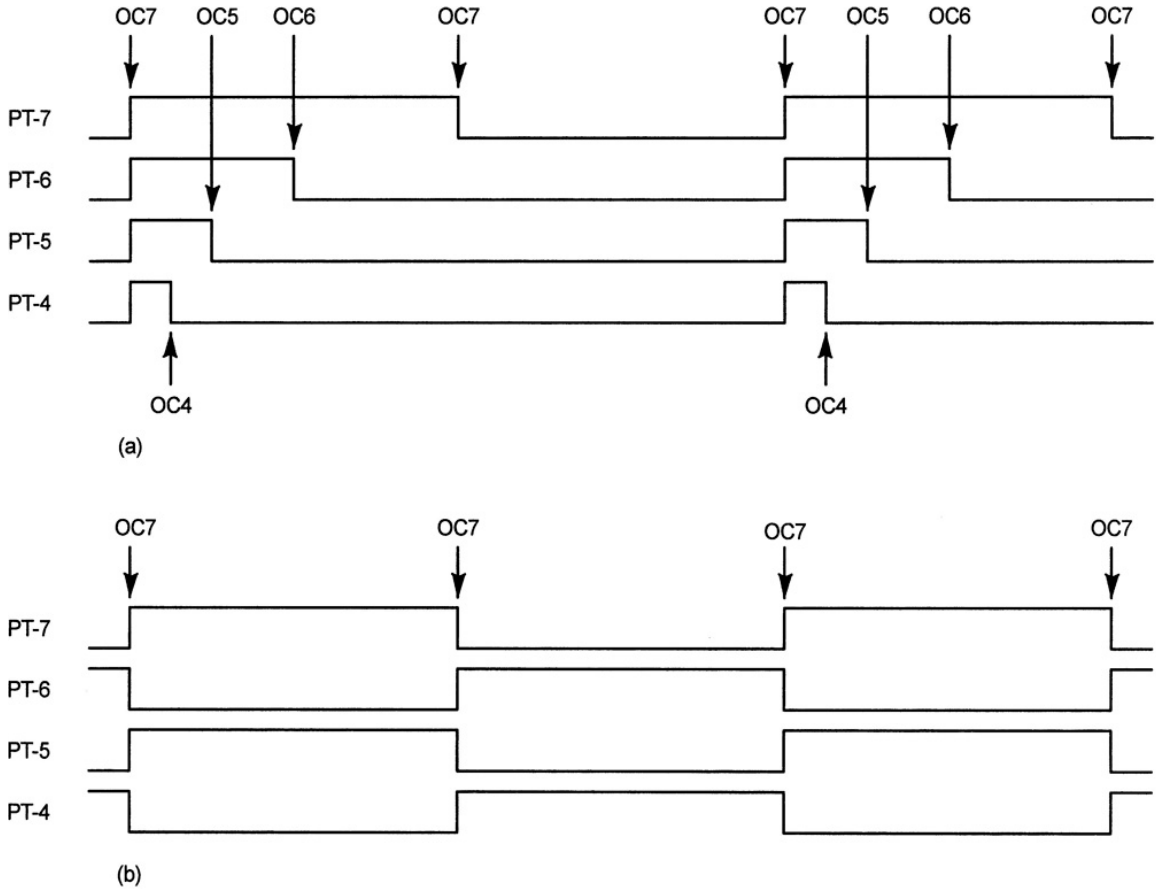


Figure 14-4 Output Compare 7 controlling 4 bits each with (a) shorter comparison times and (b) equal comparison times.

Timer input capture/output compare registers TC0 – TC7

- TC0 – TC7 are 16 bit registers

  TC0 = $0050,51

  TC1 = $0052,53

  ....

  TC7 = $005E,5F
- Output compare mode:
  - TCn compared to TCNT, with "event" on match
- Input capture mode:
  - TCNT copied to TCn at time of a specified edge on pin PTn

Timer output compare hardware

Main TCNT circuit

One output compare channel

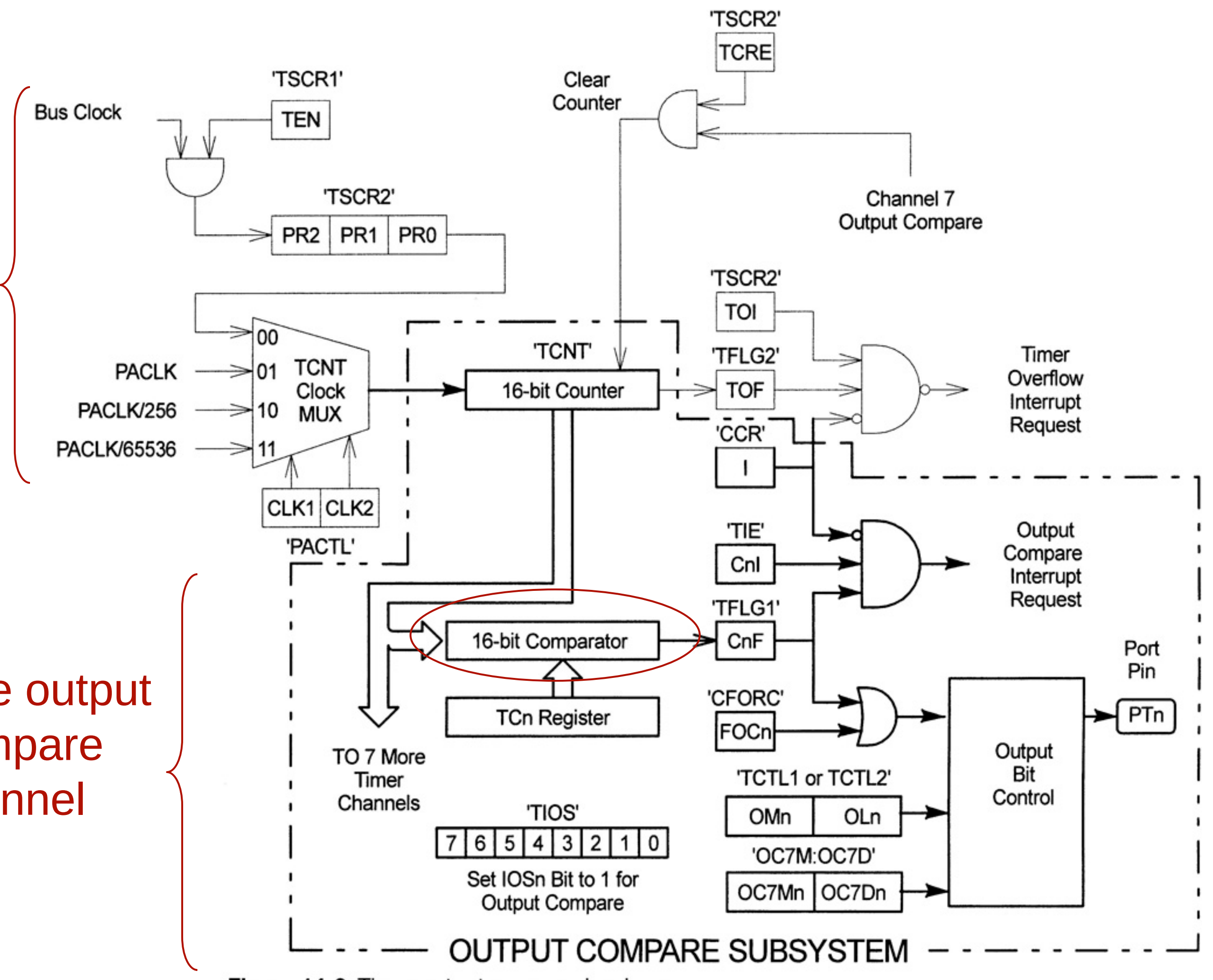


Figure 14-3 Timer output compare hardware.

