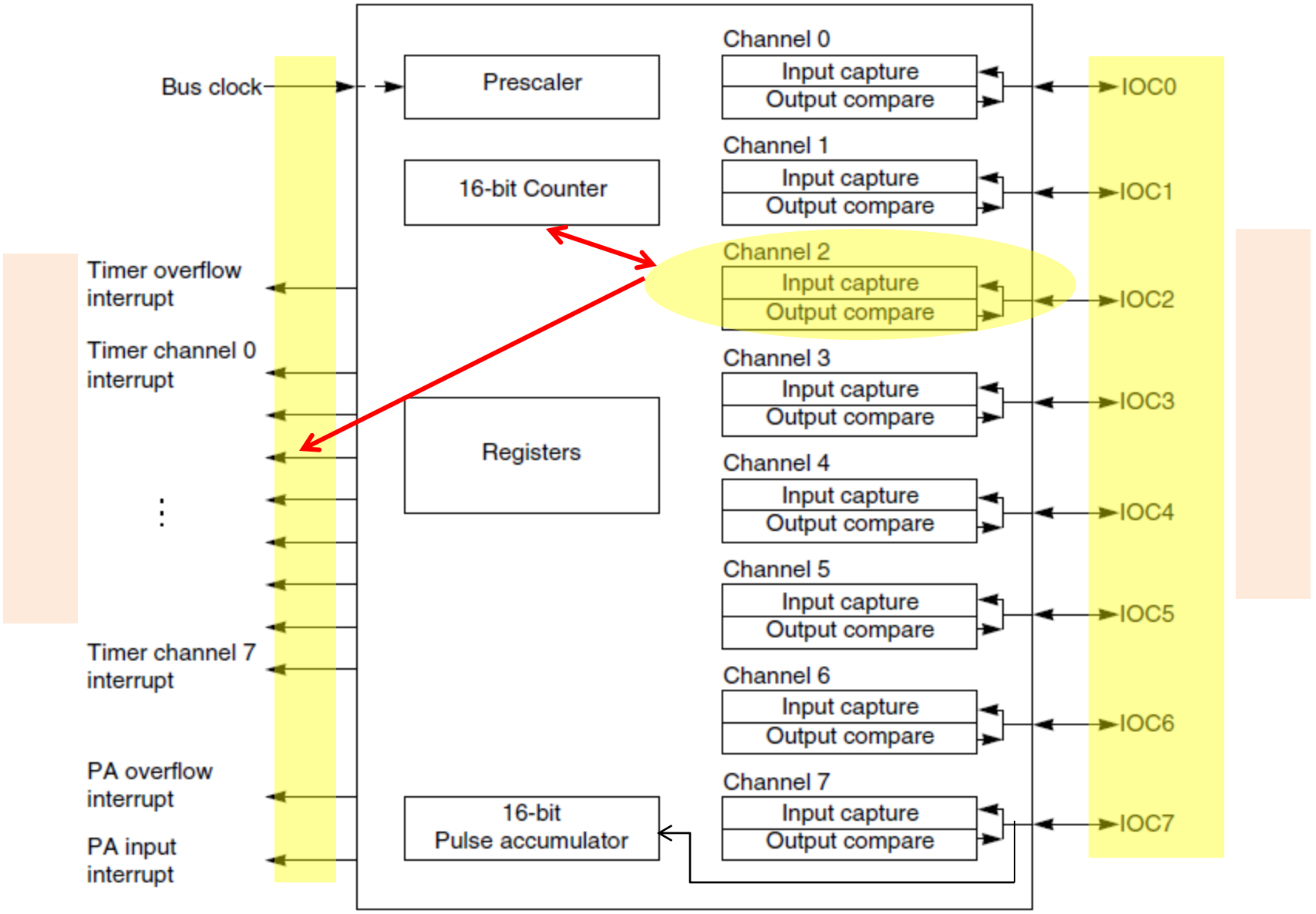
TIM16B8C Timer Module



TIM16B8C Timer Module

The TIM16B8CV1 includes :

- 16-bit counter. software-programmable
- Clock prescaling. seven-stage software-programmable
- Eight input capture / output compare channels.
- 16-bit pulse accumulator event counter or gated time measurement
- Interrupt capabilities

The timer address space occupies 48 bytes in memory:

Module base address to
module base address + $2F.

Timer Applications

Applications:

- time delay creation and measurement
- frequency/time-period measurement
- event counting
- arrival time capturing
- time-of-day tracking
- binary waveform generation
- real-time interrupt generation

Timer Module External Signals

IOCx, x = 0…7
These pins serve as input capture or output compare for channels 0 … 7.

-Input capture function detects an event on the corresponding pin, and records its time.

-Output compare causes an event on the corresponding pin when the specified time instant is reached.

Timer Input Capture/Output Compare Select Register (TIOS) serves for configuring a channel for input capture or output compare:

- For input capture, clear I/O select bit, IOSx.
- For output compare, set I/O select bit, IOSx.

An event is defined as signal edge on the pin.

IOC7 can also be configured as pulse accumulator input.

