TIMERS

Definitions

- A counter counts (possibly asynchronous) input pulses from an external signal
- A timer counts pulses of a fixed, known frequency usually the system clock for the processor

Timers/Event Counters

Uses of Programmable Timers and Event Counters

1. **Generate real-time interrupts**
2. **Output precisely timed signals**
3. **Programmable baud rate generator**
4. **Measure time between events**
5. **Count external events**
6. **Generate event caused interrupts**

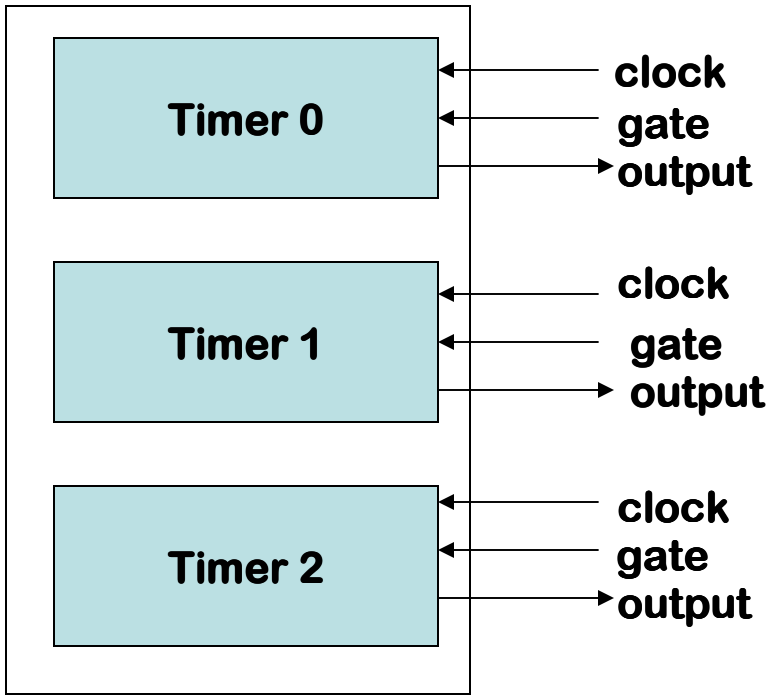
Timer applications

- Frequency measurement
- Waveform generation (e.g. sound)
- time base (multi task system, sampling system, ...)

- Physically, timer is a register whose value is continually increasing to 255, and then it starts all over again: 0, 1, 2, 3, 4...255....0,1, 2, 3.....etc.

Timer (8253 used in PC)

- Three identical timer elements
  - Actually they are counters of the clock
  - Inputs are clock and gate
  - Output
    - Changes or pulses when counting cycle is complete
  - Modes
    - Interrupt on countdown
    - One-shot
    - Pulse rate generator
    - Square-wave generator
    - Software-triggered strobe
    - Hardware-triggered strobe

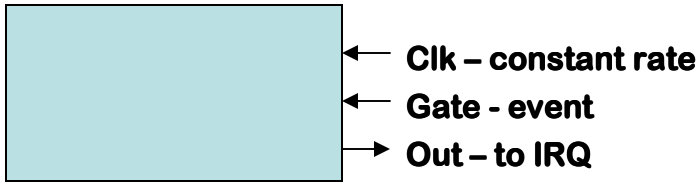


Registers

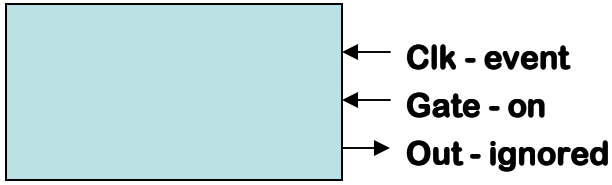
0 - Counter 0
1 - Counter 1
2 - Counter 2
3 - Control for all