Timer Input Capture/Output Compare Select Register (TIOS)

TIOS address: $0040

7	6	5	4	3	2	1	0
IOS7	IOS6	IOS5	IOS4	IOS3	IOS2	IOS1	IOS0

IOSn: Select input capture or output compare mode for channel n

0 : channel acts as <u>input</u> capture, or GPIO for pin PTn (DDRT sets direction)

1 : channel n acts as <u>output</u> compare

Timer Interrupt Flag Register 1 (TFLG1)

TFLG1 address: $008E

7	6	5	4	3	2	1	0
CF7	CF6	CF5	CF4	CF3	CF2	CF1	CF0

CFn: Timer interrupt flag n

Output Compare: CFn set when TCNT = TCn

Input Capture: CFn set when selected edge occurs on pin PTn

Reset CFn by writing 1 to it **

*** don't use "bset" – see next slide*

Do not use "bset" to reset flags

- Example: Reset flag CF7 of TFLG1
  - Assume TFLG1 = 10000001
  - Incorrect: *bset TFLG1,%10000000*

    TFLG1 is read (10000001), bit 7 is set, and result 10000001 is written back to TFLG1, resetting both CF7 and CF0.
  - Correct: *movb #%10000000,TFLG1*

    or: *ldaa #%10000000*

    staa TFLG1

Timer Interrupt Enable Register (TIE)

TIE address: $004C

7	6	5	4	3	2	1	0
C7I	C6I	C5I	C4I	C3I	C2I	C1I	C0I

CnI: Timer interrupt enable n

1 = enable interrupt when CFn set in TFLG1

0 = disable ….

Individual timer channel interrupt vectors

Vector address	Interrupt source	Local enable
$FFE0,FFE1	Timer channel 7	TIE/C7I
$FFE2,FFE3	Timer channel 6	TIE/C6I
$FFE4,FFE5	Timer channel 5	TIE/C5I
$FFE6,FFE7	Timer channel 4	TIE/C4I
$FFE8,FFE9	Timer channel 3	TIE/C3I
$FFEA,FFEB	Timer channel 2	TIE/C2I
$FFEC,FFED	Timer channel 1	TIE/C1I
$FFEE,FFEF	Timer channel 0	TIE/C0I

Example 4 – do periodic task every 1 ms

```
; From before, 1ms = 8000 periods of 8 MHz clock
       bset TSCR1,%10000000   ;enable timer
       bset TIOS,%00000010    ;enable OC chan 1
       ldd  TCNT              ;current TCNT
       addd #8000             ;1msec from now
       std  TC1               ;write OC register 1
       movb #%00000010,TFLG1  ;reset CF1
       bset TIE,%00000010     ;set CI1 to enable ch. 1 interrupt
       cli                    ;enable CPU interrupts
       ...  ;do some other processing

; Timer interrupt routine – entered every 1ms
isr:   ldd  TCNT              ;current TCNT
       addd #8000             ;1msec from now
       std  TC1               ;write OC register 1
       movb #%00000010,TFLG1  ;clear CF1
       ... DO THE TASK
       rti                       ;return to main prog

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

Timer Control Registers 1 & 2 (TCTL1, TCTL2)

Control output pin PTn action on successful output comparison

	7	6	5	4	3	2	1	0
TCTL1 $0088	OM7	OL7	OM6	OL6	OM5	OL5	OM4	OL4
TCTL2 $0089	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

Timer Compare Force Register (CFORC = $0081)

7	6	5	4	3	2	1	0
7	6	5	4	3	2	1	0

Force output compare action for channels 0-7

* Write 1 to bit k to cause action programmed by OMk:OLk.

<u>Example:</u>

```
movb #%10000000,TCTL1    ;PT7=0 on channel 7 compare
movb #%10000000,CFORC  ;Force PT7 to 0 NOW
```

Example 5 – generate 1KHz square wave

```
; f = 1KHz : T = 1ms = 8000 periods of 8 MHz clock
       bset TSCR1,%10000000   ;enable timer
       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
```

Example 6: Output 1ms pulse on PJ7

```
; Force PJ7=1, wait 1 msec, then force PJ7=0
; Assume 8MHz clock: T=.125µs
; 1 msec/.125µs = 8000 counts
    bset TSCR1,%10000000     ;enable timer
    bset TIOS,%00000010      ;enable OC chan 1
    bset DDRJ,%10000000      ;PJ7 = output
    bset PORTJ,%10000000     ;PJ7 = 1
    ldd  TCNT                ;current TCNT
    addd #8000               ;1msec from now
    std  TC1                 ;write OC register 1
    movb #%00000010,TFLG1    ;reset CF1
S1: brclr TFLG1,%00000010,S1 ;wait until CF1=1 again
    bclr PORTJ,%10000000     ;PJ7 = 0
```

Example 7: Output 10ms pulse on PJ7, using prescaler for longer time

