EECE 218
Microcontrollers

Timers
Motivation

- How to do….
  - Precise time delay in programs
  - Time delay measurement
  - Period and pulse width measurement
  - Frequency measurement
  - Event counting
  - Arrival time comparison
  - Time-of-day tracking
  - Periodic interrupt generation
  - Waveform generation

→ Complex/difficult to do using basic operations/bit I/O
The HCS12 Timer System

- The HCS12 has a standard timer module (TIM) that consists of:
  - Eight channels of multiplexed input capture and output compare functions.
  - 16-bit pulse accumulator A
  - 16-bit timer counter
- The HCS12 devices in the automotive family have implemented an Enhanced Capture Timer module (ECT). The ECT module contains:
  - All the features contained in the TIM module
  - One 16-bit buffer register for each of the input capture channels
  - Four 8-bit pulse accumulator
  - A 16-bit Modulus Down Counter with 4-bit prescaler
  - Four user selectable delay counters for increasing input noise immunity
- The TIM (of course ECT also) shares the eight Port T pins (IOC0…IOC7).
The HCS12 Timer System

Figure 8.1 HCS12 Standard Timer (TIM) block diagram
The HCS12 Timer System - Details
The Main Timer Counter
Timer Counter Register (TCNT)

- Free-running, 16 bit counter, driven by the pre-scaled bus clock
  » Pre-scaler can ‘slow down’ TCNT
- Required for input capture and output compare functions
- Must be accessed in one 16-bit operation in order to obtain the correct value
  » LDD, etc.
- Three other registers related to the operation of the TCNT: TSCR1, TSCR2, TFLG2.
Timer System Control Register 1 (TSCR1)

- **TEN**: Setting and clearing the bit 7 of TSCR1 will start and stop the counting of the TCNT.
- **TSWAI/TSFRZ**: special modes
- **TFFCA**: Setting the bit 4 will enable fast timer flag clear function. If this bit is clear, then the user must write a one to a timer flag in order to clear it.

  ➔ This is related to ‘Interrupt Acknowledge’!

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEN</td>
<td>TSWAI</td>
<td>TSFRZ</td>
<td>TFFCA</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**value after reset**

- **TEN** -- timer enable bit
  - 0 = disable timer; this can be used to save power consumption
  - 1 = allows timer to function normally
- **TSWAI** -- timer stops while in wait mode bit
  - 0 = allows timer to continue running during wait mode
  - 1 = disables timer when MCU is in wait mode
- **TSFRZ** -- timer and modulus counter stop while in freeze mode
  - 0 = allows timer and modulus counter to continue running while in freeze mode
  - 1 = disables timer and modulus counter when MCU is in freeze mode
- **TFFCA** -- timer fast flag clear all bit
  - 0 = allows timer flag clearing to function normally
  - 1 = For TFLG1, a read from an input capture or a write to the output compare channel causes the corresponding channel flag, CnF, to be cleared. For TFLG2, any access to the TCNT register clears the TOF flag. Any access to the PACN3 and PACN2 registers clears the PAOVF and PAIF flags in the PAFLG register. Any access to the PACN1 and PACN0 registers clears the PBOVF flag in the PBFLG register.

  ➔ This is related to ‘Interrupt Acknowledge’!
Timer System Control Register 2
(TSCR2)

- **TOI**: Bit 7 is the TCNT overflow interrupt enable bit.
- **TCRE**: TCNT can be reset to 0 when TCNT equals TC7 by setting bit 3 of TSCR2.
- **PR2-0**: The clock input to TCNT can be pre-scaled by a factor selecting by bits 2 to 0 of TSCR2.

<table>
<thead>
<tr>
<th>value after reset</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOI</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>TCRE</td>
<td>PR2</td>
<td>PR1</td>
<td>PR0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

TOI = timer overflow interrupt enable bit
0 = interrupt inhibited
1 = interrupt requested when TOF flag is set
TCRE = timer counter reset enable bit
0 = counter reset inhibited and counter free runs
1 = counter reset by a successful output compare 7
If TC7 = $0000$ and TCRE = 1, TCNT stays at $0000$ continuously. If TC7 = $FFFF$ and TCRE = 1, TOF will never be set when TCNT rolls over from $FFFF$ to $0000$.

