ECE 2534

More about Timers

on the PIC32

Basic idea

- A timer is essentially a hardware counter that is updated at a known rate
  - The counter an be configured/accessed through software
- Common uses:
  - Measure (or wait) a predetermined amount of time
  - Measure the time that elapses between 2 external events
  - Generate an output pulse of known duration
  - Generate a square wave with a desired period and duty cycle
  - Generate interrupts periodically
  - Count external pulses
- Why are timers so popular?
  - Many applications of microcontrollers involve time-related actions
  - A timer can be used to measure time intervals much more accurately than a software delay loop

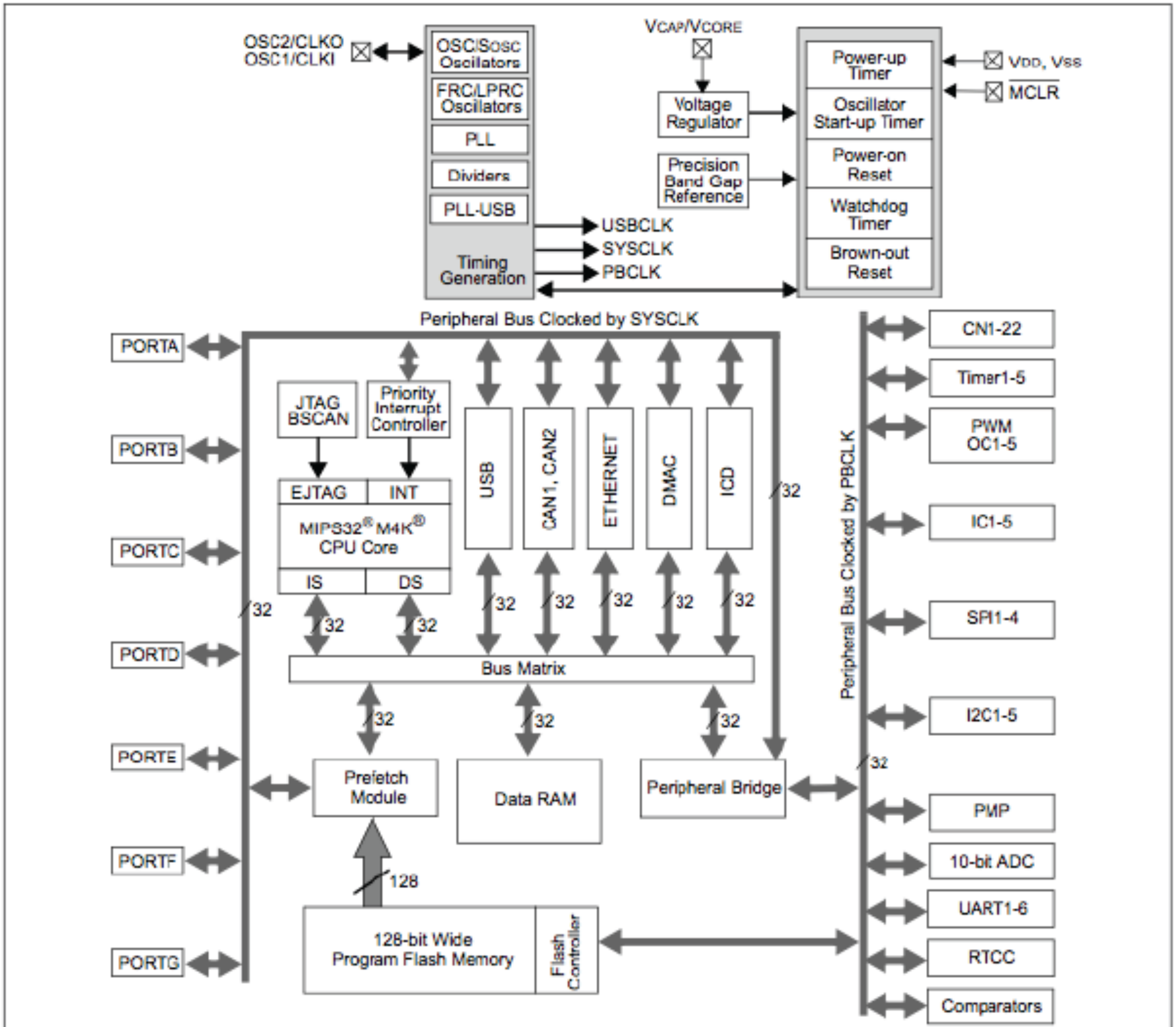
- On the next slide, try to find the Timer peripherals

- On the next slide, try to find SYSCLK and PBCLK

PIC32

(PIC32 datasheet, section 1)

FIGURE 1-1: BLOCK DIAGRAM(1,2)



Note 1: Some features are not available on all device variants.

