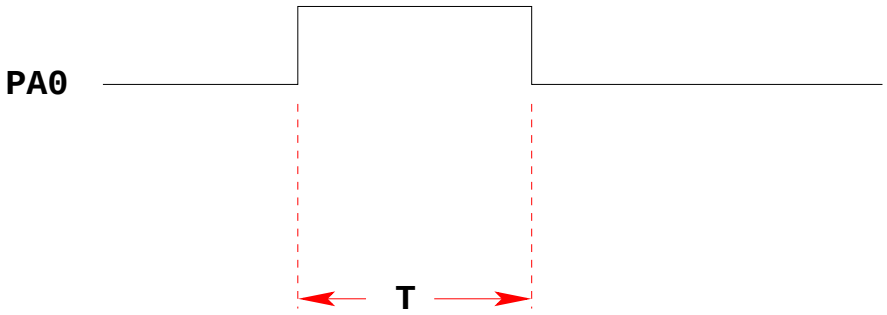
The HC12 Output Compare Function

;

Want event to happen at a certain time

Want to produce pulse pulse with width T



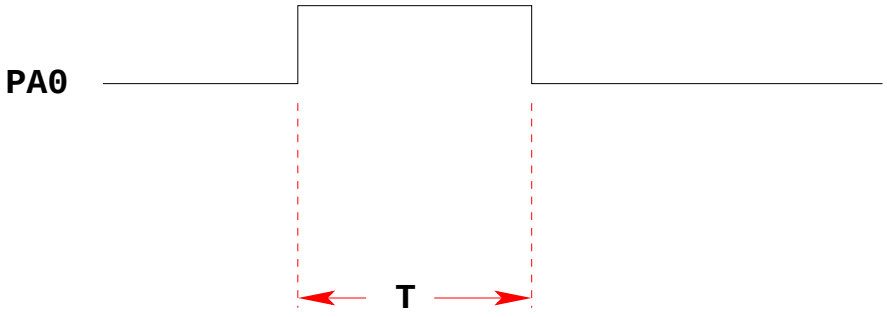
Wait until TCNT == 0x0000, then bring PA0 high

Wait until TCNT == T, then bring PA0 low

```
while (TCNT != 0x0000) ;
PORTA = PORTA | 0x01;
while (TCNT != T) ;
PORTA = PORTA & ~0x01;
```

Want event to happen at a certain time

Want to produce pulse pulse with width T



Wait until TCNT == 0x0000, then bring PA0 high

Wait until TCNT == T, then bring PA0 low

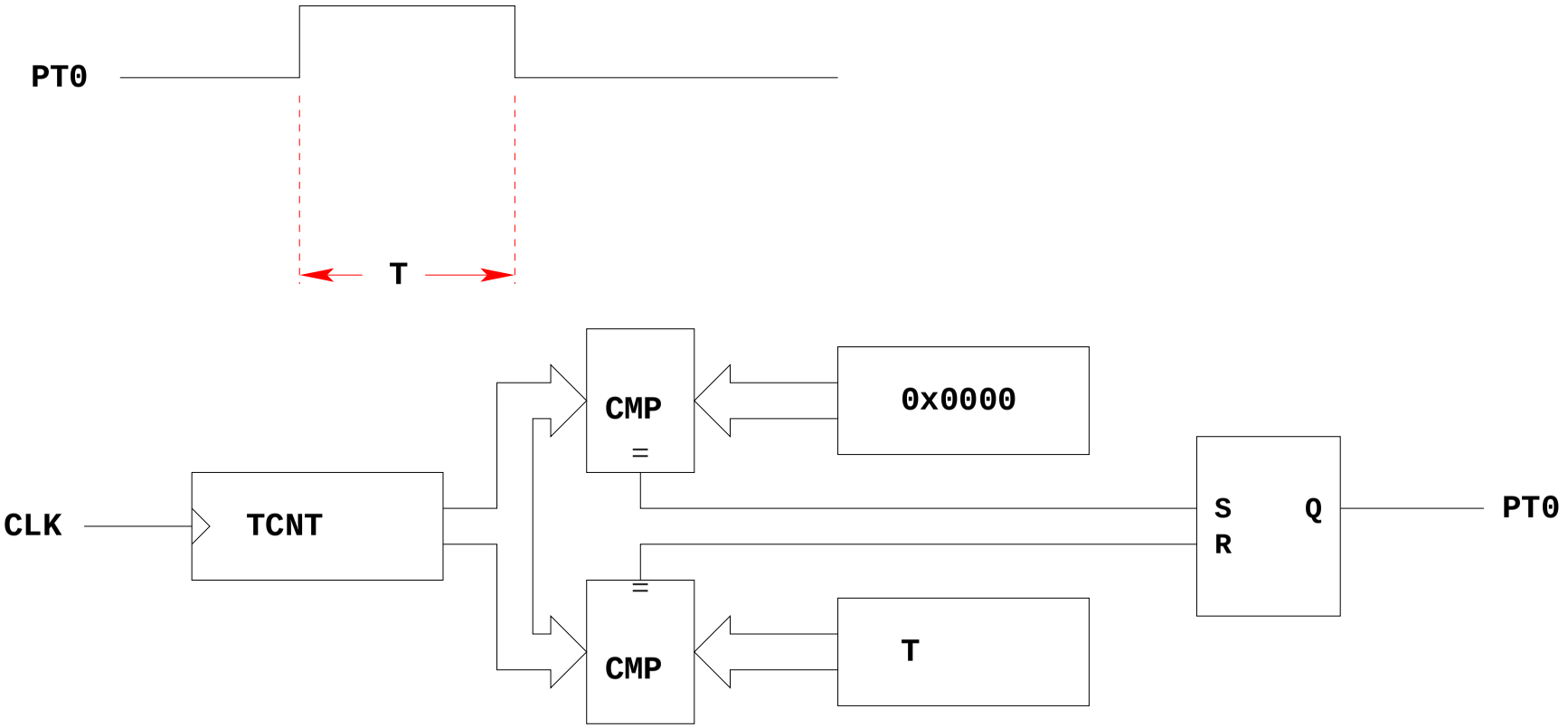
```
while (TCNT != 0x0000) ;
PORTA = PORTA | 0x01;
while (TCNT != T) ;
PORTA = PORTA & ~0x01;
```