Figure 8.3 Timer system control register 2 (TSCR2)

<table>
<thead>
<tr>
<th>PR2</th>
<th>PR1</th>
<th>PR0</th>
<th>Prescale Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>32</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>64</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>128</td>
</tr>
</tbody>
</table>

Table 8.1 Timer counter prescale factor
Timer Interrupt Flag 2 Register (TFLG2)

- TFLG2 indicates ‘timer status’
- Only bit 7 (TOF) is implemented.
- Bit 7 will be set whenever TCNT overflows ($FFFF->$0). This *may* cause an interrupt.
- To clear TOF: (a) write a 1 into it or (b) use TFFCA and access TCNT
  » This is the method to acknowledge the Timer Overflow Interrupt.
Input Capture Functions

- Physical time is often represented by the contents of the main timer.
- The occurrence of an event is represented by a signal edge (rising or falling edge).
- The time when an event occurs is recorded by latching the count of the main timer when a signal edge arrives.
- The HCS12 has eight input capture channels. Each channel has a 16-bit capture register, an input pin, edge-detection logic, and interrupt generation logic.
- Input capture channels share most of the circuit with output compare functions. For this reason, they cannot be enabled simultaneously.

![Figure 8.4 Events represented by signal edges](image)

**Figure 8.4 Events represented by signal edges**
Input Capture Functions
The selection of input capture and output compare is done by programming the TIOS register.

The contents of the TIOS register are shown below. Setting a bit selects the output compare function. Otherwise, the input capture function is selected.

<table>
<thead>
<tr>
<th></th>
<th>IOS7</th>
<th>IOS6</th>
<th>IOS5</th>
<th>IOS4</th>
<th>IOS3</th>
<th>IOS2</th>
<th>IOS1</th>
<th>IOS0</th>
</tr>
</thead>
<tbody>
<tr>
<td>value after reset</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

IOS[7:0] -- Input capture or output compare channel configuration bits  
0 = The corresponding channel acts as an input capture  
1 = The corresponding channel acts as an output compare

The following instruction will enable the output compare channels 7...4 and input capture channel 3...0:

```assembly
movb #$F0, TIOS
```
Timer Port Pins

- PORTT: Each port pin can be used as a general I/O pin when timer function is not selected.
- Special: Pin 7 can be used as input capture 7, output compare 7 action, and pulse accumulator input.
- When a timer port pin is used as a general I/O pin, its direction is configured by the DDRT register.
### Timer Control Register 3 and 4

- The signal edge to be captured is selected by TCTL3 and TCTL4.
- The edge to be captured is selected by two bits. The user can choose to capture the rising edge, falling edge, or both edges.

<table>
<thead>
<tr>
<th>EDG7B</th>
<th>EDG7A</th>
<th>EDG6B</th>
<th>EDG6A</th>
<th>EDG5B</th>
<th>EDG5A</th>
<th>EDG4B</th>
<th>EDG4A</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

(a) Timer control register 3 (TCTL3)

<table>
<thead>
<tr>
<th>EDG3B</th>
<th>EDG3A</th>
<th>EDG2B</th>
<th>EDG2A</th>
<th>EDG1B</th>
<th>EDG1A</th>
<th>EDG0B</th>
<th>EDG0A</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

(b) Timer control register 4 (TCTL4)

EDGnB EDGnA -- Edge configuration

- 0 0: Capture disabled
- 0 1: Capture on rising edges only
- 1 0: Capture on falling edges only
- 1 1: Capture on both edges

Figure 8.5 Timer control register 3 and 4
Timer Interrupt Flag 1 Register (TFLG1)

- Whenever a signal edge arrives, the associated timer (interrupt) flag will be set.