Common timer applications

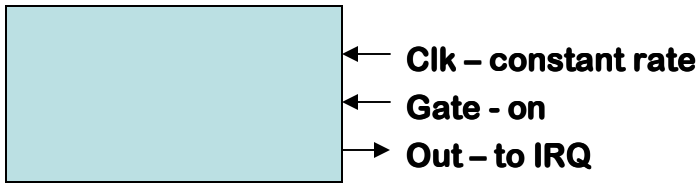
- Combinations of connection and mode
  - Examples
    - Internal speaker
    - periodic interrupt using countdown from system clock and square-wave mode
    - counter input controls frequency
    - 18HZ interrupt
    - 4.77 MHz clock counted down by 12, then by 64K
    - pulse-rate generator mode



Time interval measurement



Event counter



Delay or periodic interrupt

- This incrementing is done in the background of everything a microcontroller does.
- It is up to programmer to think up a way how he will take advantage of this characteristic for his needs.
- One of the ways is increasing some variable on each timer overflow.
  - If we know how much time a timer needs to make one complete round, then multiplying the value of a variable by that time will yield the total amount of elapsed time.

PIC Timers

- Available in all PICs
- may generate interrupts on timer overflow
- Some 8 bits some 16 bits some have prescalers
- Some can connect to external clock, some to the processor clock, some to either.
- Read/write a number as the current count is possible for some.

PIC Timers

- Can use external pin as clock in / clock out (i.e. for counting events)
- Warning: Some Peripherals shares timer resources.

PIC Timers

- The device has three readable and writeable hardware **timers** that can increment automatically each instruction cycle (if no prescaler is used).
- All **timers** can cause an interrupt on overflow, and then restart from zero.

PIC Timers

- Timer 0

  timer/counter with prescale
- Timer 1

  timer/counter with prescale
- Timer 2

  timer only with prescale and postscale
- Watch Dog Timer (discussed later)

Timers

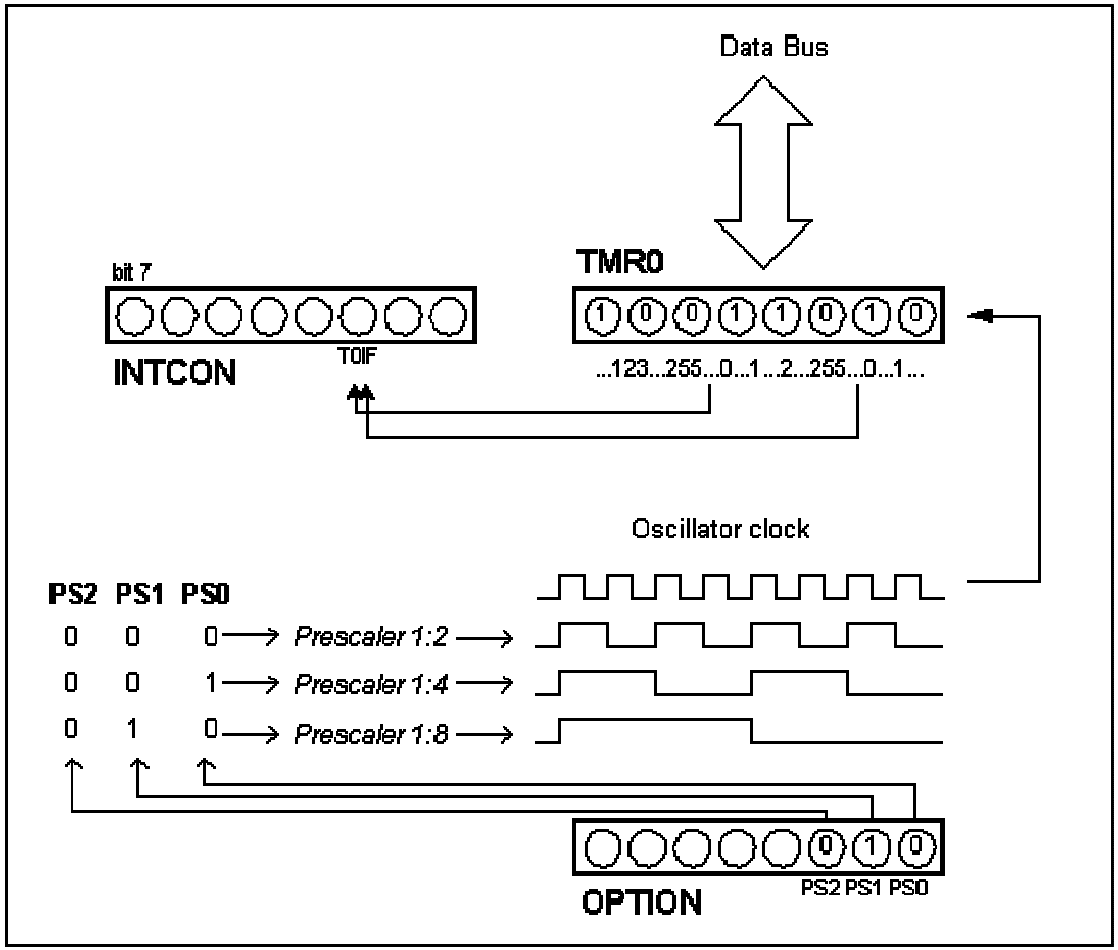
- TIMER0 is an 8-bit timer with an eight bit prescaler, which can make the timer run 2 to 256 times slower than normal
- TIMER1 is a 16-bit timer (two 8-bit registers) with a 1:1 to 1:8 prescaler and some other features. Used by given C code to generate soft timer and sound
- TIMER2 is an 8-bit timer with 1:1 to 1:16 prescaler and a 1:1 to 1:16 postscaler It also has a period register.Used by given C code for PWM motor control