Problems:

1) May miss TCNT == 0x0000 or TCNT == T
2) Time not exact -- software delays
3) Cannot do anything else while waiting

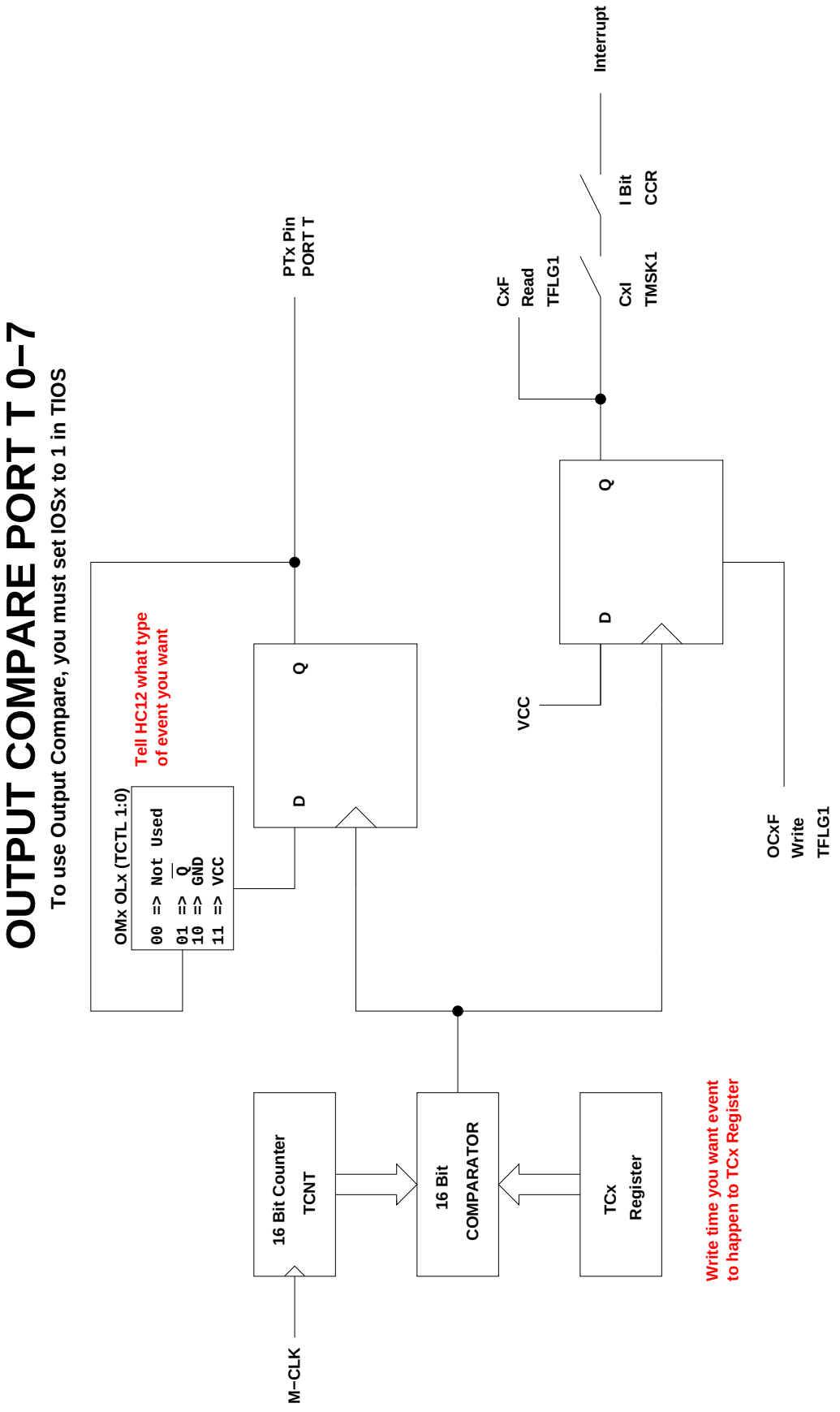
Want event to happen at a certain time

Want to produce pulse pulse with width T



When TCNT == 0x0000, the output goes high
When TCNT == T, the output goes low

Now pulse is exaclty T cycles long


OUTPUT COMPARE PORT T 0–7
To use Output Compare, you must set IOSx to 1 in TIOS
Interrupt
I Bit
CCR
PTx Pin
PORT T
CxF
Read
TFLG1
CxI
TMSK1
OMx OLx (TCTL 1:0)
00 => Not Used
01 => Q
10 => GND
11 => VCC
Tell HC12 what type
of event you want
Q
D
Q
D
VCC
OCxF
Write
TFLG1
16 Bit Counter
TCNT
16 Bit
COMPARATOR
TCx
Register
Write time you want event
to happen to TCx Register
M–CLK

The HC12 Output Compare Function

- The HC12 allows you to force an event to happen on any of the eight PORTT pins
- An external event is a rising edge, a falling edge, or a toggle
- To use the Output Compare Function:
  - Enable the timer subsystem (set TEN bit of TSCR)
  - Set the prescaler
  - Tell the HC12 that you want to use Bit x of PORTT for output compare
  - Tell the HC12 what you want to do on Bit x of PORTT (generate rising edge, falling edge, or toggle)
  - Tell the HC12 what time you want the event to occur
  - Tell the HC12 if you want an interrupt to be generated when the event is forced to occur

Write a 1 to Bit 7 of TSCR to turn on timer

TEN	TSWAI	TSBCK	TFFCA					**0x0086 TSCR**

To turn on the timer subsystem: TSCR = 0x80;

Set the prescaler in TMSK2