Timer Module Internal Signals

- Clock
- Interrupts:
    - Timer input-capture/output compare channels interrupts
    - Timer overflow interrupt
    - Pulse accumulator input interrupt
    - Pulse accumulator overflow interrupt

The Timer Clock (TIMCLK)

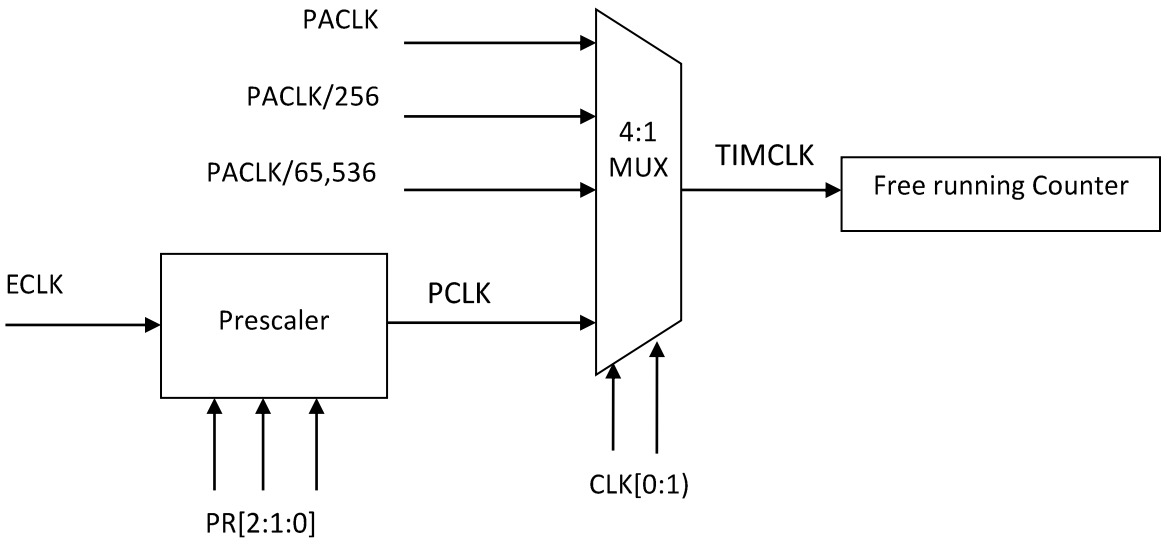
The Timer clock (TIMCLK) of the free running counter is selected from one of four sources:

- Prescaled clock (PCLK) = MCLK/ 1...32
- PACLK, PACLK/256, and PACLK/65,536 (from the Pulse Accumulator if enabled)

The clock source is selected by bits CLK[1:0] in the Pulse Accumulator Control Register PACTL.

The prescaler divisor is selected by bits PR[2:1:0] in Timer System Control Register2 (TSCR2) .

The prescaler divides the bus clock (ECLK) by: 1 2 4 8 16 32 64 128.



The Timer Counter (TCNT)

The free running 16-bit counter (TCNT) operates from the Timer clock (TIMCLK)

Can be started and stopped by program:
To start, set Timer Enable TEN in Timer System Control Register1 (TSCR1).

Can be caused to stop in WAIT mode/debug mode to save power:
set bit TSWAI/TSFRZ in TSCR1.

After Enable it starts at zero and counts continuously to $FFFF, then it rolls over to $0000, and sets the Timer Overflow Flag TOF in Timer Flag Register2 (TFLG2).
If Timer Overflow Interrupt TOI in Timer System Control Register2 (TSCR2) is set to 1, TOF causes an interrupt
The overflow flag can extend the counter range.

The timer count can be read by a sixteen-bit memory read instruction from Timer Count Register High and Low (TCNTH & TCNTL) located at $0004 & $0005 from the base address. Example: LDD $84 or LDX $84.
Write: Has no meaning or effect in the normal mode; only writable in special modes (test_mode = 1).

The Timer Module
Timer System Control Register1 (TSCR1)

Address Base+$0008	7	6	5	4	3	2	1	0
R	TEN	TSWAI	TSFRZ	TFFCA	0	0	0	0
W								
Reset	0	0	0	0	0	0	0	0

TEN — Timer Enable
0 = Disable count
1 = Enable count

TSWAI — Stop in WAIT Mode
0 = Continue running in WAIT mode
1 = Disable the timer module in WAIT mode

TSFRZ — Stop in Freeze Mode
0 = Continue running in freeze mode
1 = Disable the timer module in freeze mode

TFFCA — Timer Fast Flag Clear All
0 = Timer flag clearing functions normally
1 = Allow fast flag clearing:
For TFLG1 register, the corresponding channel flag CnF is cleared by read from input capture/write to output compare.
For TFLG2 register, the timer overflow flag TOF is cleared by any access to the timer count register TCNT.
For the pulse accumulator flag register PAFLG, any access to the PACNT clears the PAOVF and PAIF,.

The Timer Module
Timer System Control Register2 (TSCR2)

Address Base+$000D	7	6	5	4	3	2	1	0
R	TOI	0	0	0	TCRE	PR2	PR1	PR0
W								
Reset	0	0	0	0	0	0	0	0

TOI — Timer Overflow Interrupt Enable

0 = Disable timer overflow interrupt

1 = Eable timer overflow interrupt

TCRE — Timer Counter Reset Enable

0 = Counter runs free

1 = Counter resets by a successful output compare 7 (used to up count to a value other than $FFFF)

PR[2:0] — Timer Prescaler Select

PR2	PR1	PR0	Timer Clock
0	0	0	Bus Clock / 1
0	0	1	Bus Clock / 2
0	1	0	Bus Clock / 4
0	1	1	Bus Clock / 8
1	0	0	Bus Clock / 16
1	0	1	Bus Clock / 32
1	1	0	Bus Clock / 64
1	1	1	Bus Clock / 128

Interrupt Configuration
Timer Interrupt Enable/Flag Registers (TIE/TFLG1&2)

Base+$000C	7	6	5	4	3	2	1	0
R W	C7I	C6I	C5I	C4I	C3I	C2I	C1I	C0I
Reset	0	0	0	0	0	0	0	0

C[7:0]I — Enable Input Capture/Output Compare channel x flag, CxF in TFLG1 register, to cause interrupt

Base+$000E	7	6	5	4	3	2	1	0
R W	C7F	C6F	C5F	C4F	C3F	C2F	C1F	C0F
Reset	0	0	0	0	0	0	0	0

C[7:0]F — Input Capture/Output Compare channel x flag

- Set when an input capture or output compare event occurs
- Cleared by: Writing 1 to the corresponding flag
  Read from input capture/Write to output compare channel while TFFCA is set.