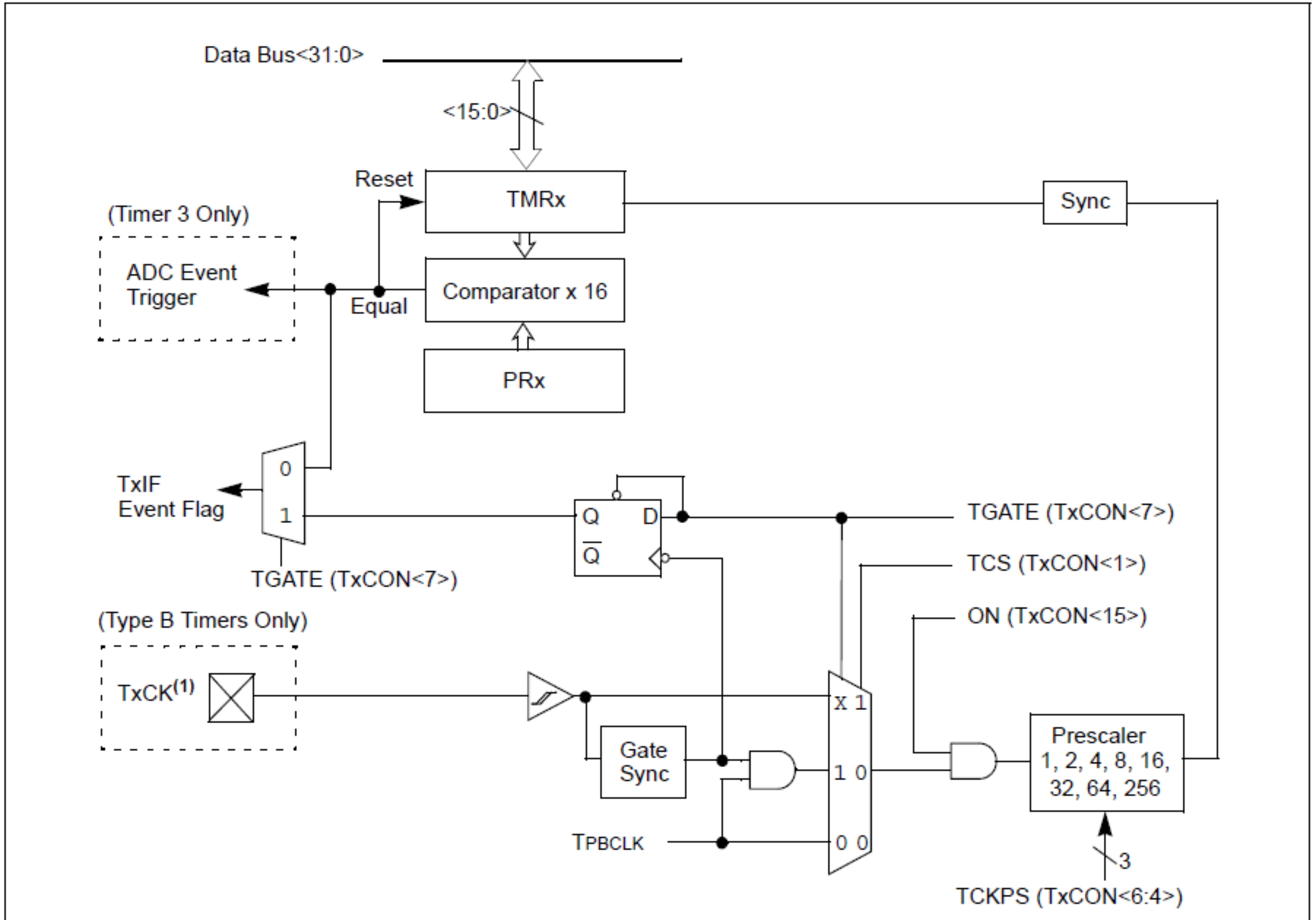
2: BOR functionality is provided when the on-board voltage regulator is enabled.

Our PIC32 has 5 timer modules and a core timer

- Type A: **Timer 1**
  - 16-bit sync/async timer with gate active during CPU sleep (operable from a built-in 32 kHz clock)
  - can be used to implement periodic wake-up, for example
- Type B: **Timer 2 – Timer 5**
  - 16-bit synchronous timer, or
  - 32-bit synchronous timer by combining 2 timer modules
  - external gate is possible
  - Timer 3 can trigger the ADC module
- **Core timer**:
  - counts at rate SYSCLK/2 (increments every 2 cycles of SYSCLK)
  - can generate interrupts when a preset count is reached

Timer 2 / 3 / 4 / 5 (16-bit operation)

Figure 14-2: Type B Timer Block Diagram (16-Bit)



Note 1: The TxCK pin is not available in certain variants of the PIC32MX device family. In such cases, the timer must use the peripheral clock as its input clock. Refer to the specific device data sheet for the I/O pin details.

Focus on Type B, 16-bit mode

- Clock source → prescaler → TMRx register

- TMRx
  - “Timer Count Register”
  - can write/read from software
  - automatically increments
  - automatically compared with contents of the “Period Register”, PRx
- Clock source: PBCLK or external
- Prescaler
  - programmable frequency divider
  - N = 1, 2, 4, 8, 16, 32, 64, ~~128,~~ 256
  - With internal clock source, TMRx updates at frequency f_{PBCLK} / N

Modes of operation

- ❑ Synchronous clock counter mode
  - TMRx increments until it reaches PRx
  - then TMRx is reset, and it sets a flag,
    and optionally an interrupt is generated
  - Example: PRx contains 5
    TMRx = 0, 1, 2, 3, 4, 5, 0, 1, 2, 3, 4, 5, 0, 1, ….
  - note: TMRx continues to increment after reset,
    so an interrupt should be serviced before TMRx reaches PRx again
  - in this mode, the clock signal is provided by PBCLK
- ❑ Synchronous external clock counter mode
  - in this mode, the clock signal is provided on the TxCK pin
- ❑ Gated timer mode
  - TMRx increments (using PBCLK) when an external "gating" signal is high (provided on the TxCK pin)
  - on the falling edge of the gating signal, an interrupt can be generated
- ❑Asynchronous external counter mode (Type A only)

Not every timer mode is supported for a particular timer type:

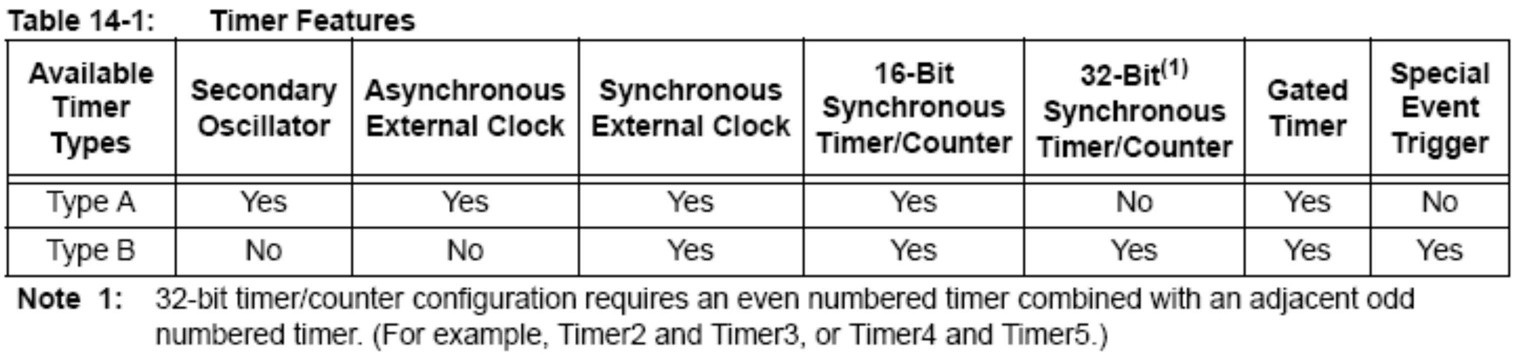


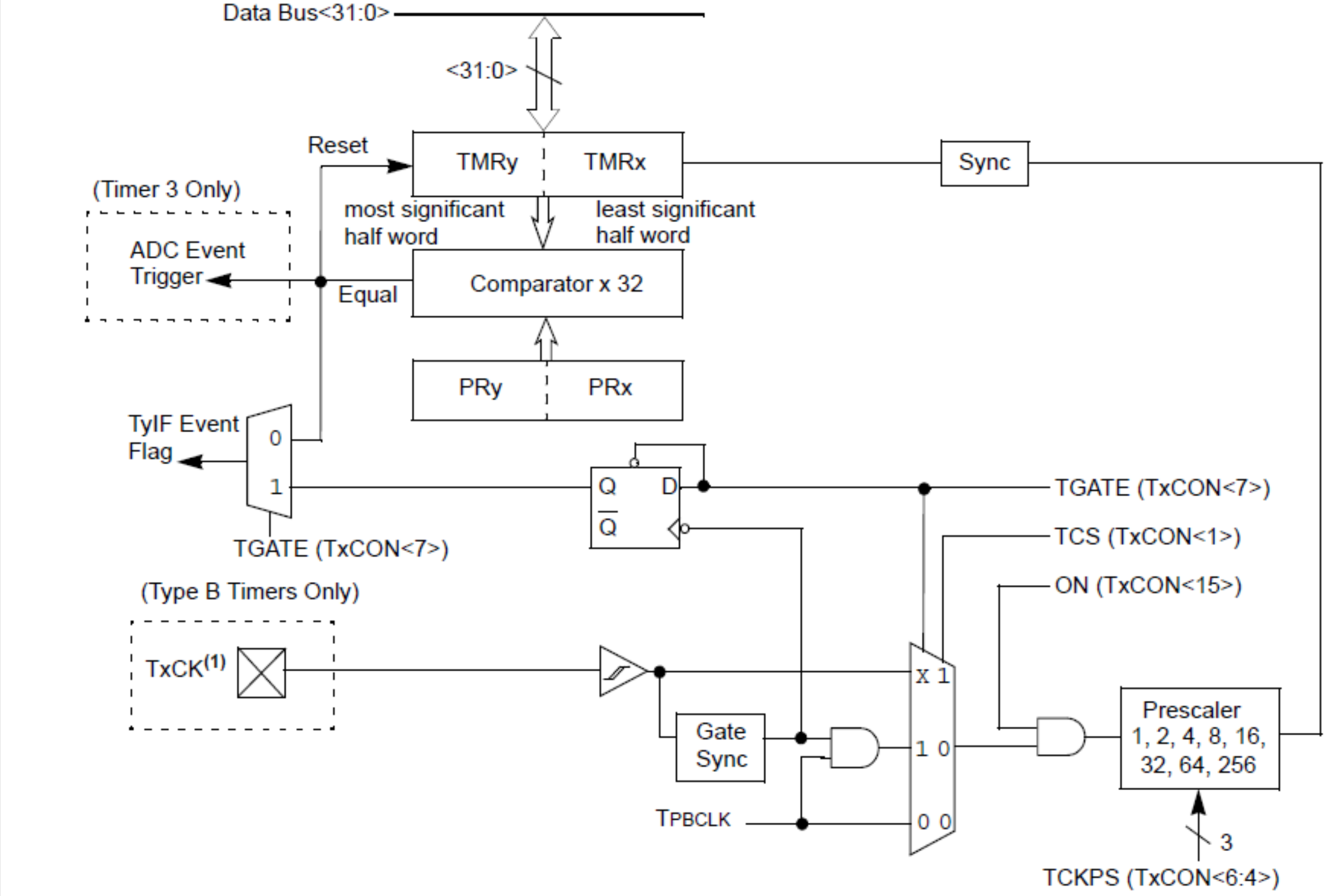
Table 14-1: Timer Features

Available Timer Types	Secondary Oscillator	Asynchronous External Clock	Synchronous External Clock	16-Bit Synchronous Timer/Counter	32-Bit[1] Synchronous Timer/Counter	Gated Timer	Special Event Trigger
Type A	Yes	Yes	Yes	Yes	No	Yes	No
Type B	No	No	Yes	Yes	Yes	Yes	Yes

Note 1: 32-bit timer/counter configuration requires an even numbered timer combined with an adjacent odd numbered timer. (For example, Timer2 and Timer3, or Timer4 and Timer5.)

Timer 2 / 3 / 4 / 5 (32-bit operation)

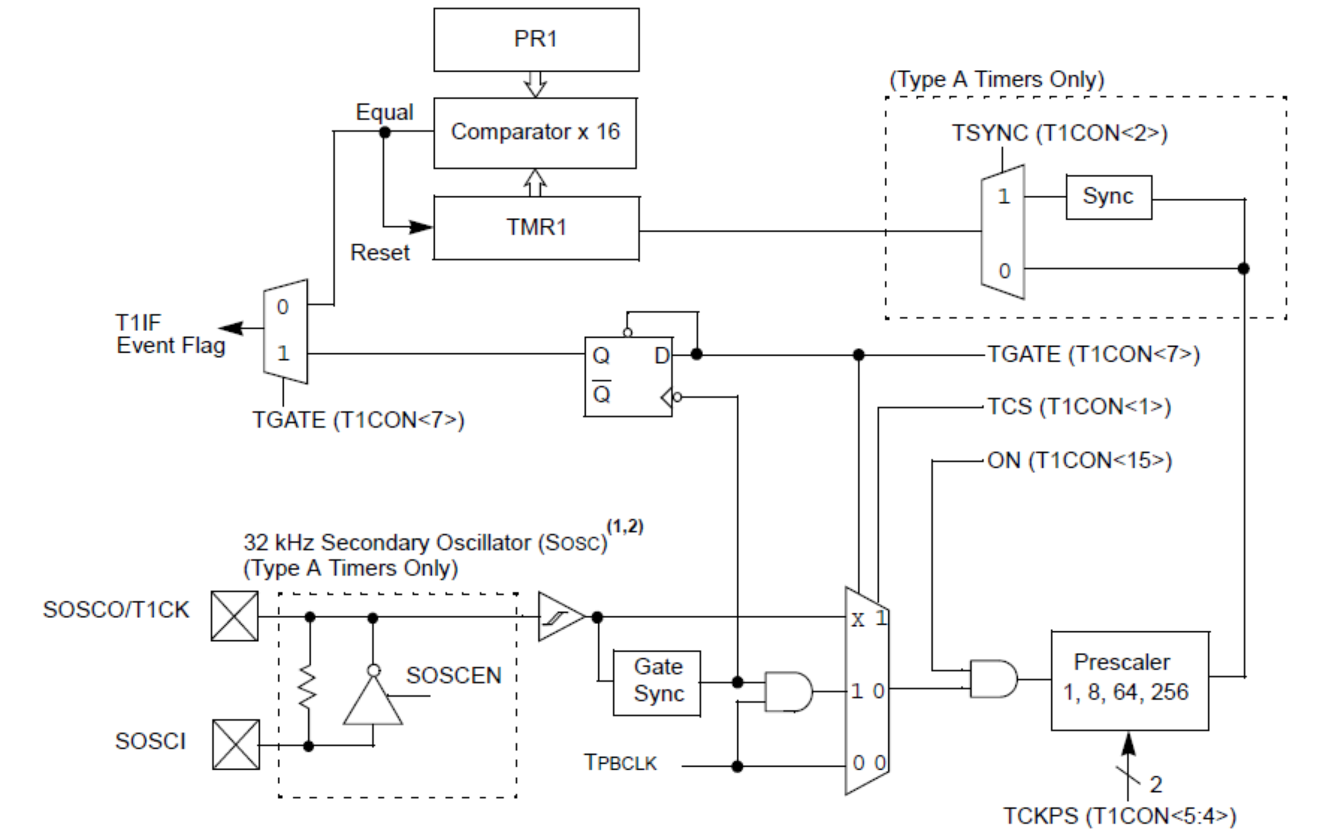
Figure 14-3: Type B Timer Block Diagram (32-Bit)