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>C7F</td>
<td>C6F</td>
<td>C5F</td>
<td>C4F</td>
<td>C3F</td>
<td>C2F</td>
<td>C1F</td>
<td>C0F</td>
</tr>
</tbody>
</table>

reset: 0 0 0 0 0 0 0 0

CnF: input capture/output compare interrupt flag bits
0 = interrupt condition has not occurred
1 = interrupt condition has occurred

Figure 8.8 Timer interrupt flag register 1 (TFLG1)
The arrival of a signal edge (and the setting of the CxF bit) may \textit{optionally} generate an interrupt to the CPU.

The enabling of the interrupt is controlled by the Timer Interrupt Enable Register.

![Figure 8.7 Timer interrupt enable register (TIE)]
How to Clear a Timer Flag Bit

- **Why?** Must ‘tell’ the timer that the software has processed event.

- **In normal** mode, write a 1 to the flag bit to be cleared.
  - **Method 1**
    - Use the BCLR instruction with a 0 at the bit position (s) corresponding to the flag (s) to be cleared. For example,
      
      ```
      BCLR TFLG1, $FE
      ```
      
      will clear the C0F flag.
  - **Method 2**
    - Use the movb instruction with a 1 at the bit position (s) corresponding to the flag (s) to be cleared. For example,
      
      ```
      movb #$01,TFLG1
      ```
      
      will clear the C0F flag.

- **When fast timer flag clear function is enabled (in TSCR1)**
  - (re TFLG1): read from input capture/write to output compare
  - (re TFLG2): access to TCNT
  - (re Pulse Acc): access to PACN_ register/s
  
  ... will **automatically** clear the relevant flag.
Applications of Input Capture Function

- Event arrival time recording
- Period measurement: need to capture the main timer values corresponding to two consecutive rising or falling edges

- Pulse width measurement: need to capture the rising and falling edges

Figure 8.9 Period measurement by capturing two consecutive edges

Figure 8.10 Pulse-width measurement using input capture
Input Capture

- **Interrupt generation**: Each input capture function can be used as an edge-sensitive interrupt source.
- **Event counting**: count the number of signal edges arrived during a period

![Diagram of input capture](image)

**Figure 8.11 Using an input-capture function for event counting**

- **Time reference**: often used in conjunction with an output compare function

![Diagram of time reference](image)

**Figure 8.12 A time reference application**
Duty Cycle Measurement

\[ \text{duty cycle} = \frac{\Delta T}{T} \times 100\% \]

Figure 8.13 Definition of duty cycle
Phase Difference Measurement

Figure 8.14 Phase difference definition for two signals

\[
\text{phase difference} = \frac{\Delta T}{T} \times 360^\circ
\]
Period Measurement

- Use the IC0 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. 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:

\[ 2^{16} \div 24 \text{ MHz} = 2.73 \text{ ms}. \]

- To measure a period that is equal to 128 ms, we have two options:
  - Set the pre-scale factor to 1 and keep track of the number of times the timer counter overflows.
  - Set the pre-scale factor to 64 and do not keep track of the number of times the timer counter overflows.

- We will set the pre-scale factor to TCNT to 64.
Period Measurement

Start

Choose to capture the rising edge
Set the timer counter prescale factor to 16
Enable the timer counter

Clear the C0F flag

C0F = 1?

Saved the captured first edge
Clear the C0F flag

C0F = 1?

Take the difference of the second and the first captured edges

Stop

Figure 8.16 Logic flow of period measurement program
Assembly Program for Period Measurement

```assembly
org $1000
edge1 ds.b 2 ; memory to hold the first edge
period ds.b 2 ; memory to store the period
org $1500
movb #$90,TSCR1 ; enable timer counter and enable fast timer flags clear
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 of PT0 signal
movb #C0F,TFLG1 ; clear the C0F flag
brclr TFLG1,C0F,* ; wait for the arrival of the first rising edge
ldd TC0 ; save the first edge and clear the C0F flag
std edge1
brclr TFLG1,C0F,* ; wait for the arrival of the second edge
ldd TC0
subd edge1 ; compute the period
std period
swi
end
```
Pulse Width Measurement

- Write a program to measure the pulse width of a signal connected to the PT0 pin. The E clock frequency is 24 MHz.