Base+$000F	7	6	5	4	3	2	1	0
R	TOF	0	0	0	0	0	0	0
W								
Reset	0	0	0	0	0	0	0	0

TOF — Timer Overflow flag

- Set when 16-bit free-running timer overflows from $FFFF to $0000
- Cleared by writing 1 to the flag

The Pulse Accumulator (PACNT) Functions

The pulse accumulator (PACNT) is a 16-bit counter that can operate in two modes (The PAMOD bit selects the mode of operation):

1- Event counter mode :

Counting the number of occurrence of an event (edges of selected polarity on the pulse accumulator input pin, PT7).

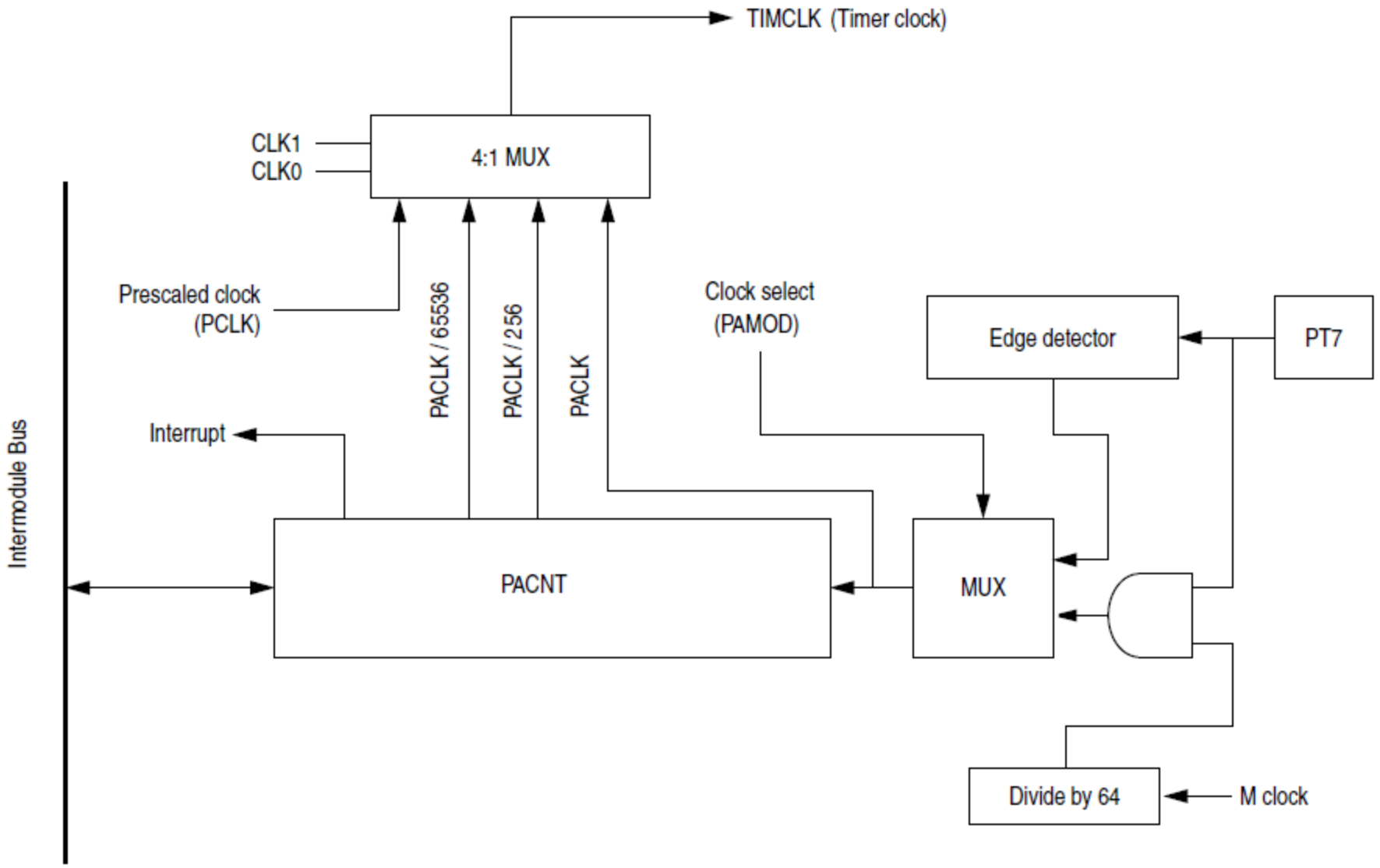
When this count is given in a specified time interval it represents the frequency of that event.

2- Gated time accumulation mode:

Counting pulses from a divide-by-64 clock (ECLK/64) while PT7 is active.

- The Pulse Accumulator system is enabled by bit PAEN in the Pulse Accumulator Control Register (PACTL).
- The counter value can be read from the Pulse Accumulator Counter Registers (PACNTH/L).
- An overflow flag (PAOVF) in the Pulse Accumulator Flag Register (PAFLG) is set when the counter rolls over from $FFFF to $0000.
- If The pulse accumulator overflow interrupt enable bit, in the Pulse Accumulator Control (PACTL) register is set, the PAFLG causes an interrupt.
-The minimum pulse width for the PAI input is greater than two bus clocks.

The Pulse Accumulator (PCNT) Block Diagram



The Pulse Accumulator (PACNT)
Pulse Accumulator Control Register (PACTL)

Address Base+$0020	7	6	5	4	3	2	1	0
R	0	PAEN	PAMOD	PEDGE	CLK1	CLK0	PAOVI	PAI
W								
Reset	0	0	0	0	0	0	0	0

PAEN — Enable PACT

PAMOD — Pulse Accumulator Mode (Active only when PAEN=1)

0 = Event counter mode

1 = Gated time accumulator mode

PEDGE — Pulse Accumulator Edge Control (Active only when PAEN=1)

For event counter mode (PAMOD=0):

0 — Increment count on falling edges on IOC7 pin.