Make sure the overflow time is greater than the time difference you want to generate

TOI	0	PUPT	RDPT	TCRE	**PR2**	**PR1**	**PR0**	**0x008D TMSK2**

PR2	PR1	PR0	Period (μs)	Overflow (ms)
0	0	0	0.125	8.192
0	0	1	0.250	16.384
0	1	0	0.500	32.768
0	1	1	1.000	65.536
1	0	0	2.000	131.072
1	0	1	4.000	262.144
1	1	0	------	------
1	1	1	------	------

To have overflow rate of 65.536 ms:

TMSK2 = 0x03;

Write a 1 to the bits of TIOS to make those pins output compare

IOS7	IOS6	IOS5	IOS4	IOS3	IOS2	IOS1	IOS0	0x0080 TIOS

To make Pin 4 an output compare pin: TIOS = TIOS | 0X10;

Write to TCTL1 and TCTL2 to choose action to take

OM7	OL7	OM6	OL6	OM5	OL5	OM4	OL4	0x0088 TCTL1

OM3	OL3	OM2	OL2	OM1	OL1	OM0	OL0	0x0089 TCTL2

OMn	OLn	Configuration
0	0	Disconnected
0	1	Toggle
1	0	Clear
1	1	Set

To have Pin 4 toggle on compare:

TCTL1 = (TCTL1 | 0x01) & ~0x02;

Write time you want event to occur to TCn register.

To have event occur on Pin 4 when TCNT == 0x0000: TC4 = 0x0000;

To have next event occur T cycles after last event, add T to TCn.

To have next event occur on Pin 4 500 cycles later: TC4 = TC4 + 500;

When TCNT == TCn, the specified action will occur, and flag CFn will be set.

To clear the flag, write a 1 to the bit you want to clear (0 to all others)

CF7	CF6	CF5	CF4	CF3	CF2	CF1	CF0	0x008E TFLG1

To wait until TCNT == TC4: while ((TFLG1 & 0x10) == 0) ;

To clear flag bit for Pin 4: TFLG1 = 0x10;

To enable interrupt when compare occurs, set corresponding bit in TMSK1 register

C7I	C6I	C5I	C4I	C3I	C2I	C1I	C0I	0x008C TMSK1

To enable interrupt when TCNT == TC4: TMSK1 = TMSK1 | 0x10;