**Solution:**
- Set the prescale factor to TCNT to 32. Use clock cycle as the unit of measurement.
- The pulse width may be longer than $2^{16}$ clock cycles. We need to keep track of the number of times that the TCNT timer overflows. Let
  - $ovcnt$ = TCNT counter overflow count
  - $diff$ = the difference of two consecutive edges
  - $edge1$ = the captured time of the first edge
  - $edge2$ = the captured time of the second edge
- The pulse width can be calculated by the following equations:

  **Case 1:** $edge2 \geq edge1$
  
  $\text{pulse width} = ovcnt \times 2^{16} + diff$

  **Case 2:** $edge2 < edge1$

  $\text{pulse width} = (ovcnt - 1) \times 2^{16} + diff$
Pulse width measurement

Figure 8.17 Logic flow for measuring pulse width of slow signals
Pulse width measurement

```
org $1000
edge1 ds.b 2
overflow ds.b 2
pulse_width ds.b 2
org $1500
movw #tov_isr,UserTimerOvf ; set up TCNT overflow interrupt vector
lds #$1500 ; set up stack pointer
movw #$0,overflow
movb #$90,TSCR1 ; enable TCNT and fast timer flag clear
movb #$05,TSCR2 ; disable TCNT interrupt, set prescaler to 32
bclr TIOS,IOS0 ; select IC0
movb #$01,TCTL4 ; capture rising edge
movb #C0F,TFLG1 ; clear C0F flag
wait1 brclr TFLG1,C0F,wait1 ; wait for the first rising edge
movw TC0,edge1 ; save the first edge & clear the C0F flag
movb #TOF,TFLG2 ; clear TOF flag
bset TSCR2,$80 ; enable TCNT overflow interrupt
cli
movb #$02,TCTL4 ; capture the falling edge on PT0 pin
wait2 brclr TFLG1,C0F,wait2 ; wait for the arrival of the falling edge
ldd TC0
subd edge1
```
Pulse width measurement

std               pulse_width
bcc               next ; is the second edge smaller?
ldx               overflow ; second edge is smaller, so decrement
dex               ; overflow count by 1
stx               overflow ; "
next              swi

tov_isr           movb #TOF,TFLG2 ; clear TOF flag
ldx               overflow
inx
stx               overflow
rti
end
Output Compare Function

- The HCS12 has eight output compare functions.
- Each output compare channel consists of
  - A 16-bit comparator
  - A 16-bit compare register TCx (also used as inout capture register)
  - An output action pin (PTx, can be pulled high, pulled low, or toggled)
  - An interrupt request circuit
  - A forced-compare function (CFOCx)
  - Control logic
Output Compare Function
Operation of the O/C Function

- To trigger an action (event) at a specific time in the future.
- To use an output-compare function, the user
  » Makes a copy of the current contents of the TCNT register
  » Adds to this copy a value equal to the desired delay
  » Stores the sum into an output-compare register (TCx, x = 0..7)
Operation of the O/C Function

- The actions that can be activated on an output compare pin include:
  - Pull up to high
  - Pull down to low
  - Toggle

- The action is determined by the Timer Control Register 1 & 2 (TCTL1 & TCTL2):

<table>
<thead>
<tr>
<th>Value after reset</th>
<th>OM7</th>
<th>OL7</th>
<th>OM6</th>
<th>OL6</th>
<th>OM5</th>
<th>OL5</th>
<th>OM4</th>
<th>OL4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

(a) TCTL1 register

<table>
<thead>
<tr>
<th>Value after reset</th>
<th>OM3</th>
<th>OL3</th>
<th>OM2</th>
<th>OL2</th>
<th>OM1</th>
<th>OL1</th>
<th>OM0</th>
<th>OL0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

(b) TCTL2 register

read: anytime
write: anytime

OMn  OLn : output level

0 0  no action (timer disconnected from output pin)
0 1  toggle OCN pin
1 0  clear OCN pin to 0
1 1  set OCN pin to high

Figure 8.18 Timer control register 1 and 2 (TCTL1 & TCTL2)
Operation of the O/C Function

- A successful compare will set the corresponding flag bit in the TFLG1 register.
- An interrupt may be optionally requested if the associated interrupt enable bit in the TIE register is set.
- Generate an active high 1 KHz digital waveform with 30 percent duty cycle from the PT0 pin. Use the polling method to check the success of the output compare operation. The frequency of the E clock is 24 MHz.