```
; Force PJ7=1, wait 10 msec, then force PJ7=0
; Assume 8MHz clock divided by 4: T = .5µs
; 10 msec/.5µs = 20000 counts
    bset TSCR2,%00000010      ;set prescale=4 in TSCR2
    bclr TSCR2,%00000101
    bset TSCR1,%10000000      ;enable timer
    bset TIOS,%00000010       ;enable OC chan 1
    bset DDRJ,%10000000       ;PJ7 = output
    bset PORTJ,%10000000      ;PJ7 = 1
    ldd  TCNT                 ;current TCNT
    addd #2000                ;10msec from now
    std  TC1                  ;write to OC register 1
    movb #%00000010,TFLG1     ;reset CF1
S1: brclr TFLG1,%00000010,S1 ;wait until CF1=1 again
    bclr PORTJ,%10000000      ;PJ7 = 0
```

Output compare software checklist – to set up channel n for output compare

1. Initialize interrupt vector(s) for timer channel n
2. Set bit 7 (TEN) in TSCR1 to enable timer
3. Set bits in TIOS to enable output compare channels
4. Initialize OLMn:OLDn in TCNTL1/2 if timer output pins to be used
5. Load current TCNT and add to desired delay
6. Store (TCNT+delay) into TCn register to be used
7. Reset flag(s) in TFLG1
8. Enable any required interrupts in TIE
9. Clear I bit in CCR to unmask interrupts, if interrupt to be used
10. Wait for output compare: poll flag CFn in TFLG1 or wait for interrupt
11. After compare, reinitialize TCn to time of next event by adding delay to current TCn value
12. Reset CFn flag bit to enable next event

Timer input capture function

- Watch for events on Port T input pins
  - “event” = rising edge and/or falling edge
- When a specified event occurs on PTn:
  - capture TCNT value in register TCn
  - set flag CFn in TFLG1
  - trigger interrupt if CnI set in TMSK1 (and I=1 in CCR)
- Uses:
  - determine exact time of an external event
  - measure width of an external pulse
  - measure period of external signal

HCS12 Timer "Input Capture"

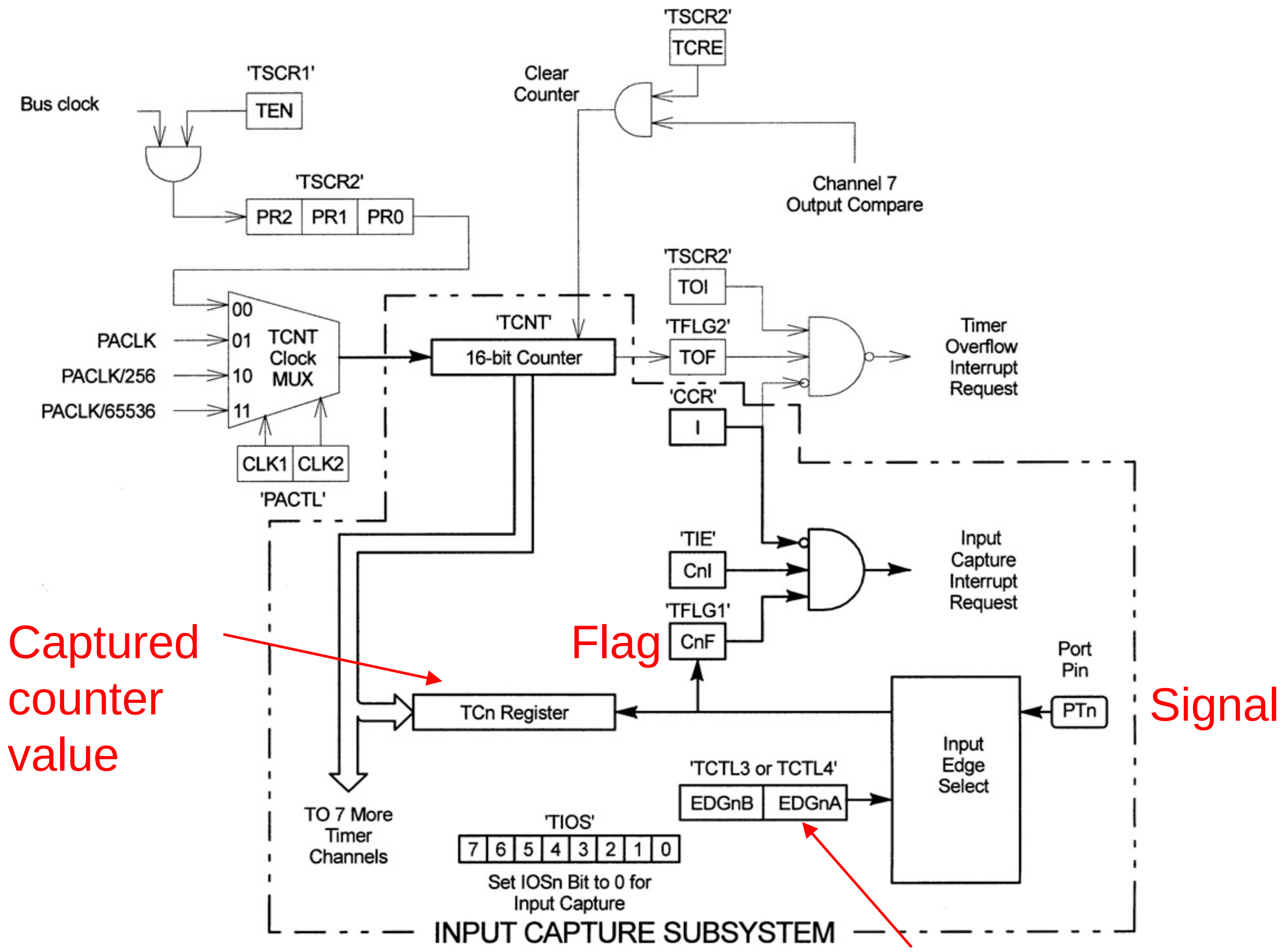


Figure 14-5 Input capture hardware.

Timer Input Capture/Output Compare Select Register (TIOS)

TIOS address: $0080

7	6	5	4	3	2	1	0
IOS7	IOS6	IOS5	IOS4	IOS3	IOS2	IOS1	IOS0

IOSn: Select input capture or output compare mode for channel n

0 : channel acts as <u>input</u> capture, or GPIO for pin PTn (DDRT sets direction)

1 : channel n acts as <u>output</u> compare

Timer Control Registers 3 & 4 (TCTL3, TCTL4)

Define events to be captured on port T pins

	7	6	5	4	3	2	1	0
TCTL3 $004A	EDG7B	EDG7A	EDG6B	EDG6A	EDG5B	EDG5A	EDG4B	EDG4A
TCTL4 $004B	EDG3B	EDG3A	EDG2B	EDG2A	EDG1B	EDG1A	EDG0B	EDG0A

EDGnB:EDGnA – Events on pin PTn

00	input capture disabled (default)
01	Capture on rising edges only
10	Capture on falling edges only
11	Capture on rising or falling edges

Example 8 – Measure the period of a waveform on pin PT1 (Example 14-21)

```
     bset  TSCR1,#%10000000     ;TEN=1 to enable timer
     bclr  TIOS,#%00000010      ;IOS1=0 (input capture)
     movb  #%00000100,TCTL4     ;EDG1A/B=01(PT1 rise edge)
     movb  #%00000010,TFLG1     ;reset C1F
S1:  brclr TFLG1,%00000010,S1   ;wait for C1F (rising edge)
     ldd   TC1                  ;time of 1st rising edge
     pshd                       ;save it
     movb  #%00000010,TFLG1     ;reset C1F
S2:  brclr TFLG1,%00000010,S2   ;wait for C1F (rising edge)
     ldd   TC1                  ;time of 2nd rising edge
     subd  0,SP                 ;period=2nd edge time – 1st
     std   Period               ;store it
     leas  2,sp                 ;adjust SP
```

Example 9 – Measure the period of a waveform on pin PT1 using interrupts