Timer 0

- 8 bit timer/counter with prescaler
- Readable and writeable
- 8-bit software programmable prescaler
- Internal or external clock set
- Interrupt on overflow from 0xFF to 0x00
- Edge Select for external clock

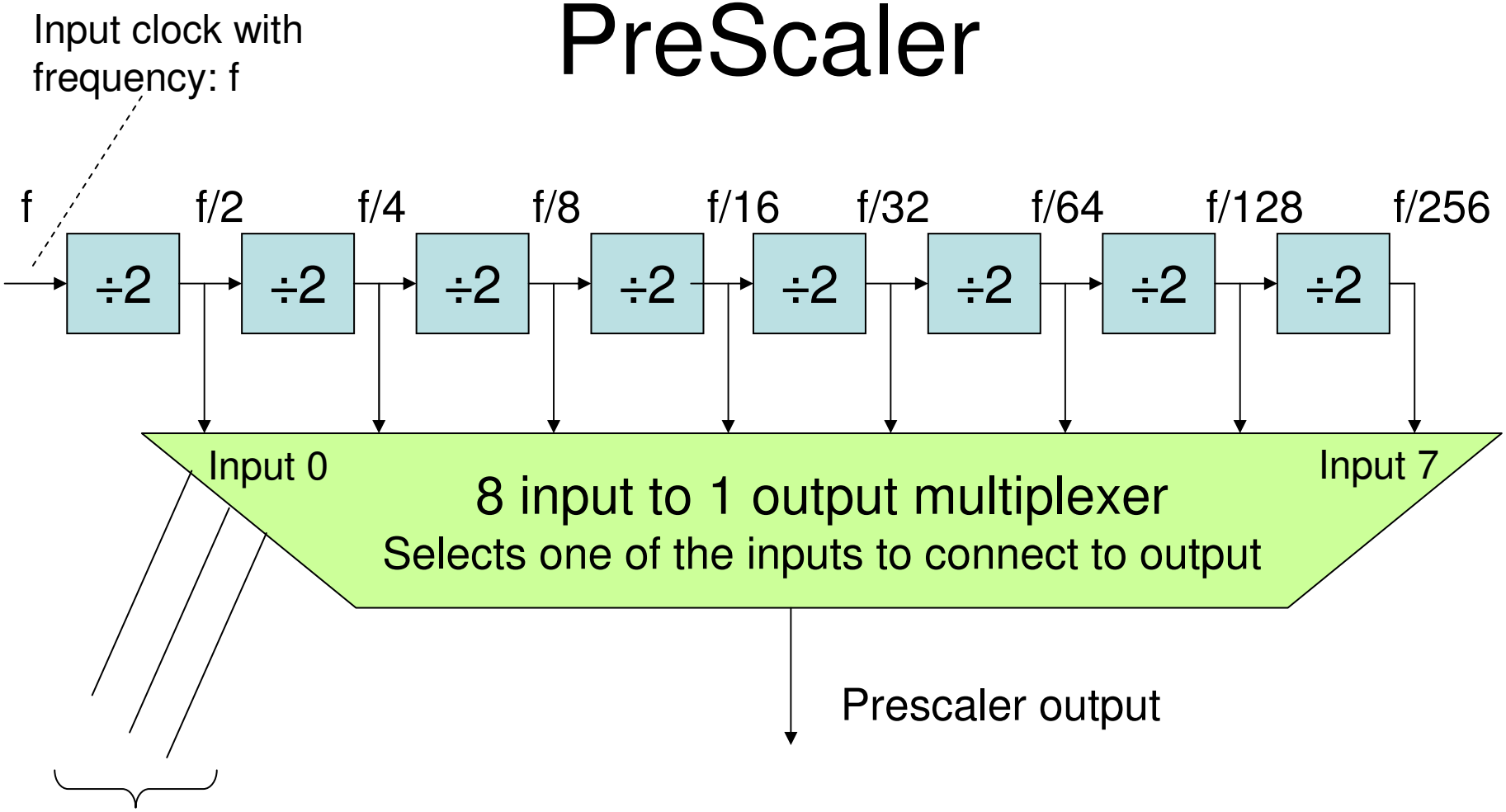
Prescaler

- Prescaler is a name for the part of a microcontroller which divides oscillator clock before it will reach logic that increases timer status.
- Number which divides a clock is defined through first three bits in OPTION register.