1 — Increment count on rising edges on IOC7 pin.

For gated time accumulation mode (PAMOD=1).

0 — IOC7 input pin high enables bus clock/64 to Pulse Accumulator. The trailing falling edge sets the PAIF flag.

1 — IOC7 input pin low enables bus clock/64 to Pulse Accumulator. The trailing rising edge sets the PAIF flag.

CLK[1:2] — Clock Select Bits

CLK1	CLK0	Timer Clock
0	0	Use timer prescaler clock as timer counter clock
0	1	Use PACLK as input to timer counter clock
1	0	Use PACLK/256 as timer counter clock frequency
1	1	Use PACLK/65536 as timer counter clock frequency

PAOVI — Pulse Accumulator Overflow Interrupt Enable

0 — PA overflow interrupt disabled

1 — PA overflow interrupt enabled

PAI — Pulse Accumulator Input Interrupt Enable

0 — PA input interrupt disabled

1 — PA input interrupt enabled

The Pulse Accumulator (PACNT)
Pulse Accumulator Flag Register (PAFLG)

Address Base+$0021	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	PAOVF	PAIF
W								
Reset	0	0	0	0	0	0	0	0

PAOVF — PA Overflow Flag

Set when PA overflows from $FFFF to $0000.
Cleared by writing 1 to this flag
Cleared also when Timer Fast Flag Clear (TFFCA) in TSCR is set and the PA count (PACNT) is accessed

PAIF — Pulse Accumulator Input Flag

For event counter mode (PAMOD=0):
Set when selected edge is detected on IOC7 pin.
For gated time accumulation mode (PAMOD=1):
Set by the trailing edge of the gate signal on IOC7 pin
Cleared by writing 1 to this flag.
Cleared also when Timer Fast Flag Clear (TFFCA) in TSCR is set and the PA count (PACNT) is accessed

---

Pulse Accumulator Count Register (PACNT)

Pulse Accumulator Count Register High (PACNTH): Module Base + $0022
Pulse Accumulator Count Register low (PACNTL): Module Base + $0023

PACNTH and PACNTL can be accessed separately, but recommended to be accessed by as a whole word.

Pulse Accumulator Example
Time Delay Subroutine

Example: Write a subroutine to generate a time delay that is multiple of 1ms. The number of ms is passed in Y. The E-clock frequency is 24MHz.

Solution: Select the prescaler 8 to make the timer clock period 8÷(24MHz) = $\frac{1}{3}$ µs, so that a 1 ms is made up by 3000 clock cycles.

```
delay_1ms               ; number of ms is passed in reg. y
      pshd              ; save register d
      movb  #$90,TSCR1  ; enable TCNT and TFFCA
      movb  #$03,TSCR2  ; configure prescaler to 8
      bset  TIOS,OC0    ; enable OC0
again0
      ldd   TCNT
      addd  #3000       ; start an OC operation
      std   TC0         ; with 1-ms time delay
wait_lp0                ; wait for OC0 flag
      brclr TFLG1,OC0,wait_lp0
      ldd   TC0
      dbne  y,again0    ; dec. num. of ms & loop if not 0
      puld              ; if done, resume reg. d & return
      rts
```

```
#include “c:\cwHSC12\include\hcs12.h”
void delayby1ms(int k)
{
int ix;
TSCR1 = 0x90;   /* enable TCNT and fast timer flag clear */
TSCR2 = 0x03;   /* disable timer interrupt, set prescaler to 8*/
TIOS |= OC0;    /* enable OC0 */
TC0 = TCNT + 3000;
for(ix = 0; ix < k; ix++)
{
      while(!(TFLG1 & C0F));
      TC0 += 3000;
}
TIOS &= ~OC0; /* disable OC0 */
}
```

Note on Pulse Accumulator Operation

The PACNT input and timer channel 7 use the same pin IOC7.

To use the IOC7 pin for event counting:

- clear OM7 and OL7 bits in TCTL1/TCTL2 registers to disconnect the pin from the output logic
- clear OC7M7 in OC7M register to disable the output compare 7 mask operation

Input Capture Function

Captures the time at which an external event occurs by latching the contents of the timer into a latch.
When an active edge occurs on the pin of an input capture channel, the value in the timer counter is transferred into the timer channel registers TCx.

Each input capture channel includes:

- a 16-bit register
- an input pin
- Input edge detection circuit
- interrupt generation circuit

Applications of input capture function:

- recording event time
- pulse width/duty cycle measurement
- measuring signal periodic time
- timing interface

Input Capture Channel Configuration

To configure channel x as an input capture channel, clear I/O select bit, IOSx.

Timer control registers 3 and 4 are used to specify what signal edge to capture:
two bits for each channel (rising/falling/both edges).
If both bits are zero, the capture function is disabled and the corresponding channel can be used for GPIO.

An input capture event on channel x sets the CxF flag in the TFLG1 $008E which can be polled by the program.
If the corresponding interrupt mask bit CxI in the interrupt enable register TIE is set to 1, the CxF flag generates an interrupt request.

To clear a flag in the TFLG1, write a *one* to it.
A better way to clear the flag is by setting the timer fast flag clear TFFCA of the TSCR1 register which causes the flag to clear when reading the corresponding input capture register (or writing into the output compare register).

The minimum pulse width for the input capture input is greater than two bus clocks.

Input Capture Channel Configuration
Timer Input Capture/Output Compare Select Register (TIOS).

To configure channel x as an input capture channel, clear I/O select bit, IOSx in the Timer Input Capture/Output Compare Select Register (TIOS).