```
    movw  #0,Period         ;initialize measured period
    movw  #1,Count          ;count edges
    bset  TSCR1,#%10000000 ;TEN=1 to enable timer
    bclr  TIOS,#%00000010  ;IOS1=0 (input capture)
    movb  #%00000100,TCTL4 ;EDG1A/B=01(PT1 rise edge)
    movb  #%00000010,TFLG1 ;reset C1F
    bset  TIE,%00000010    ;set CI1 to enable ch. 1 interrupt
    cli                    ;enable CPU interrupts
      ...do some other processing
; Timer interrupt routine - entered on PT1 rising edge
isr:  ldd   TC1                    ;time of rising edge
      subd  Period                 ;this edge - previous
      std   Period                 ;save 1st edge time or period
      movb  #%00000010,TFLG1       ;clear CF1
      inc   Count                  ;1 if 1st edge, 2 if period
      rti
      ORG  $FFEC
      dc.w  isr       ;Timer channel 1 interrupt vector
```

Pulse accumulator

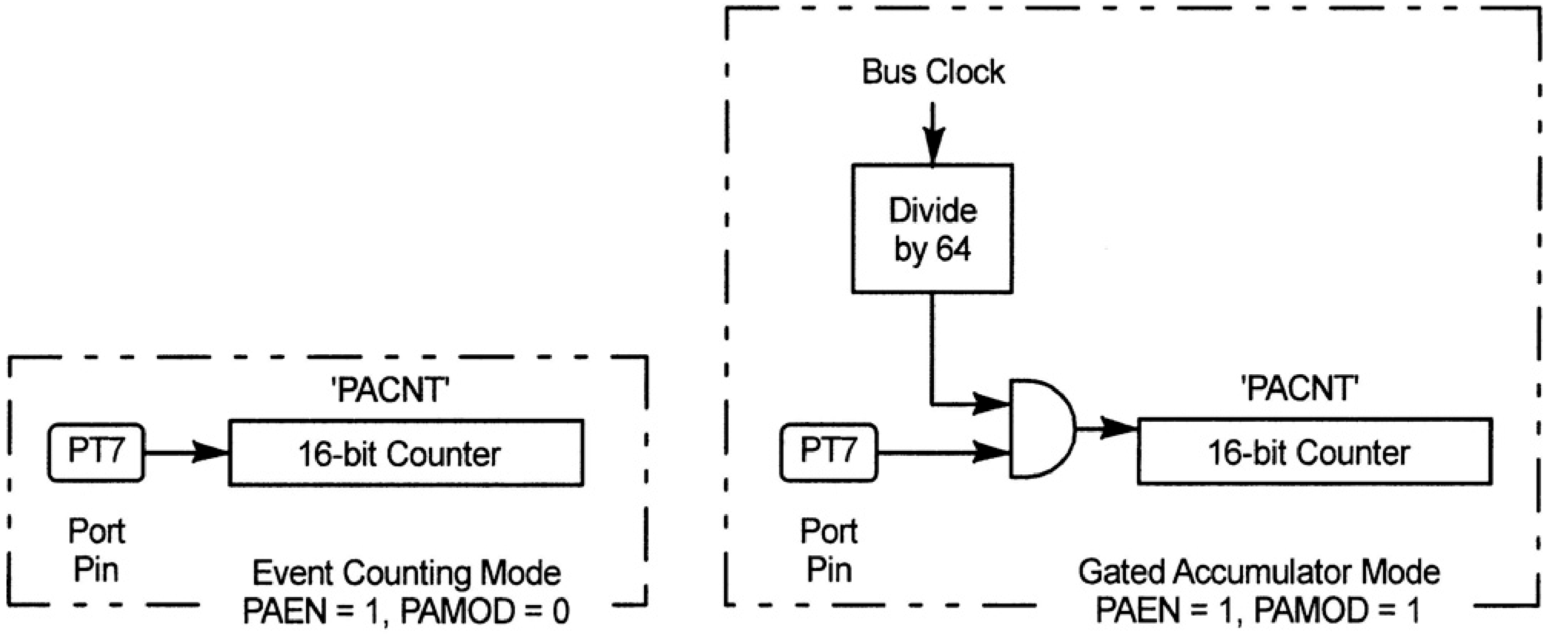


Figure 14-6 Pulse accumulator operating modes.

Real-time interrupt (RTI)