**Solution:** An active high 1 KHz waveform with 30 percent duty cycle is shown below.

- Setting the prescaler to the TCNT to 8, then the period of the clock signal to the TCNT will be 1/3 ms. The numbers of clock cycles that the signal is high and low are 900 and 2100, respectively.

![Waveform Diagram](image)

Figure 8.19 1 KHz 30 percent duty cycle waveform
Operation of the O/C Function

Figure 8.20 The program logic flow for digital waveform generation
Waveform generation

<table>
<thead>
<tr>
<th>hi_time</th>
<th>equ</th>
<th>900</th>
</tr>
</thead>
<tbody>
<tr>
<td>lo_time</td>
<td>equ</td>
<td>2100</td>
</tr>
<tr>
<td></td>
<td>org</td>
<td>$1500</td>
</tr>
</tbody>
</table>
|              | movb   | #$90,TSCR1 | ; enable TCNT with fast timer flag clear
|              | movb   | #$03,TSCR2 | ; disable TCNT interrupt, set prescaler to 8
|              | bset   | TIOS,OC0   | ; enable OC0
|              | movb   | #$03,TCTL2 | ; select pull high as pin action
|              | ldd    | TCNT       | ; start an OC0 operation with 700 us as delay
|              | repeat |           | ;
|              | addr   | #lo_time   | ;
|              | std    | TC0        | ;
|              | low    |           | ; wait until OC0 pin go high
|              | brclr  | TFLG1,C0F,low | ;
|              | movb   | #$02,TCTL2 | ; select pull low as pin action
|              | ldd    | TC0        | ; start an OC operation with 300 us as delay
|              | addd   | #hi_time   | ;
|              | std    | TC0        | ;
|              | high   |           | ; wait until OC0 pin go low
|              | brclr  | TFLG1,C0F,high | ;
|              | movb   | #$03,TCTL2 | ; select pull high as pin action
|              | ldd    | TC0        | ;
|              | bra    | repeat     | ;
Write a function to generate a time delay which is a multiple of 1 ms.

Assume that the E clock frequency is 24 MHz. The number of milliseconds is passed in Y. Also write an instruction sequence to test this function.

**Solution:** One method to create 1 ms delay is as follows:
- Set the prescaler to TCNT to 64
- Perform the number of output-compare operations (given in Y) with each operation creating a 1-ms time delay.
- The number to be added to the copy of TCNT is 375. \(375 \times 64 \div 24000000 = 1\) ms

```assembly
delayby1ms pshd movb #$90,TSCR1 ; enable TCNT & fast flags clear
                  movb #$06,TSCR2 ; configure prescaler to 64
                  bset TIOS,OC0 ; enable OC0
                  ldd TCNT

again0           addd #375 ; start an output-compare operation
                  std TC0 ; with 1 ms time delay

wait_lp0         brclr TFLG1,OC0,wait_lp0
                  ldd TC0
                  dbne y,again0
                  puld rts
```
Frequency measurement

- Use an input-capture and an output-compare functions to measure the frequency of the signal connected to the PT0 pin.