Address Base+$0000	7	6	5	4	3	2	1	0
R W	IOS7	IOS6	IOS5	IOS4	IOS3	IOS2	IOS1	IOS0
Reset	0	0	0	0	0	0	0	0

IOS[7:0] — I/O Select Bits

0 — The corresponding channel acts as an input capture

1 — The corresponding channel acts as an output compare

Input Capture Channel Configuration
Timer Control Register3 & 4 (TCTL3/TCTL4)

Address Base+$000A	7	6	5	4	3	2	1	0
R W	EDG7B	EDG7A	EDG6B	EDG6A	EDG5B	EDG5A	EDG4B	EDG4A
Reset	0	0	0	0	0	0	0	0

Address Base+$000B	7	6	5	4	3	2	1	0
R W	EDG3B	EDG3A	EDG2B	EDG2A	EDG1B	EDG1A	EDG0B	EDG0A
Reset	0	0	0	0	0	0	0	0

Input Capture Edge Control — Each pair of bits configures the corresponding input capture edge detector circuit.

EDGnB	EDGnA	Configuration
0	0	Capture disabled
0	1	Capture on rising edges only
1	0	Capture on falling edges only
1	1	Capture on any edge (rising or falling)

Input Capture Channel Example
Measuring a Signal Time Period

Example:

Use the input capture channel 0 to measure the period of an unknown signal.

The period is known to be shorter than 128 ms.

Assume that the E-clock frequency is 24 MHz (E-clock period 1÷24 μs).

Use the number of clock cycles as the unit of the period.

Solution:

Since the input-capture register is 16-bit, the longest period of the signal that can be measured with the prescaler to TCNT set to 1 is:

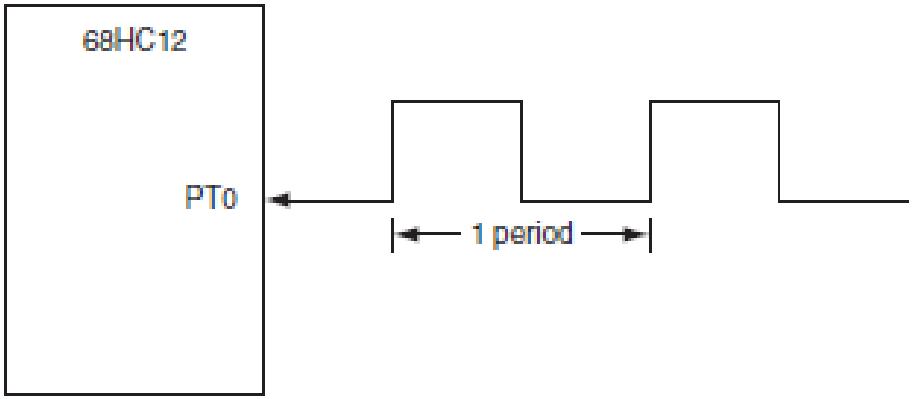
$$2^{16} \times (1 \div 24\ \mu s) \approx 2.73\ ms$$

To measure a period that is equal to 128 ms, we have two options.

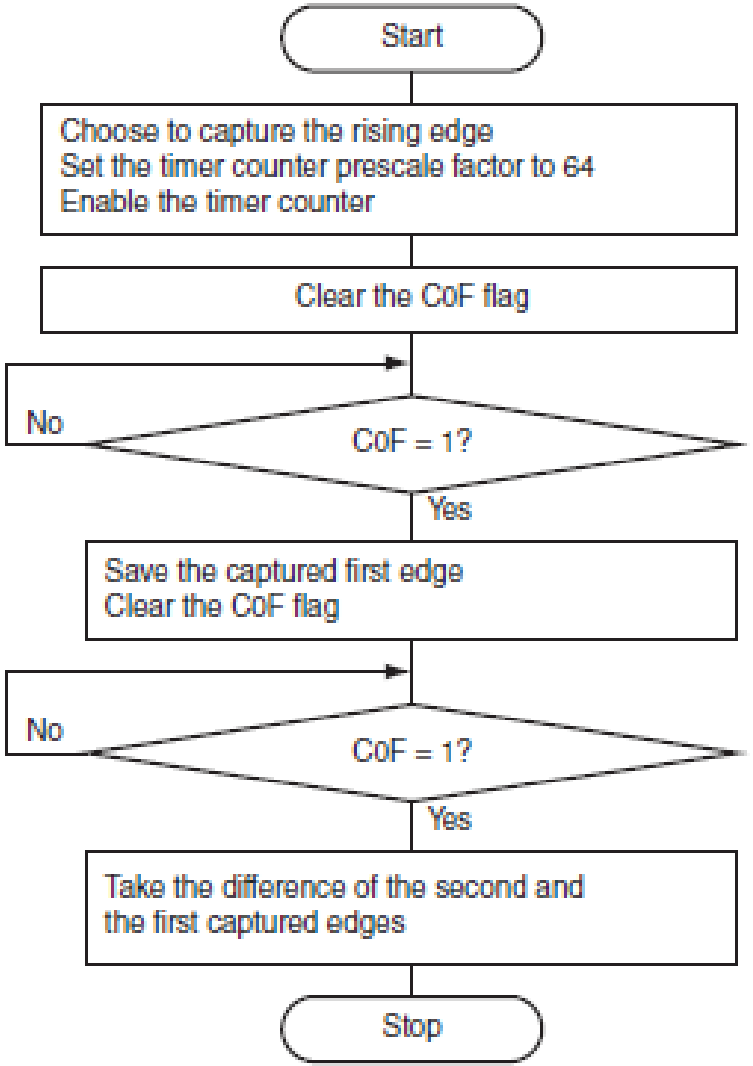
1. Set the prescale factor to 1 and keep track of the number of times that the timer counter overflows.
2. Set the prescale factor to 64 and do not keep track of the number of times that the timer counter overflows. $2^{16} \times 64(1 \div 24\ \mu s) \approx 174.763\ ms > 128\ ms$

Adopting the second approach to makes the programming easier.