- In Clocks and Reset Generator (CRG) module
- Divide oscillator by a programmable value
  - Divider has three parts:
    - Divide by constant 1024 (2^{10})
    - RTI prescale: divide by 1,2,4,8,16,32,64
    - RTI modulus counter: divide by 1-16
  - Set RTIF flag to 1 when modulus count reached
    - Reset
  - Interrupt on RTIF=1 if RTIE (RTI interrupt enable) set

Real-time interrupt hardware.

- Like timer overflow interrupt, but with selectable rate
- Rate set by fixed 13-bit counter and programmable prescaler
- Flag set on overflow, interrupt if so enabled

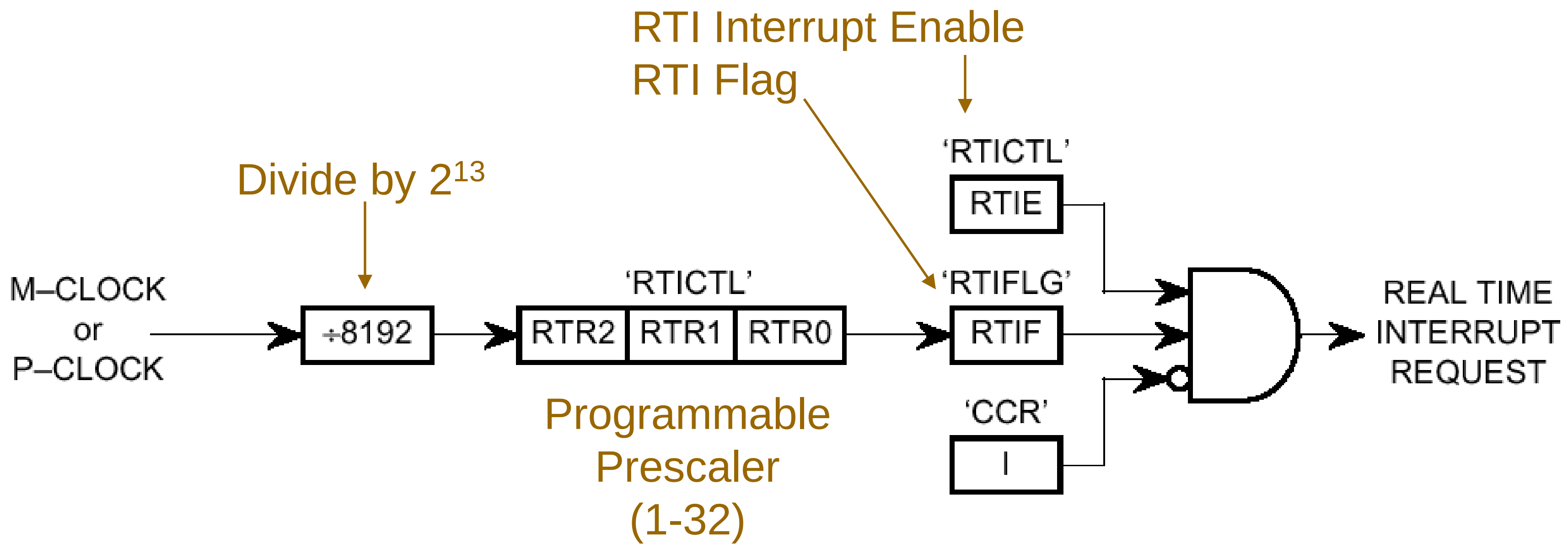
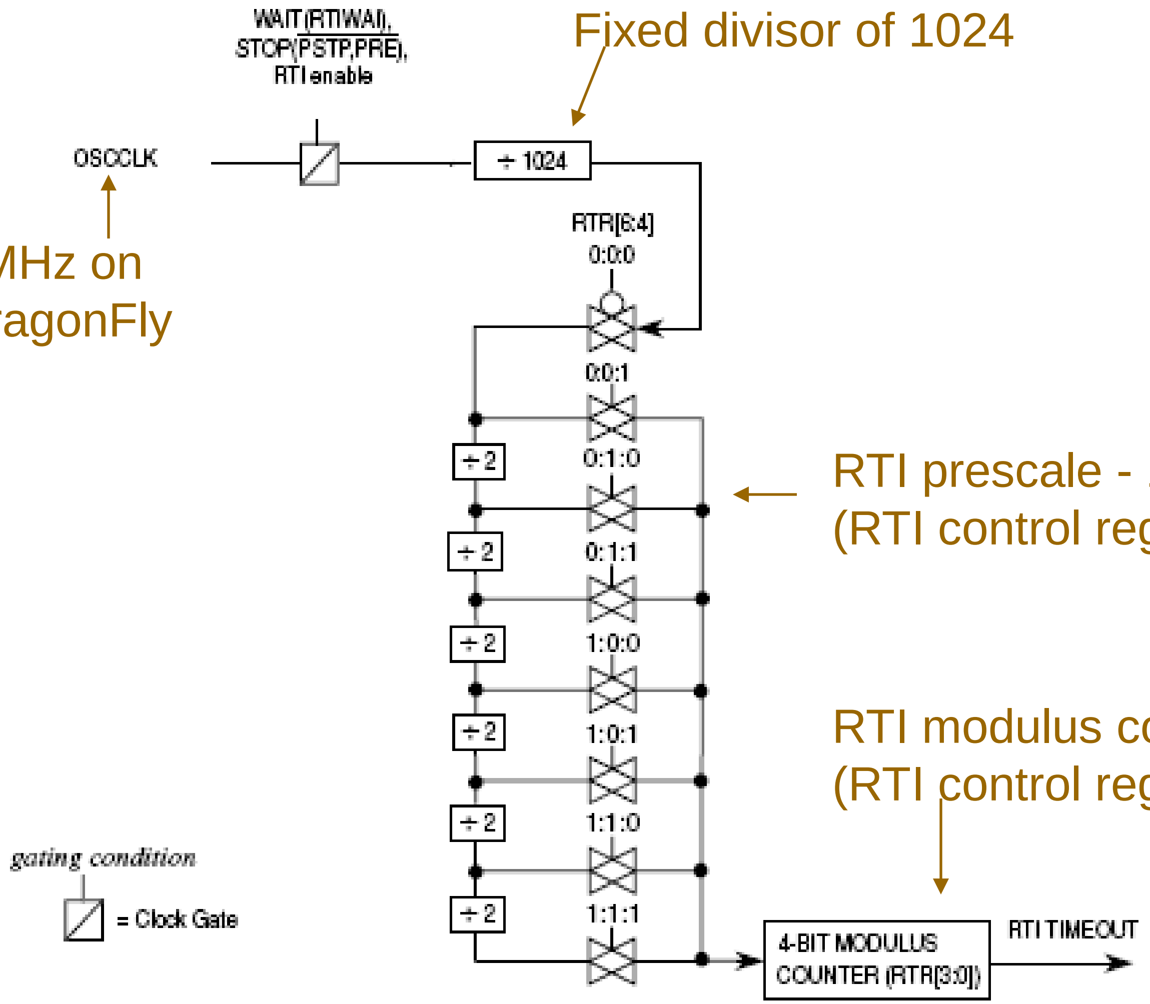


Fig. 10-7

Real-time interrupt clock chain



Fixed divisor of 1024

8MHz on DragonFly

RTI prescale - 1,2,4,8,16,32,64
(RTI control reg. RTICTL bits 6:4)

RTI modulus counter: 1-6
(RTI control reg. RTICTL bits 3:0)

Sets RTI flag

Real-Time Interrupt Control Register (RTICTL - $0014)

7	6	5	4	3	2	1	0
0	RTR6	RTR5	RTR4	RTR3	RTR2	RTR1	RTR0

RTI Prescale Select