- **Solution:** To measure the frequency, we will
  - Use one of the output-compare function to create a 1-second time base.
  - Keep track of the number of rising (or falling) edges that arrived at the PT0 pin within one second.

```assembly
CR equ $0D
LF equ $0A
org $1000
oc_cnt rmb 1
frequency rmb 2
org $1500
movb #$90,TSCR1 ; enable TCNT and fast timer flags clear
movb #$02,TSCR2 ; set prescale factor to 4
movb #$02,TIOS ; enable OC1 and IC0
movb #100,oc_cnt ; prepare to perform 100 OC1 operation, each
                 ; creates 10 ms delay and total 1 second
movw #0,frequency ; initialize frequency count to 0
movb #$01,TCTL4 ; prepare to capture the rising edges of PT0
movb #C0F,TFLG1 ; clear the C0F flag
bset TIE,IC0 ; enable IC0 interrupt
cli
```
ldd TCNT      ; start an OC1 operation with 10 ms delay
continue addd #60000    ;   
std TC1          ;   
wlp brclr TFLG1,C1F,w_lp ; wait for 10 ms
ldd TC1
dec oc_cnt
bne continue
ldd frequency
pshd
ldd #msg
jsr [printf,PCR]
leas 2,sp
swi
msg db CR,LF,"The frequency is %d",CR,LF,0
TC0_isr ldd TC0          ; clear C0F flag
ldx frequency      ; increment frequency count by 1
inx                  ;   
stx frequency      ;   
rti
org $3E6E           ; set up interrupt vector number
fdb TC0_isr         ; for TC0
end
Pulse Accumulator

- Special counter (‘accumulator’) for counting pulses.
- Pulses may be clock-generated or event-generated.
Pulse Accumulator

Configurations:

- The HCS12 standard timer system has a 16-bit pulse accumulator PACA.
- The HCS12 ECT system has four 8-bit pulse accumulators (PAC3…PAC0).
- Two adjacent 8-bit pulse accumulators can be concatenated into a 16-bit pulse accumulator. There are four possible pulse accumulator configurations:
  - Two 16-bit pulse accumulators PACA and PACB
  - One 16-bit pulse accumulator PACA and two 8-bit pulse accumulators PAC1 and PAC0
  - One 16-bit pulse accumulator PACB and two 8-bit pulse accumulators PAC3 and PAC2
  - Four 8-bit accumulators PAC3…PAC0
  - Four 8-bit pulse accumulators PAC3…PAC0 are sharing the signal pins PT3…PT0.
Load holding register and reset pulse accumulator

PT0 ➔ edge detector ➔ delay counter ➔ EDG0 ➔ 8-bit PAC0 (PACN0) ➔ PA0H holding register ➔ interrupt ➔ Host CPU data bus

PT1 ➔ edge detector ➔ delay counter ➔ EDG1 ➔ 8-bit PAC1 (PACN1) ➔ PA1H holding register

PT2 ➔ edge detector ➔ delay counter ➔ EDG2 ➔ 8-bit PAC2 (PACN2) ➔ PA2H holding register ➔ interrupt ➔ 8-bit PAC3 (PACN3) ➔ PA3H holding register

Figure 8.25 Block diagram of four 8-bit pulse accumulators
Pulse Accumulator

Figure 8.26 16-bit Pulse accumulator block diagram
Pulse Accumulator Operation Modes

- **Event counting mode.** The 16-bit PACA can operate in this mode and count the number of events arrived at the PT7 pin. The 16-bit PACB and all four 8-bit pulse accumulators can operate only in this mode.

- **Gated accumulation mode.** The 16-bit PACA can also operate in this mode. As long as the PT7 signal is active (can be high or low), the PACA counter is clocked by a free-running $E \div 64$ signal.

- The active edge of the PACB and PAC3…PAC0 are identical to those of IC0 and IC3…IC0, respectively. Therefore, one needs to use the TCTL4 register to select the active edges for these pulse accumulators.
Interrupt Sources for Pulse Accumulators

- The 16-bit PACA has two interrupt sources: PT7-edge and PACA counter overflow.
- Only two (PAC3 and PAC1) of the 8-bit pulse accumulators can generate interrupt.
  » These two pulse accumulators can interrupt whenever their counters overflow.
- PACB can interrupt the MCU whenever its upper 8-bit counter overflows.
Enhanced Capture Timer (ECT)

- Has all the features provided in the standard timer module
- Suitable for high-frequency operation (MCU does not need to read the first edge before the second edge arrives).
- Simplifies software for signal measurement
- Avoids capture values to be overwritten before they have been read (using an extra ‘holding’ register)
- True edge detection:
  » The ECT module uses a delay counter to distinguish true edge and false.
  » A transition is determined to be a true edge if the transition is longer than the preset duration.
  » The duration used to determine the true edge is controlled by a register.