Input Capture Channel Example (cont.)
Measuring a Signal Time Period



The period of this signal is the time between two successive rising(or falling) edges



Input Capture Channel Example (cont.)
Measuring a Signal Time Period

```
#include    "c:\miniide\hcs12.inc"
      org     $1000
edge_1st  rmb   2       ; memory to hold the first edge
Period    rmb   2       ;memory to store the period
      org     $1500
      movb    #$90,TSCR1  ;enable TCNT and TFFCA
      bclr    TIOS,IOS0   ;enable input-capture 0
      movb    #$06,TSCR2  ;disable TCNT overflow interrupt
                          ;set prescaler to 64
      movb    #$01,TCTL4 ; capture the rising edge on PT0
      movb    #C0F,TFLG1 ; clear the C0F flag
      brclr   TFLG1,C0F,* ; wait for the first rising edge
      ldd     TC0         ; save the first edge and clear C0F
      std     edge_1st
      brclr   TFLG1,C0F,* ; wait for the second rising edge
      ldd     TC0
      subd    edge_1st    ; compute the period
      std     period
      swi
      end
```

```
#include "c:\cwHCS12\include\hcs12.h"
void main(void)
{
unsigned int edge1, period;
TSCR1 = 0x90;       // enable TCNT and TFFCA
TIOS &= ~IOS0;      // enable input-capture 0
TSCR2 = 0x06;       // disable TCNT overflow interrupt,
                    // set prescaler to 64
TCTL4 = 0x01;       // capture the rising edge on PT0 pin
TFLG1 = 0x01;       // clear the C0F flag */
while (!(TFLG1 & C0F)); // wait for the  the first rising edge
edge1 = TC0;        // save the first edge and clear C0F flag
while (!(TFLG1 & C0F)); // wait for the second rising edge
period = TC0 2 edge1;
while(1);
}
```

Output Compare Function

Generates a time delay, trigger an action at some future time

The key to using the output compare function is to:

- read the value in the timer
- add a delay to it
- store the sum back in the output compare register
- after a time period specified by the delay (when the output compare register and the timer value are equal):
  1. the specified action on the output pin is activated (set/clear/toggle)
  2. the CxF flag in the timer interrupt flag 1 register is set.
  3. if the corresponding interrupt mask bit CxI in TMSK1 is set to 1, the CxF flag generates an interrupt request.

The output compare channel consists of:

- 16-bit register TCx (same register used for input capture)
- 16-bit comparator
- Output pin (same pin used for input capture or GPIO)
- Interrupt circuit
- Forced output compare function (FOCx bits of CFORC register)

Output Compare Channel Configuration

Output compare and input capture functions share the same pins and registers so they cannot be enabled simultaneously, when one is enabled, the other is disabled.

The selection is done by the I/O Select Register (IOS).
To configure channel x as an output compare channel, set I/O select bit, IOSx.

The action on the output compare pin (pull-up, pull-down, toggle) can be selected by programming the TCTL1 and TCTL2 registers.

Output Compare Channel Example
Time Delay Subroutine

Example: Write a subroutine to generate a time delay that is multiple of 1ms. The number of ms is passed in Y. The E-clock frequency is 24MHz.

Solution: Select the prescaler 8 to make the timer clock period $8 \div (24\text{MHz}) = \frac{1}{3}$ µs, so that a 1 ms is made up by 3000 clock cycles.

```
delay_1ms               ; number of ms is passed in reg. y
      pshd              ; save register d
      movb  #$90,TSCR1  ; enable TCNT and TFFCA
      movb  #$03,TSCR2  ; configure prescaler to 8
      bset  TIOS,OC0    ; enable OC0
      ldd   TCNT
again0
      addd  #3000       ; start an OC operation
      std   TC0         ; with 1-ms time delay
wait_lp0                ; wait for OC0 flag
      brclr TFLG1,OC0,wait_lp0
      ldd   TC0
      dbne  y,again0    ; dec. num. of ms & loop if not 0
      puld              ; if done, resume reg. d & return
      rts
```

```
#include “c:\cwHSC12\include\hcs12.h”
void delayby1ms(int k)
{
int ix;
TSCR1 = 0x90;   /* enable TCNT and fast timer flag clear */
TSCR2 = 0x03;   /* disable timer interrupt, set prescaler to 8*/
TIOS |= OC0;    /* enable OC0 */
TC0 = TCNT + 3000;
for(ix = 0; ix < k; ix++)
{
      while(!(TFLG1 & C0F));
      TC0 += 3000;
}
TIOS &= ~OC0; /* disable OC0 */
}
```

Output Compare Channel Configuration
Timer Control Register1 & 2 (TCTL1/TCTL2)

Address Base+$0008	7	6	5	4	3	2	1	0
R W	OM7	OL7	OM6	OL6	OM5	OL5	OM4	OL4
Reset	0	0	0	0	0	0	0	0

Address Base+$0009	7	6	5	4	3	2	1	0
R W	OM3	OL3	OM2	OL2	OM1	OL1	OM0	OL0
Reset	0	0	0	0	0	0	0	0

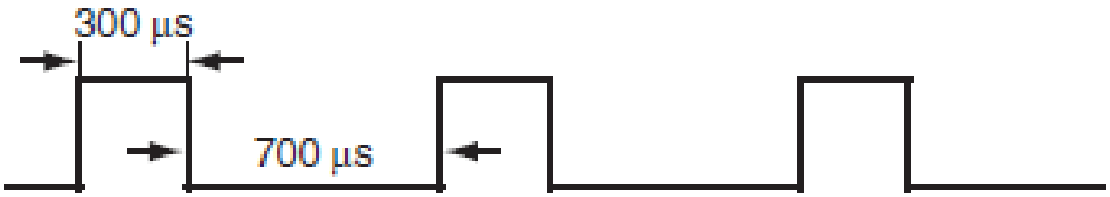
Each pair of bits specifies the output action to be taken as a result of a successful OCx compare. When either OMx or OLx is 1, the pin associated with OCx becomes an output tied to OCx.