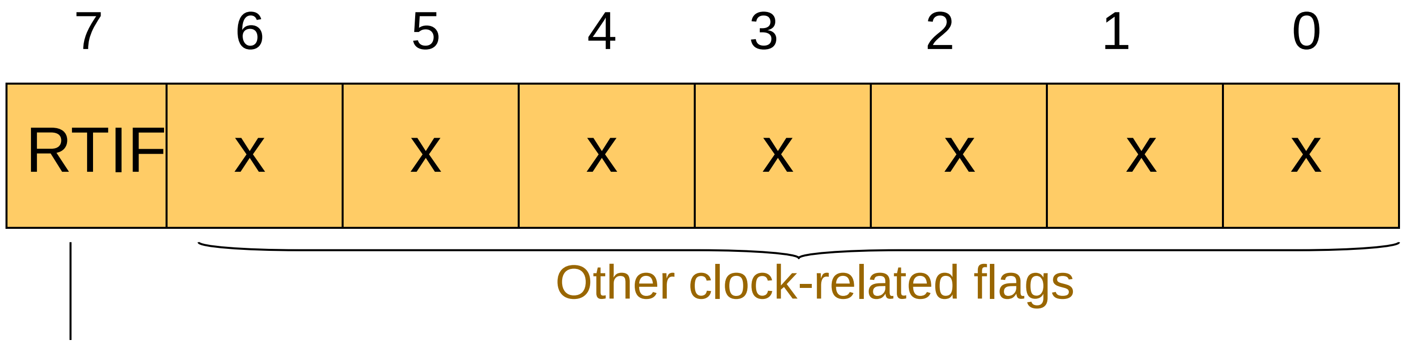
RTI Modulus Counter

<u>RTR6:4</u>	<u>Divide OSC by:</u>
000	Off
001	$2^{10} = 1x2^{10}$
010	$2^{11} = 2x2^{10}$
011	2^{12}
100	2^{13}
101	2^{14}
110	2^{15}
111	$2^{16} = 64x2^{10}$

RTR3:0 = (count -1) : counts 1-16
Set flag when this value reached.

<u>Example</u>:
RTR6:4 = 011, RTR3:0 = 1001
OSC divided by (2^{12} x 10) = 40960
Assume OSC = 8MHz
Interrupt @ 8MHz/40960 = 195.3 Hz
1/195.3 = 5.1 msec

CRG Flag Register (CRGFLG = $0037)



Real-Time Interrupt Flag

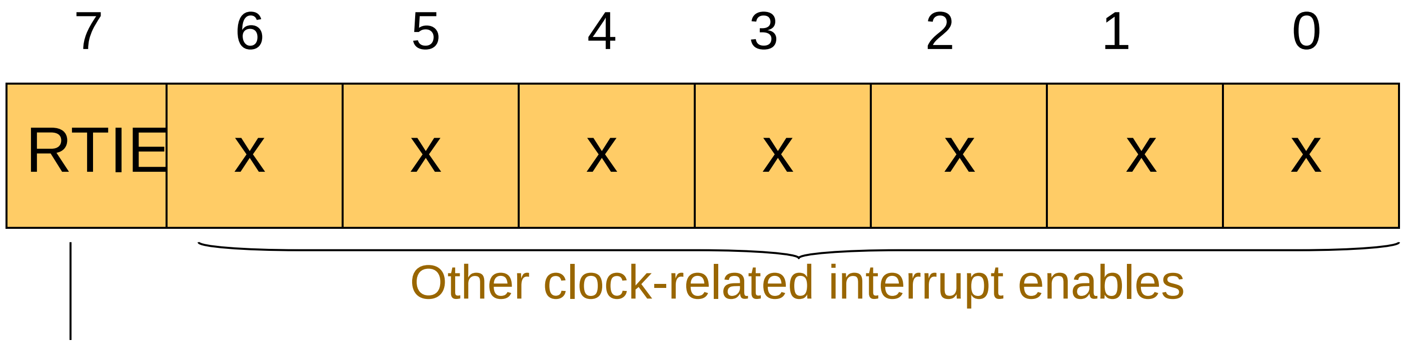
-Set when RTI modulus counter rolls over

(interrupt if RTIE bit set in CRGINT)

-Reset by writing 1 to RTIF bit

Ex. movb #%10000000,CRGFLG

CRG Interrupt Enable Register (CRGINT = $0038)



Real-Time Interrupt Enable

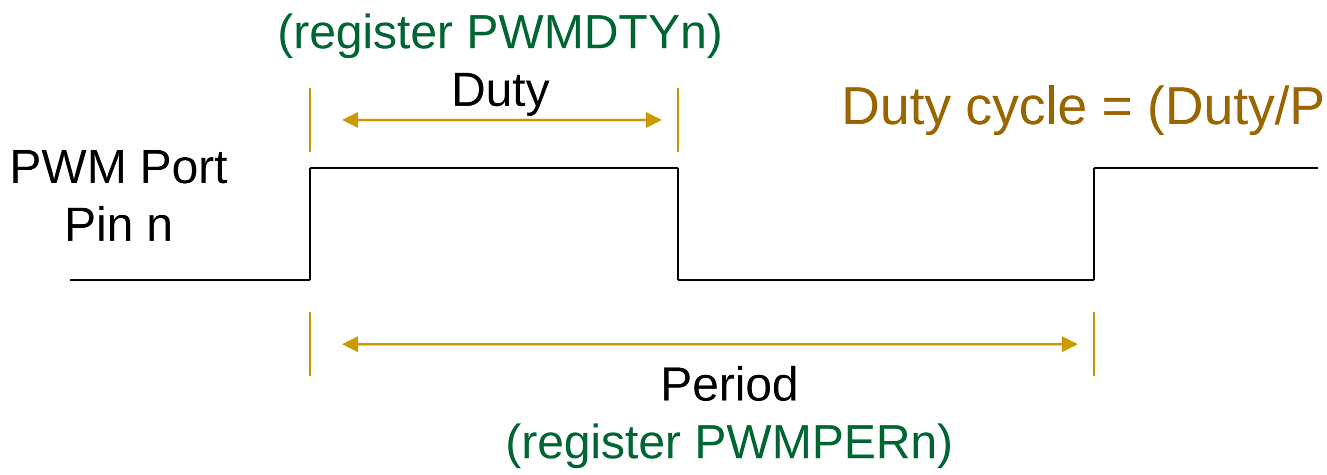
- 0 – RTI interrupt requests disabled (default)
- 1 – RTI interrupt requests enabled

Notes:

- RTI interrupt also requires CCR I flag = 0
- Real-time interrupt vector at $FFF0:$FFF1