USING OUTPUT COMPARE ON THE HC12

1. In the main program:
   (a) Turn on timer subsystem (TSCR reg)
   (b) Set prescaler (TMSK2 reg)
   (c) Set up PTx as OC (TIOS reg)
   (d) Set action on compare (TCTL 1-2 regs, OMx OLx bits)
   (e) Clear Flag (TFLG1 reg)
   (f) Enable int (TMSK1 reg)
2. In interrupt service routine
   (a) Set time for next action to occur (write TCx reg)
      - For periodic events add time to TCx register
   (b) Clear flag (TFLG1 reg)

```
/*
 * Program to generate square wave on PT2
 * Frequency of square wave is 500 Hz
 * Period of square wave is 2 ms
 * Set prescale to give 1 us cycle
 * 2 ms is 2,000 cycles of 1 us/cycle
 *
 */
#include "hc12b32.h"

#define PERIOD      2000
#define HALF_PERIOD (PERIOD/2)

#define TRUE       1

main()
{

    TSCR = 0x80;                    /* Turn on timer subsystem */
    TMSK2 = 0x03;                   /* Set prescaler to 8 */

    TIOS = TIOS | 0x04;             /* Configure PT2 as Output Compare */
    TCTL2 = (TCTL2 | 0x10) & ~0x20;  /* Set up PT2 to toggle on compare */
    TFLG2 = 0x04;                   /* Clear flag and enable interrupt on C2 *
    TMSK1 = TMSK1 | 0x04;

    enable();

    while (TRUE)
    {
        _asm("wai");
    }
}

@interrupt void toc2_isr(void)
{
    TC2 = TC2 + HALF_PERIOD;
    TFLG1 = 0x04;
}
```

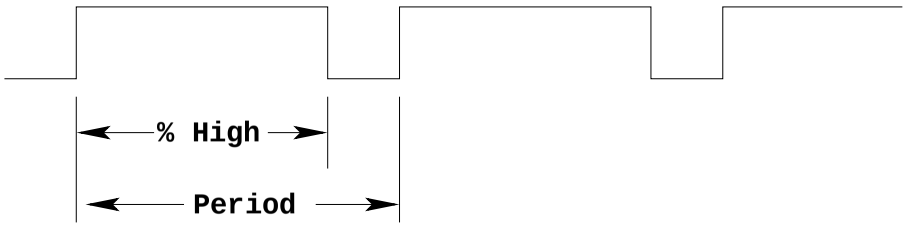
```
/*      INTERRUPT VECTORS TABLE 68HC12
 */
void toc2_isr();         /* character receive handler */

void (* const _vectab[])() = {   /* 0x0B10 */
        0,              /* BDLC            */
        0,              /* ATD             */
        0,              /* reserved        */
        0,              /* SCI0            */
        0,              /* SPI             */
        0,              /* Pulse acc input */
        0,              /* Pulse acc overf */
        0,              /* Timer overf     */
        0,              /* Timer channel 7 */
        0,              /* Timer channel 6 */
        0,              /* Timer channel 5 */
        0,              /* Timer channel 4 */
        0,              /* Timer channel 3 */
        toc2_isr,       /* Timer channel 2 */
        0,              /* Timer channel 1 */
        0,              /* Timer channel 0 */
        0,              /* Real time       */
        0,              /* IRQ             */
        0,              /* XIRQ            */
        0,              /* SWI             */
        0,              /* illegal         */
        0,              /* cop fail        */
        0,              /* cop clock fail  */
        (void *)0xff80, /* RESET           */
        };
```

Pulse Width Modulation

- Often want to control something by adjusting the percentage of time the object is turned on
- For example,
  - A DC motor — the higher the percentage, the faster the motor goes
  - A light – the higher the percentage, the brighter the light
  - A heater – the higher the percentage, the more heat output
- Can use Output Compare to generate a PWM signal
- Because PWM is used so often the HC12 has a built-in PWM system
- The PWM system on the HC12 is very flexible
  - It allows you to set a wide range of PWM frequencies
  - It allows you to generate up to 4 separate PWM signals, each with a different frequency
  - It allows you to generate 8-bit PWM signals (with 0.5% accuracy) or 16-bit PWM signals (with 0.002% accuracy)
  - It allows you to select high polarity or low polarity for the PWM signal
  - It allows you to use left-aligned or center-aligned PWM signals
- Because the HC12 PWM systes is so flexible, it is fairly complicated to program
- To simplify the discussion we will only discuss 8-bit, left-aligned, high-polarity PWM signals.

Pulse Width Modulation



Need a way to set the PWM period and duty cycle

The HC12 sets the PWM period by counting from 0 the some maximum count with a special PWM clock

PWM Period = PWM Clock Period x (Max Count + 1)

Once the PWM period is selected, the PWM duty cycle is set by telling the HC12 how many counts it should keep the signal high for

PWM Duty Cycle = (Count High + 1)/(Max Count + 1)

The hard part about PWM on the HC12 is figuring out how to set the PWM Period