Note: To enable output action by OMx bits on timer port, the corresponding bit in OC7M should be cleared.

OMx	OLx	Action
0	0	Timer disconnected from output pin logic
0	1	Toggle OCx output line
1	0	Clear OCx output line to zero
1	1	Set OCx output line to one

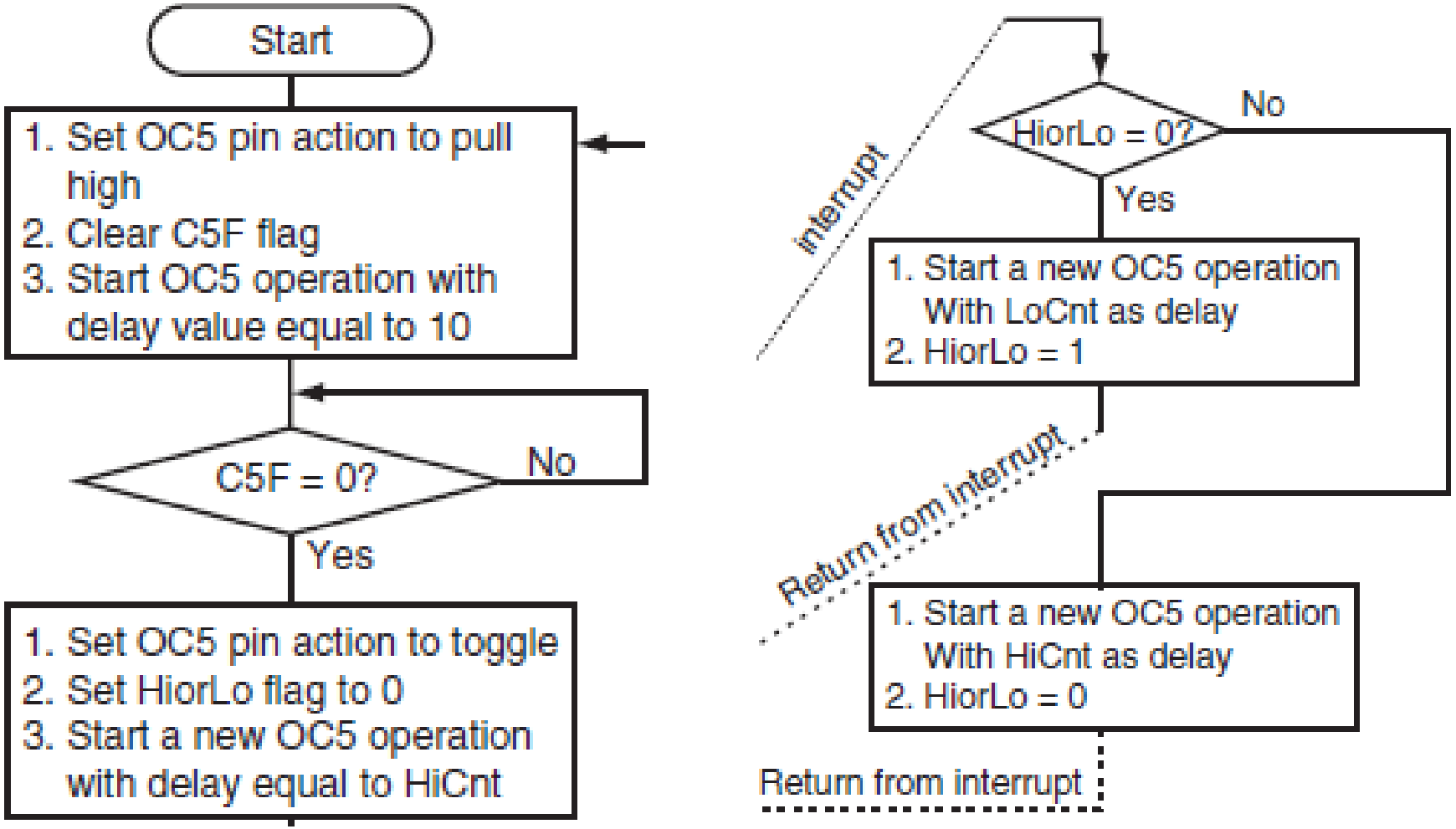
Output Compare Channel Example
Generating Digital Waveform

Example:
Generate a 1-kHz digital waveform with a 30% duty cycle from the PT5 pin.
Use the interrupt-driven by the success of the output-compare operation.
The frequency of the E-clock is 24 MHz.



Solution:
If we set the prescale factor to 8, the period of the clock to TCNT is $8 \div 24 = 1/3$ µs.
Then the intervals of the high period is 900 clock cycles
the interval of the low period is 2100 clock cycles.