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



Duty cycle = (Duty/Period) x 100%

PWM signal formats

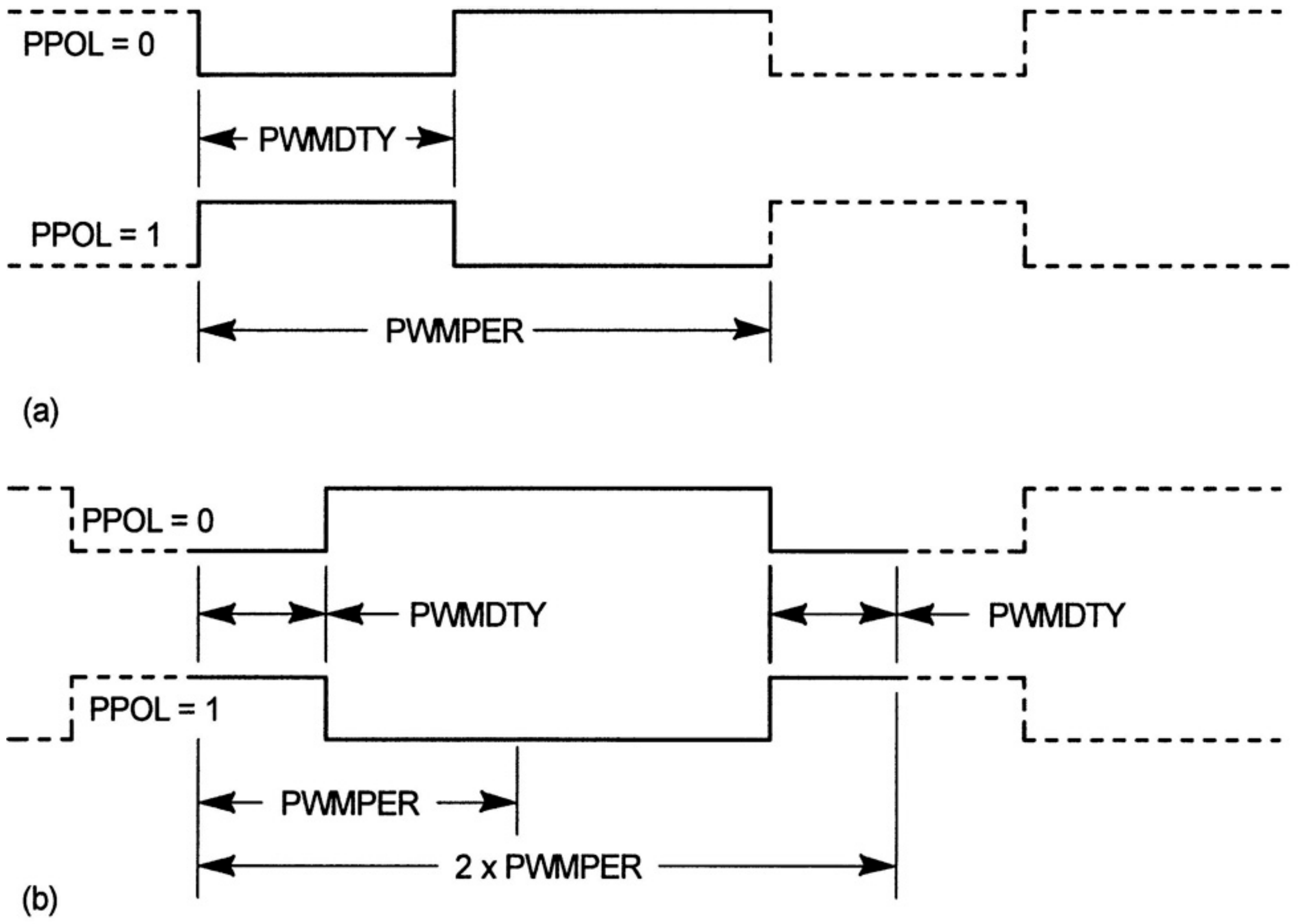
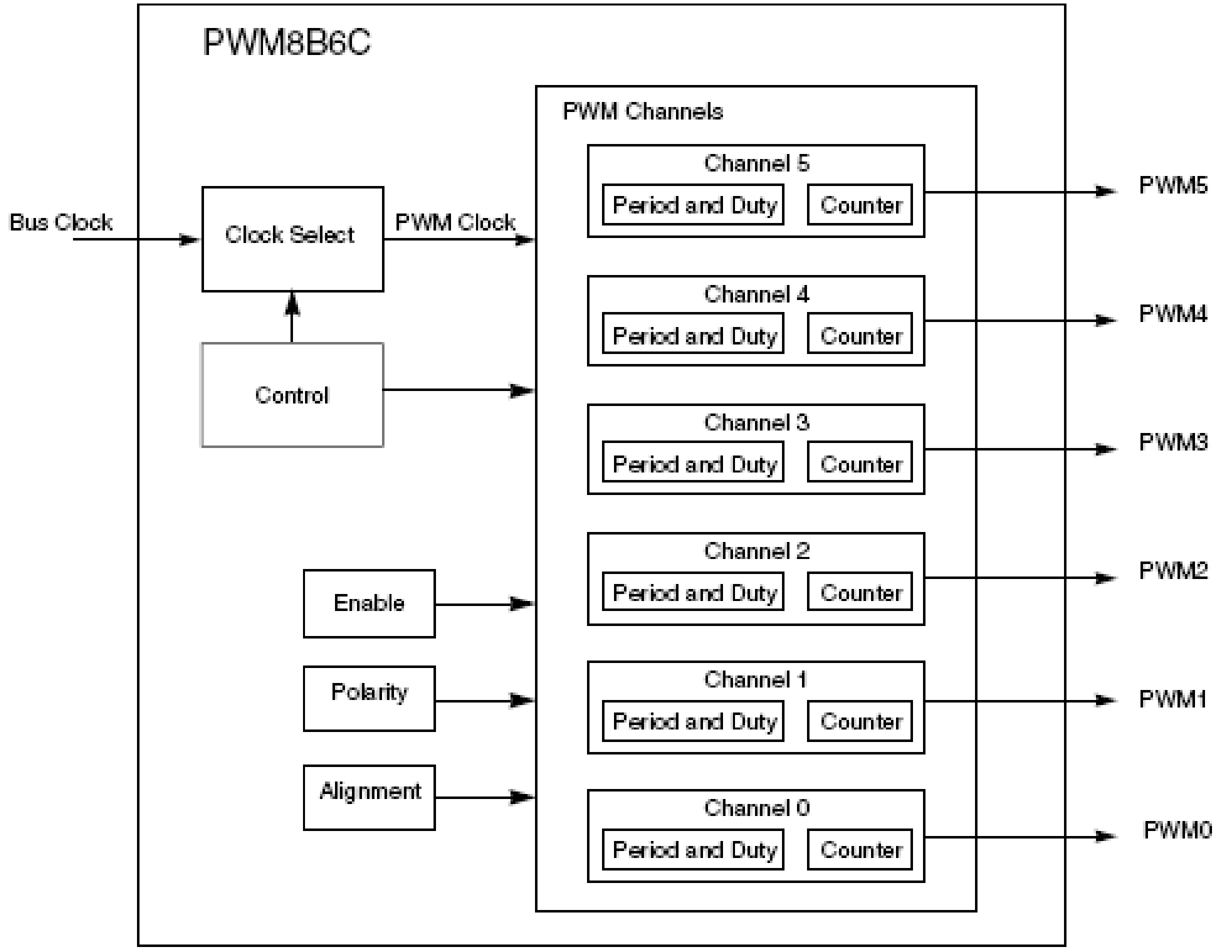


Figure 14-11 (a) Left-aligned and (b) center-aligned pulse-width modulator waveforms.

PWM block diagram

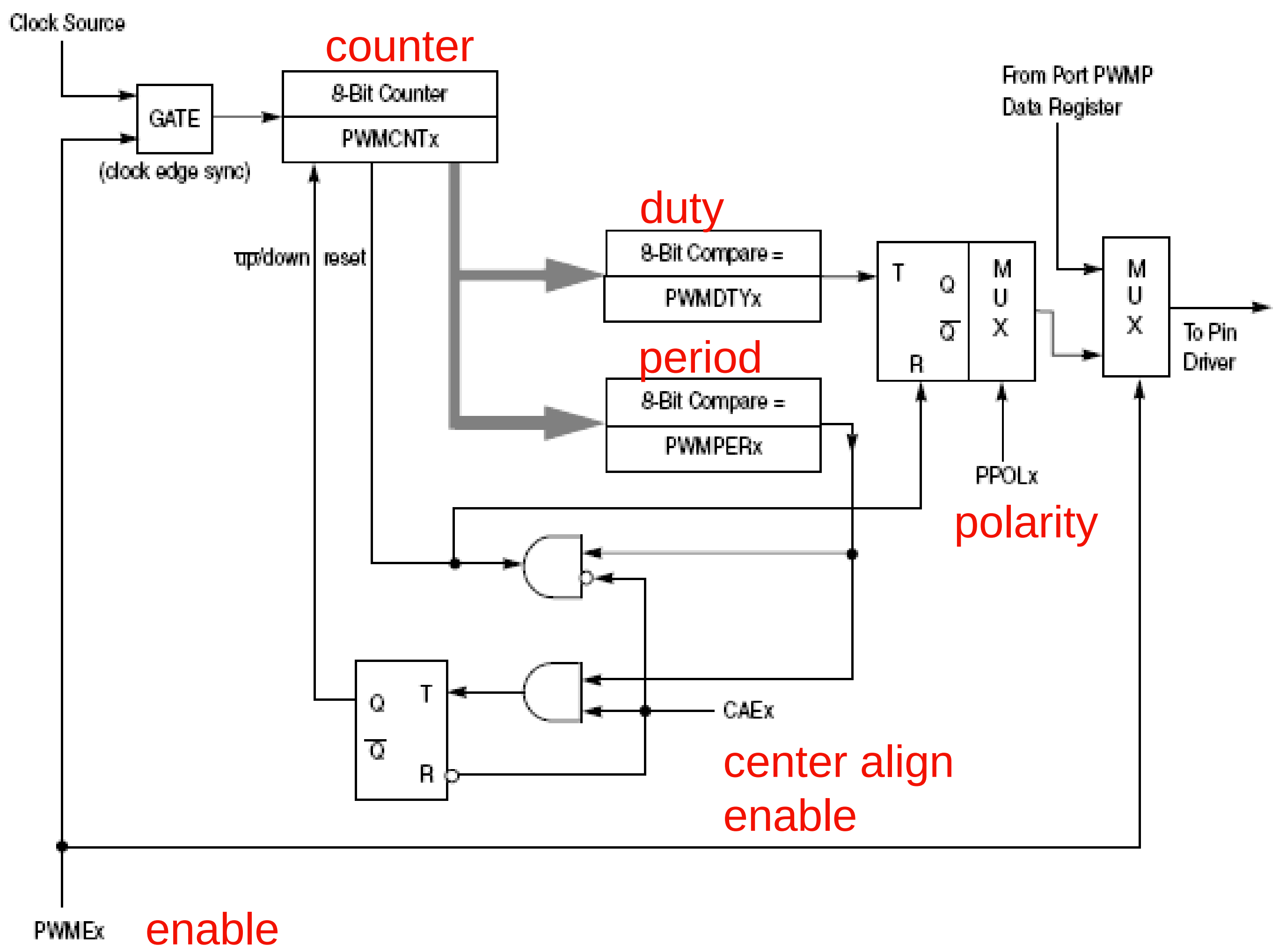


Pins in
Ports P or T

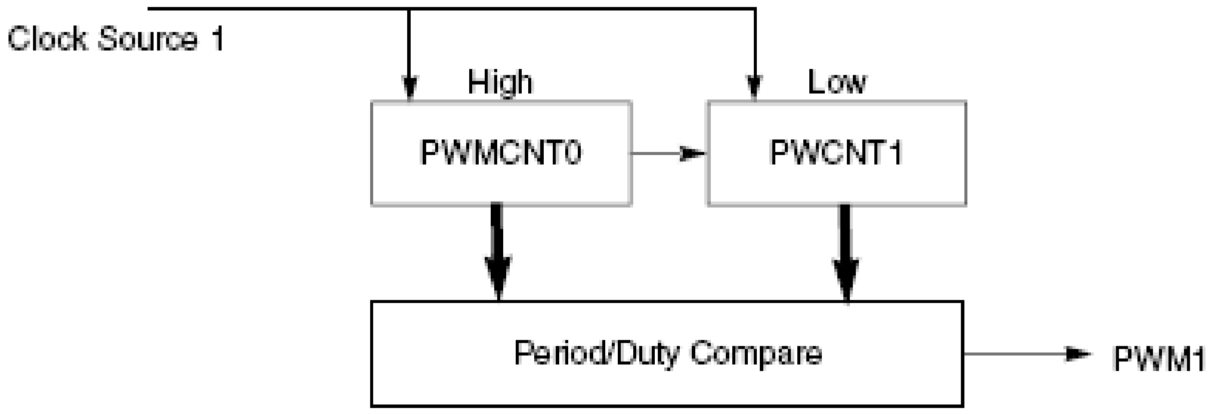
PWM period, duty & counter registers

- 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 timer channel



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

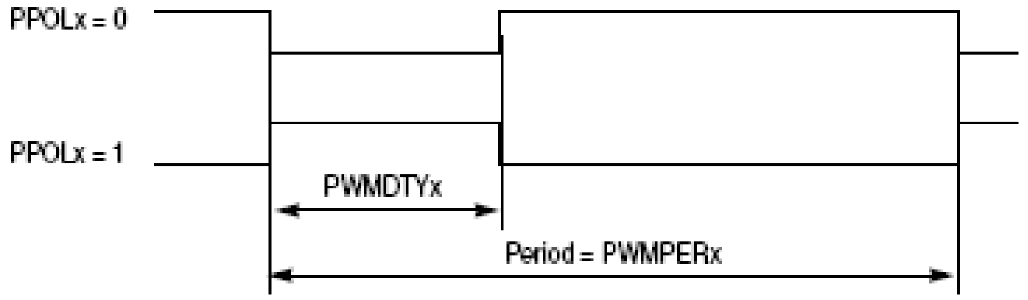


Figure 12-36. PWM Left Aligned Output Waveform

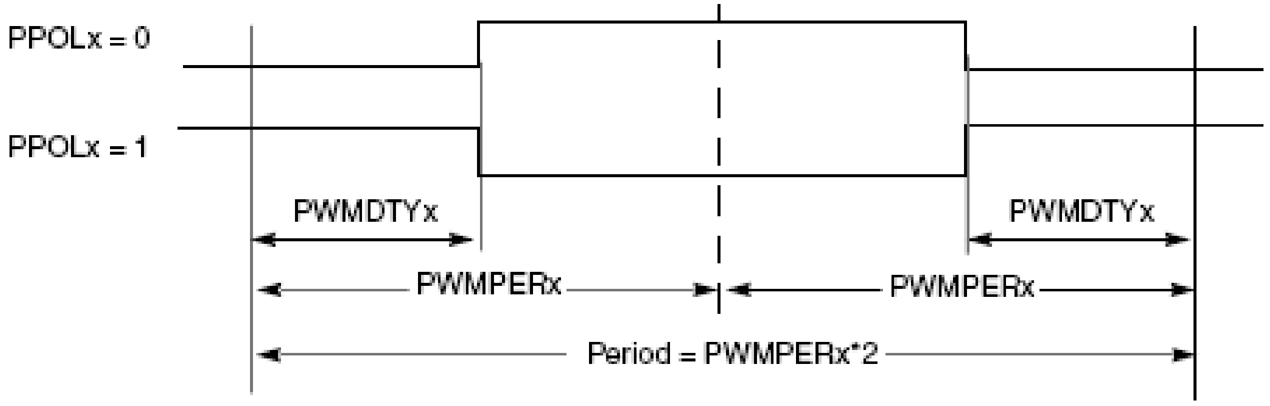


Figure 12-38. PWM Center Aligned Output Waveform

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)

<u>Channels 0,1,4,5:</u>	<u>Channels 2,3:</u>
0 = select clock A	0 = select clock B
1 = select clock SA	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)	*100 – bus clock/16*
001 – bus clock/2	*101 – bus clock/32*