Note 1: The TxCK pin is not available in certain variants of the PIC32MX device family. In such cases, the timer must use the peripheral clock as its input clock. Refer to the specific device data sheet for the I/O pin details.

Timer 1 (has asynchronous option)

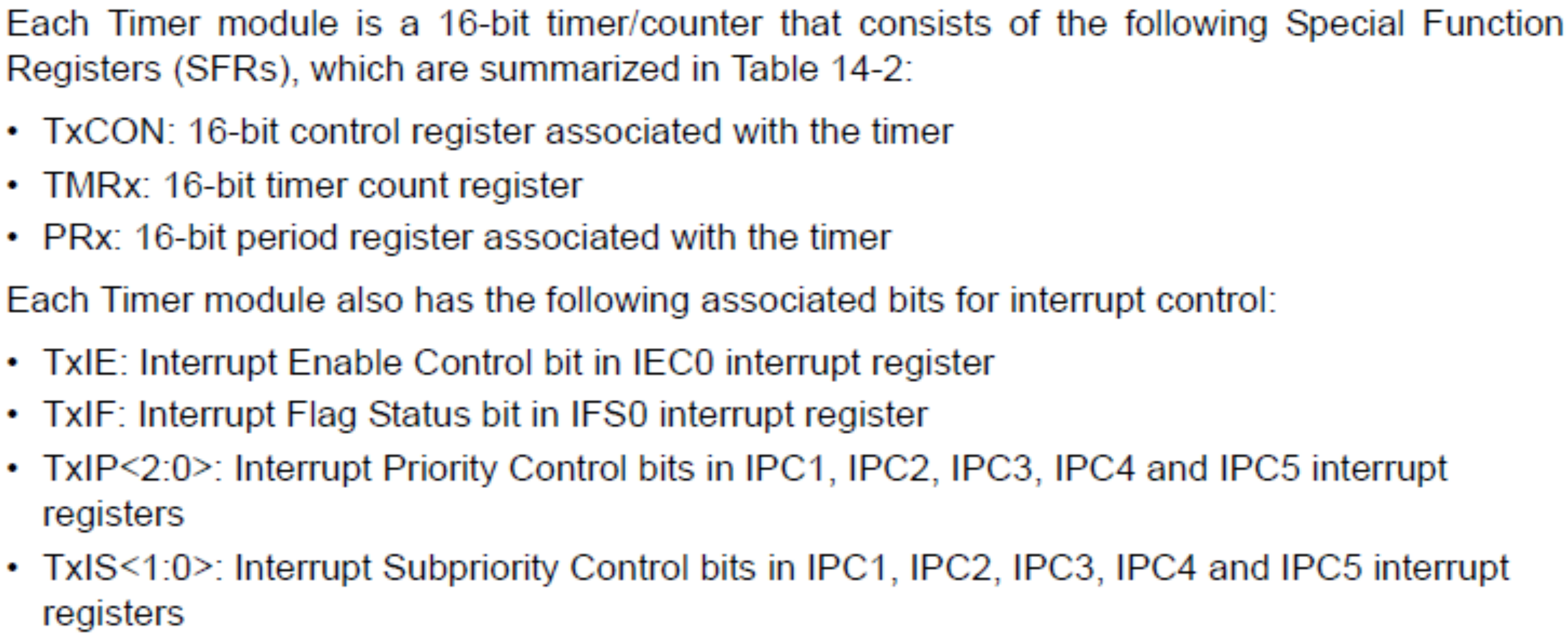
Figure 14-1: Type A Timer Block Diagram



Note 1: For information on enabling the 32 kHz Secondary Oscillator (SOSC), refer to **Section 6. "Oscillators"** (DS61112).

2: The default state of the SOSCEN bit (OSCCON<1>) during a device reset is controlled by the FSOSCEN bit (DEVCFG1<5>).