Output Compare Channel Example (cont.)
Generating Digital Waveform



Output Compare Channel Example (cont.)
Generating Digital Waveform (in Assembly)

```
#include "c:\miniide\hcs12. inc"
HiCnt equ 900      ; delay count for high interval of the waveform
LoCnt equ 2100     ; delay count for low interval of the waveform
       org      $1000
HIorLO  ds.b    1              ; flag to select HiCnt (1) or LoCnt (0)
       org      $1500
       lds      #$1500         ; establish stack pointer
       movw     #OC5ISR,UserTimerCh5      ;set up OC0 int. vector
       movb     #$90,TSCR1     ; enable TCNT and TFFCA
       movb     #$03,TSCR2     ; set TCNT clock prescaler to 8
       bset     TIOS,BIT5      ; enable OC5
       movb     #$0C,TCTL1  ; configure  OC5 pin action to pull high
       ldd      TCNT           ; start OC5 with delay count equal to 10
       addd     #10            ; and pin action to pull high
       std      TC5
       brclr    TFLG1,C5F,*    ; wait for C5F=1 and PT5 pin pulled hi
       ldd      TC5            ; start another OC5 operation with
       addd     #HiCnt         ; delay count set to HiCnt
       std      TC5            ; "
       movb     #$04,TCTL1     ; change pin action to toggle
       clr      HIorLO      ; LoCnt will be the delay count next time
       bra      $              ; prepare to perform other operations
```

```
OC5ISR
            tst     HIorLO    ; which delay count should be added?
            beq     addLow    ; if 0 then select LoCnt
            ldd     TC5       ; select HiCnt as the delay count for
            addd    #HiCnt    ; the new OC5 operation
            std     TC5       ;   "
            clr     HIorLO    ; toggle HIorLo flag
            rti
addLow ldd      TC5           ; select LoCnt as the delay count for
            addd    #LoCnt    ; the new OC5 operation
            std     TC5       ;   "
            movb    #1,HIorLO     ; toggle HIorLO flag
            rti
            end
```

Output Compare Channel Example (cont.)
Generating Digital Waveform (in C)

```
#include “c:\cwHCS12\include\hcs12.h”
#define HiCnt 1200
#define LoCnt 1800
char HiorLo;
void main (void)
{
TSCR1 5 0x90; // enable TCNT and fast timer flag clear
TSCR2 5 0x03; // disable TCNT interrupt, set prescaler to
      8
TIOS |5 OC5; // enable OC5 function
TCTL1 5 0x0C; // set OC5 action to pull high
TFLG1 5 0xFF; // clear all CxF flags
TC5 5 TCNT 1 10;
while(TFLG1 & C5F); // wait until C5F is set
TCTL1 5 0x04; // set OC5 pin action to toggle
TC5 15 HiCnt; // start an new OC5 operation
HiorLo 5 0; // add LoCnt for the next OC5 operation
TIE 5 0x20; // enable OC5 interrupt locally
asm(“cli”); // enable interrupt globally
while(1);
}
```

```
interrupt void oc5ISR (void)
{
        if(HiorLo){
        TC5 15 HiCnt;
        HiorLo = 0;
        }
        else{
        TC5 += LoCnt;
        HiorLo 5 1;
        }
}
```

Output Compare Channel Configuration

Output Compare 7 Mask & Data registers (OC7M/OC7D)

Address Base+$0002	7	6	5	4	3	2	1	0
R W	OC7M7	OC7M6	OC7M5	OC7M4	OC7M3	OC7M2	OC7M1	OC7M0
Reset	0	0	0	0	0	0	0	0

Address Base+$0003	7	6	5	4	3	2	1	0
R W	OC7D7	OC7D6	OC7D5	OC7D4	OC7D3	OC7D2	OC7D1	OC7D0
Reset	0	0	0	0	0	0	0	0

A successful output compare on channel 7 (OC7) can cause the bit OC7D[6:0] in the Output Compare 7 Data register to transfer to output pin IOCx if the corresponding bit OC7Mx in the Output Compare 7 Mask register is set.

A successful channel 7 output compare overrides any OC[6:0] compares.

Useful in applications that require multiple actions to be triggered simultaneously in the near future.

Output Compare Channel Example
Triggering multiple actions simultaneously

Example: An application requires the following operations to be triggered 50 ms later:

- Turn off the light controlled by the TC5 pin
- Turn on the temperature sensor controlled by the TC4 pin
- Turn off the heater controlled by the TC3 pin
- Turn on the music controlled by the TC2 pin

Write an instruction sequence to perform the desired operation.

Solution: To select output channels TC[5,4,3,2] we write $3C to the Output Compare 7 Mask OC7M register.

To turn TC5 off, TC4 on, TC3 off, and TC2 on we write $34 to the Output Compare 7 Data OC7D register .

By selecting the prescaler 64 we make a timer clock period 64/24MHz=8/3µs. To obtain a 50ms delay we need 50,000/(8/3)=18,750<2^{16} clocks.

```
        movb    #$90,TSCR1      ; enable TCNT and TFFCA
        movb    #$06,TSCR2      ; set prescaler to 64
        movb    #$3C,OC7M       ; allow OC7 to control TC5, TC4, TC3, and TC2 pins
        movb    #$34,OC7D       ; set pin actions for TC5, TC4, TC3, and TC2
        ldd     TCNT            ; start an OC7 operation with 50 ms as delay
        addd    #18750
        std     TC7             ; start the OC7 operation
```

Output Compare Channel Configuration
Timer Compare Force Register (CFORC)

To force the event which is programmed for output compare x to occur immediately, write to this register with the corresponding data bit(s) set.
The action taken is the same as if a successful comparison had just taken place with the TCx register except the interrupt flag does not get set.

Address Base+$0001	7	6	5	4	3	2	1	0
R W	FOC7	FOC6	FOC5	FOC4	FOC3	FOC2	FOC1	FOC0
Reset	0	0	0	0	0	0	0	0

Output Compare Channel Configuration
Timer Toggle on Overflow Register (TTOV)

Address Base+$0007	7	6	5	4	3	2	1	0
R W	TOV7	TOV76	TOV5	TOV4	TOV3	TOV2	TOV1	TOV0
Reset	0	0	0	0	0	0	0	0

Toggle on Overflow Bits — TOVx toggles output compare pin on overflow.
When set, it takes precedence over forced output compare but not channel 7 override events.

0 — Disable toggle output compare pin on overflow feature
1 — Enable toggle output compare pin on overflow feature