010 – bus clock/4	*110 – bus clock/64*
011 – bus clock/8	*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:
  - PP5/KPW5 only Port P connection available
  - All 8 bits of Port T available

Summary of HCS12 timer features

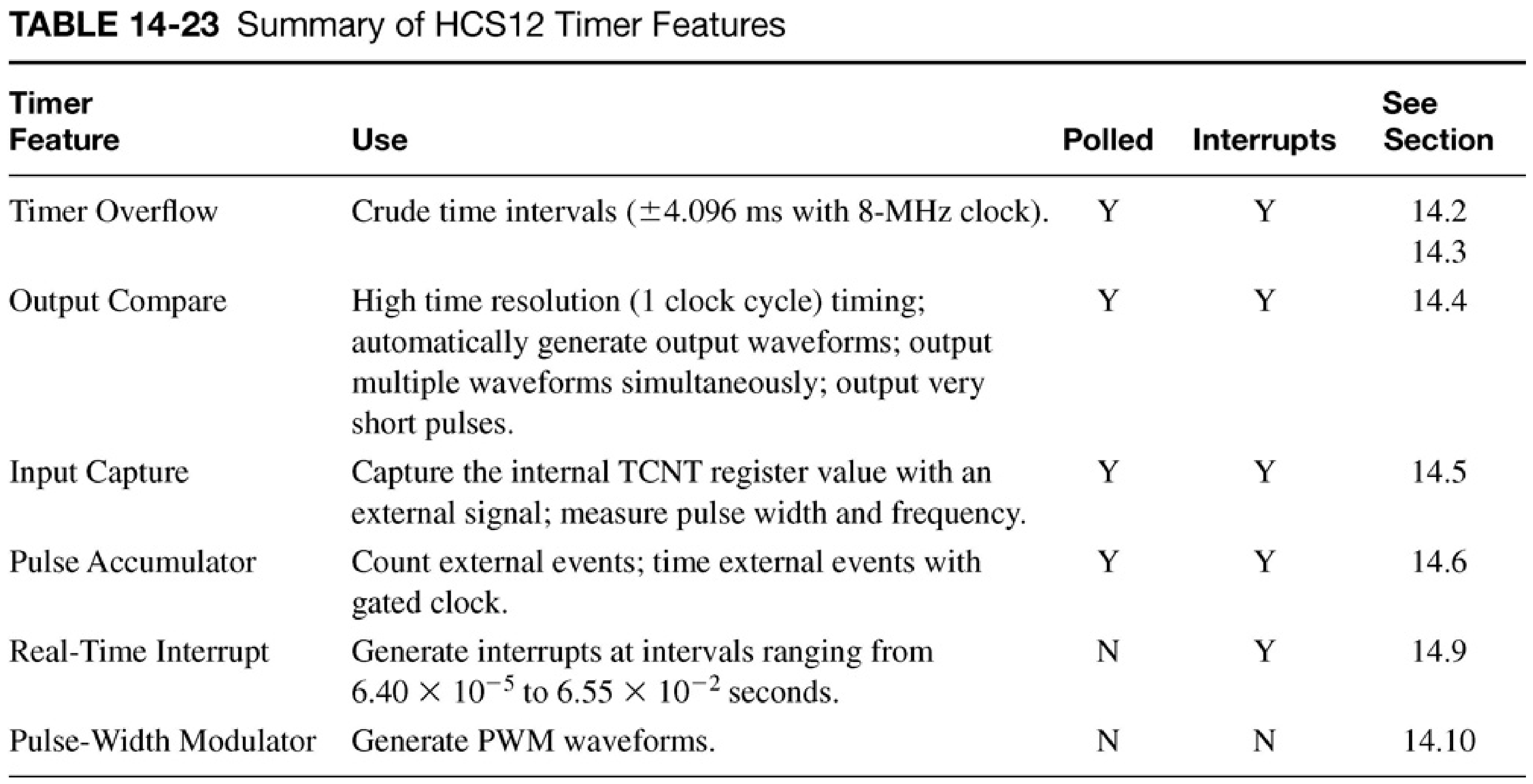


TABLE 14-23 Summary of HCS12 Timer Features

Timer Feature	Use	Polled	Interrupts	See Section
Timer Overflow	Crude time intervals (±4.096 ms with 8-MHz clock).	Y	Y	14.2 14.3
Output Compare	High time resolution (1 clock cycle) timing; automatically generate output waveforms; output multiple waveforms simultaneously; output very short pulses.	Y	Y	14.4
Input Capture	Capture the internal TCNT register value with an external signal; measure pulse width and frequency.	Y	Y	14.5
Pulse Accumulator	Count external events; time external events with gated clock.	Y	Y	14.6
Real-Time Interrupt	Generate interrupts at intervals ranging from 6.40×10^{-5} to 6.55×10^{-2} seconds.	N	Y	14.9
Pulse-Width Modulator	Generate PWM waveforms.	N	N	14.10