Control registers

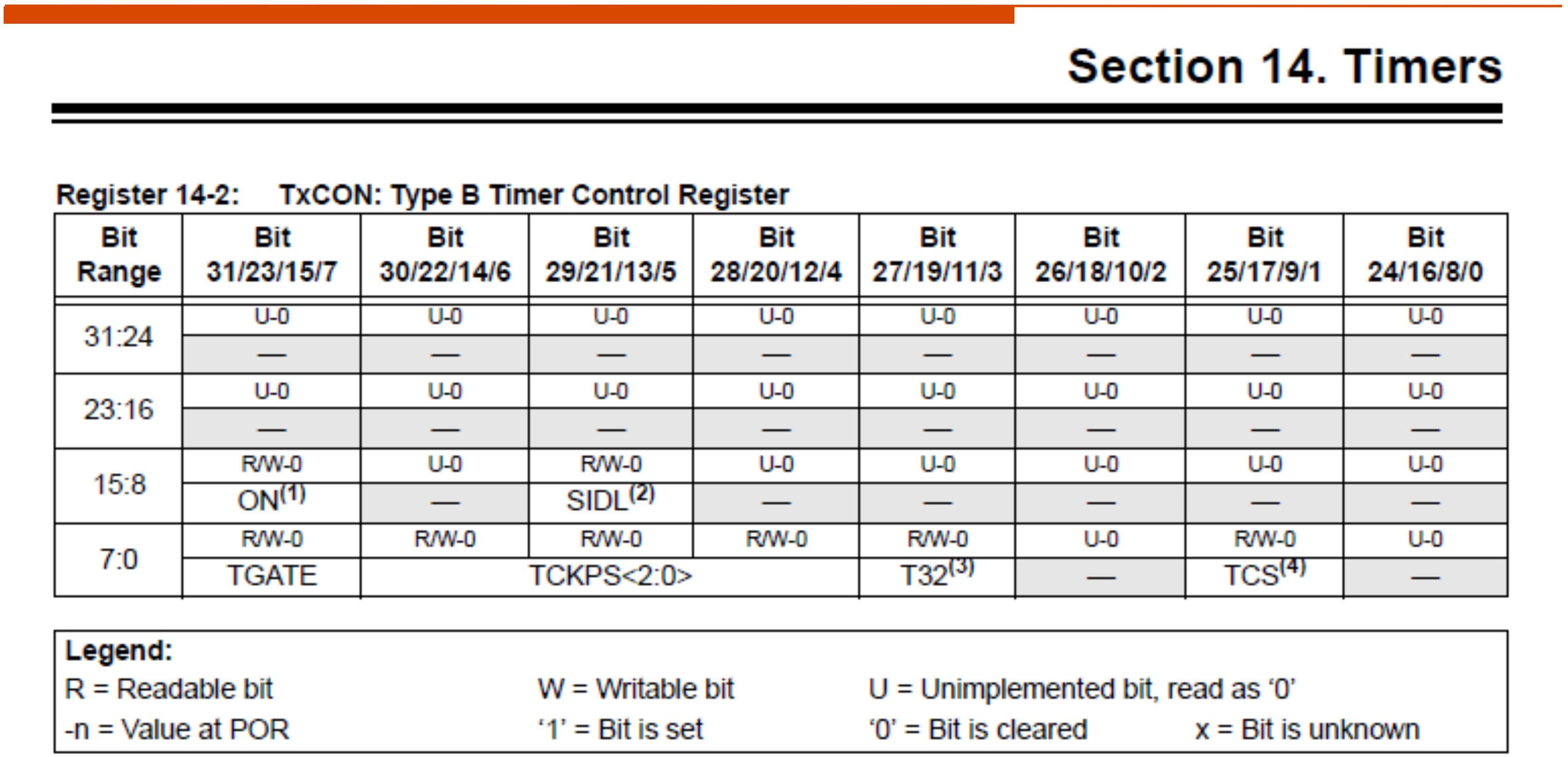


Each Timer module is a 16-bit timer/counter that consists of the following Special Function Registers (SFRs), which are summarized in Table 14-2:

- TxCON: 16-bit control register associated with the timer
- TMRx: 16-bit timer count register
- PRx: 16-bit period register associated with the timer

Each Timer module also has the following associated bits for interrupt control:

- TxIE: Interrupt Enable Control bit in IEC0 interrupt register
- TxIF: Interrupt Flag Status bit in IFS0 interrupt register
- TxIP<2:0>: Interrupt Priority Control bits in IPC1, IPC2, IPC3, IPC4 and IPC5 interrupt registers
- TxIS<1:0>: Interrupt Subpriority Control bits in IPC1, IPC2, IPC3, IPC4 and IPC5 interrupt registers



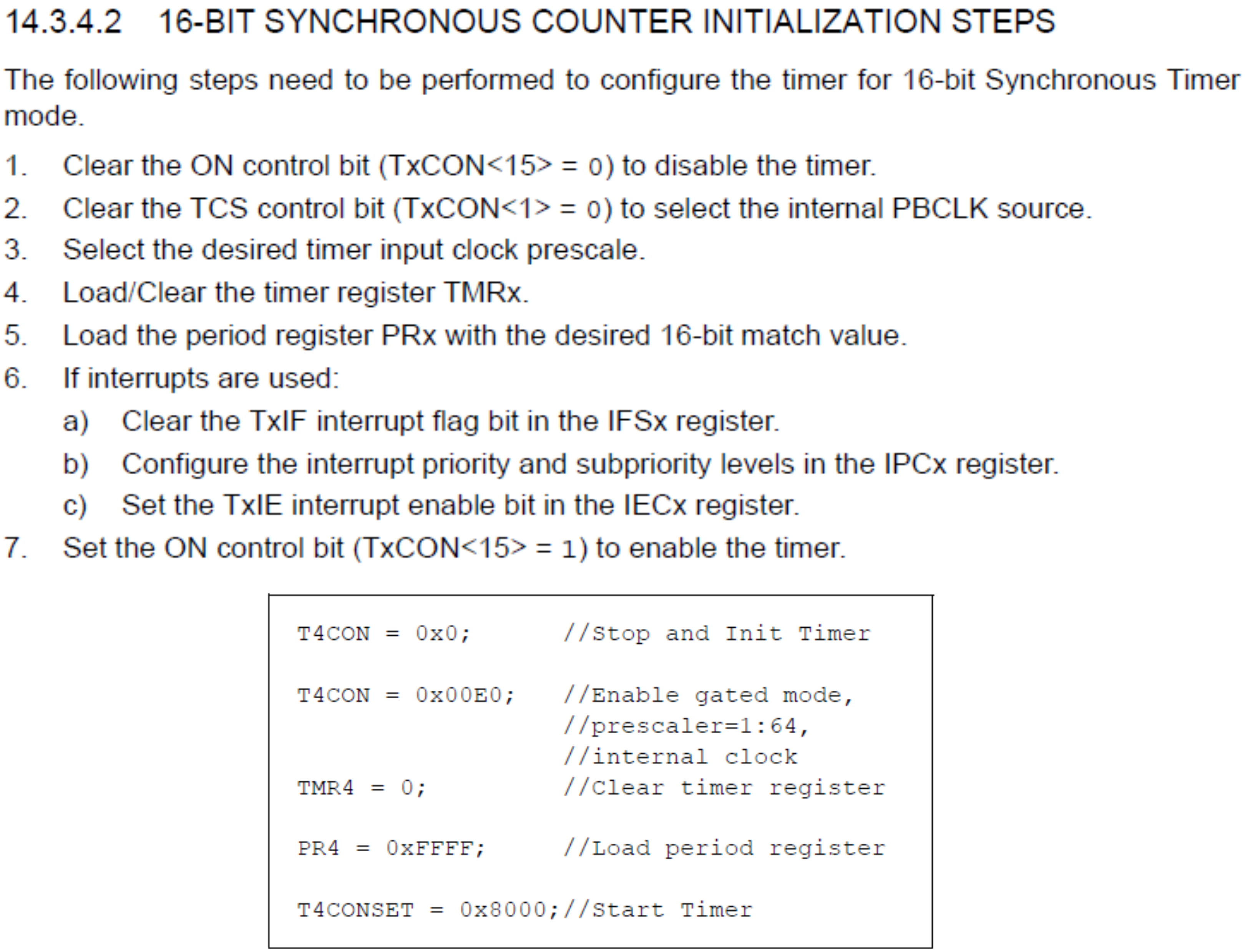
Section 14. Timers

Register 14-2: TxCON: Type B Timer Control Register

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	R/W-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
	ON[1]	—	SIDL[2]	—	—	—	—	—
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	U-0
	TGATE	TCKPS<2:0>			T32[3]	—	TCS[4]	—

Legend:

R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown



14.3.4.2 16-BIT SYNCHRONOUS COUNTER INITIALIZATION STEPS

The following steps need to be performed to configure the timer for 16-bit Synchronous Timer mode.

1. Clear the ON control bit (TxCON<15> = 0) to disable the timer.
2. Clear the TCS control bit (TxCON<1> = 0) to select the internal PBCLK source.
3. Select the desired timer input clock prescale.
4. Load/Clear the timer register TMRx.
5. Load the period register PRx with the desired 16-bit match value.
6. If interrupts are used:
   a) Clear the TxIF interrupt flag bit in the IFSx register.
   b) Configure the interrupt priority and subpriority levels in the IPCx register.
   c) Set the TxIE interrupt enable bit in the IECx register.
7. Set the ON control bit (TxCON<15> = 1) to enable the timer.

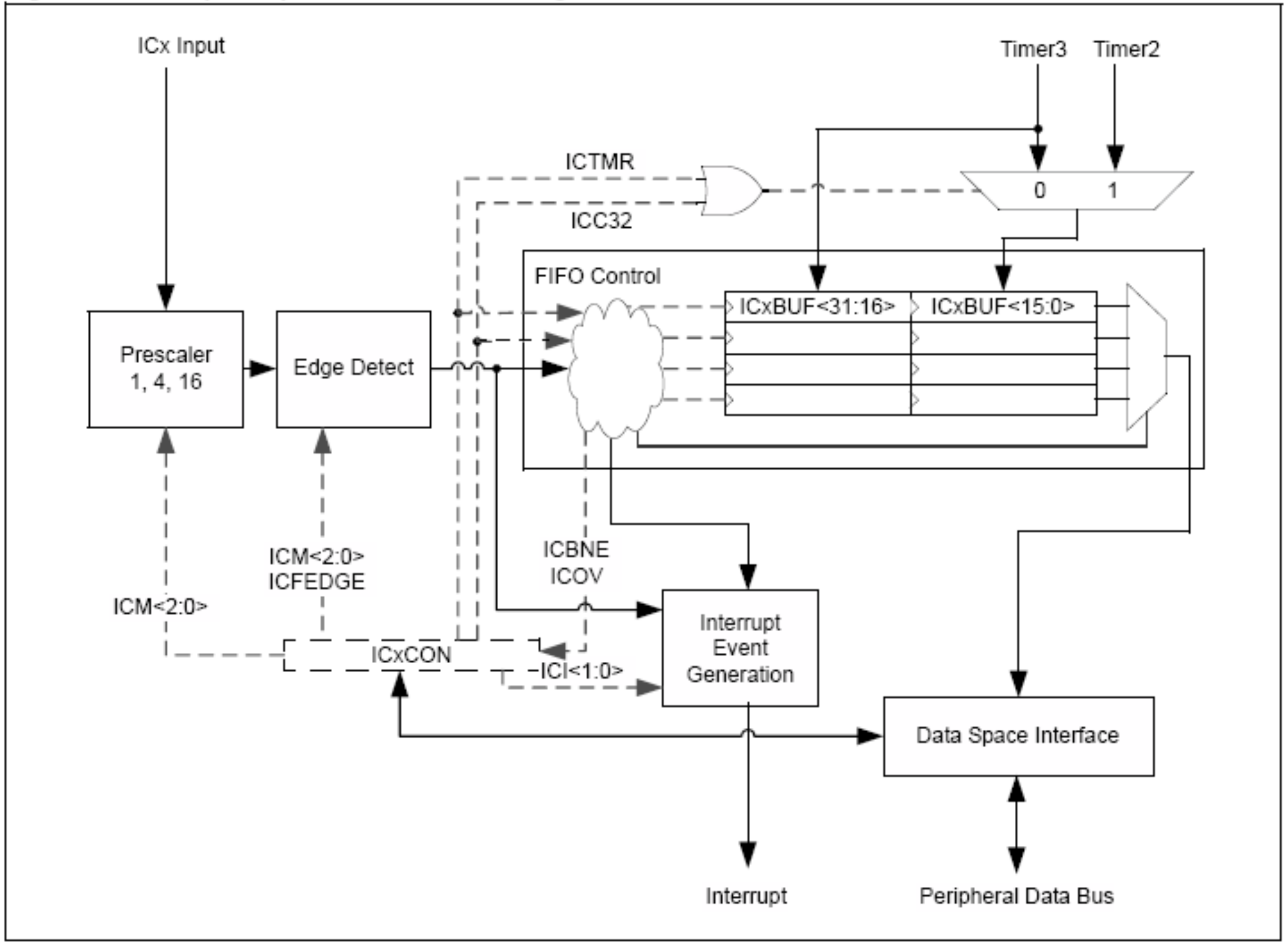
```
T4CON = 0x0;       //Stop and Init Timer

T4CON = 0x00E0;    //Enable gated mode,
                   //prescaler=1:64,
                   //internal clock
TMR4 = 0;          //Clear timer register

PR4 = 0xFFFF;      //Load period register

T4CONSET = 0x8000;//Start Timer
```

Other modules that involve timers

- *Input Capture*: The value in TMRx register is captured when some (configurable) external event occurs on an input pin
- *Output Compare*: An output signal is updated when the timer reaches a certain (user-configured) value
  (Useful for generating an single pulse or a pulse train on an output pin)
- *ADC*: TMR3 and TMR5, acting in 16-bit or 32-bit mode, can trigger an A/D conversion on a period match.

Input Capture

Figure 15-1: Input Capture Module Block Diagram



Input Capture

- **Input Capture Control Register (ICxCON).**
    - Enable Capture
    - Select First Capture Edge (special mode)
    - Set Precision
    - Select Timer (16-bit modes)
    - Interrupt Control
    - Mode Select – includes *edge selection, prescaling*
    - Check *Capture Overflow* and *Status of IC Buffer*

Input Capture Mode

- Upon the occurrence of an event that triggers a capture, read the value from the **Input Capture Buffer Register (ICxBUF)**.
- Reading **ICxBUF** reads from a four-deep FIFO buffer.
- Use the **Input Capture Buffer Not Empty Bit (ICBNE) [ICxCON<3>]** to determine if the FIFO buffer is empty. (In general, if N captures have occurred, it takes N reads to empty the buffer.)

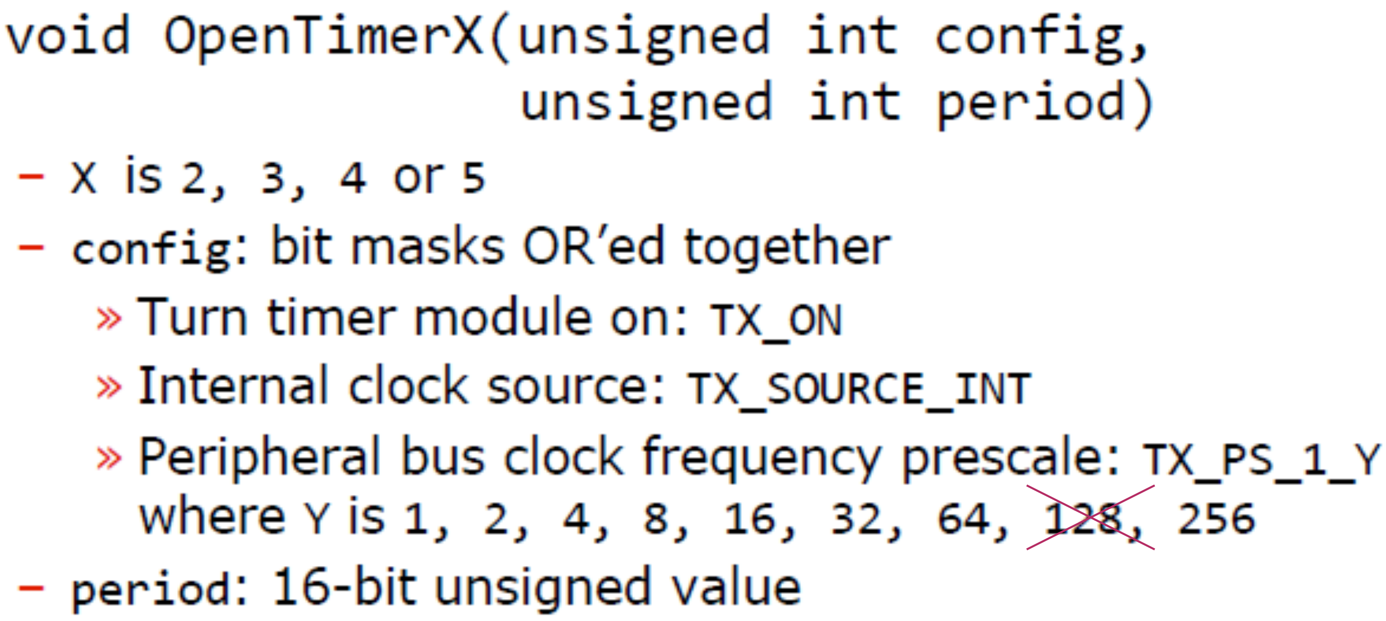
Applications of Input Capture

- Stop-watch functions
- Event recorders
- Frequency-to-Digital converters

Timer PLIB Functions

- OpenTimerX(unsigned int config, unsigned int period)
- ConfigIntTimerX(unsigned int config)
- EnableIntTX()
- DisableIntTX()
- ReadTimerX()
- WriteTimerX(unsigned int value)
- ReadPeriodX()
- WritePeriodX(unsigned int value)
- CloseTimerX()

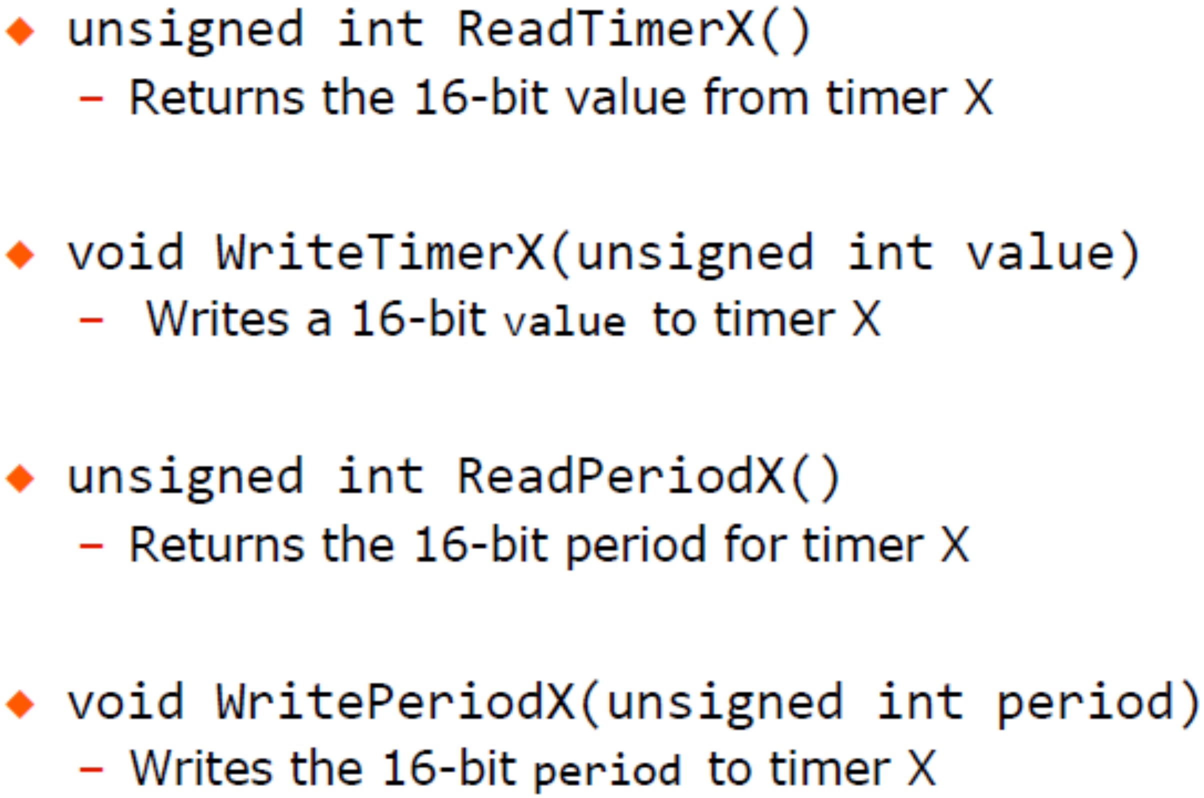
Configuring a timer with the Plib



```
void OpenTimerX(unsigned int config,
                unsigned int period)
```

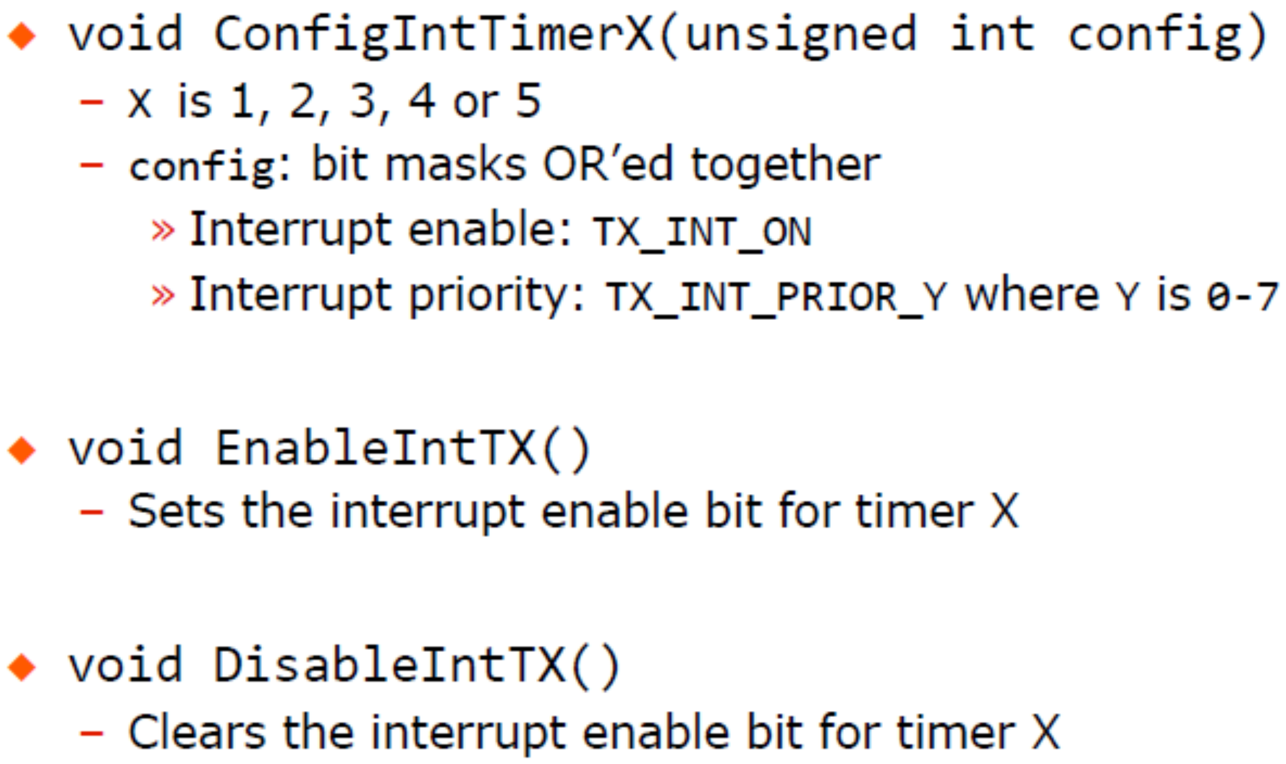
- X is 2, 3, 4 or 5
- config: bit masks OR'ed together
  - » Turn timer module on: TX_ON
  - » Internal clock source: TX_SOURCE_INT
  - » Peripheral bus clock frequency prescale: TX_PS_1_Y where Y is 1, 2, 4, 8, 16, 32, 64, ~~128,~~ 256
- period: 16-bit unsigned value

Reading / writing timer and period registers



- `unsigned int ReadTimerX()`
  - Returns the 16-bit value from timer X
- `void WriteTimerX(unsigned int value)`
  - Writes a 16-bit `value` to timer X
- `unsigned int ReadPeriodX()`
  - Returns the 16-bit period for timer X
- `void WritePeriodX(unsigned int period)`
  - Writes the 16-bit `period` to timer X

Enabling/disabling interrupts



- `void ConfigIntTimerX(unsigned int config)`
  - `X` is 1, 2, 3, 4 or 5
  - `config`: bit masks OR'ed together
    - » Interrupt enable: `TX_INT_ON`
    - » Interrupt priority: `TX_INT_PRIOR_Y` where `Y` is `0-7`

- `void EnableIntTX()`
  - Sets the interrupt enable bit for timer X

- `void DisableIntTX()`
  - Clears the interrupt enable bit for timer X

Summary (1 of 2)

- All modern microcontrollers incorporate timer modules

- Common uses:
  - Measure (or wait) a predetermined amount of time
  - Measure the time that elapses between 2 external events
  - Generate an output pulse of known duration
  - Generate a square wave with a desired period and duty cycle
  - Generate interrupts periodically
  - Count external pulses

Summary (2 of 2)

- Five 16-bit **Timer Registers (TMRx)**
  - *Type A Timer Register*: Timer 1
  - *Type B Timer Register:* Timers 2, 3, 4, and 5
- **Period Registers (PRx)**: User configurable; holds a value that, when reached by TMRx, causes TMRx to reset and (possible) the generation of an interrupt.
- **Timer Control Registers (TxCON)**: Enable a timer, choose a timer mode, choose a *prescale value*.