- The highest divisor is 256. This actually means that only at every 256th clock, timer value would increase by one.
- This provides us with the ability to measure longer timer periods.



Relation between the timer TMR0 and prescaler

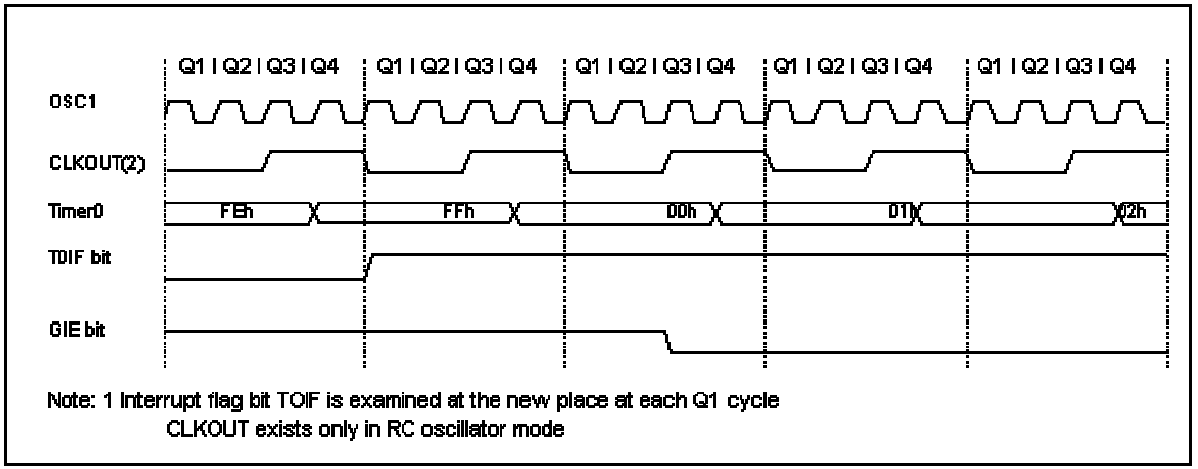
PreScaler



PS2, PS1, PS0 : PreScaler select inputs:
Binary number on these 3 bits determine which input (0..7) to be selected

Note: if prescaler is disabled, input (f) is directly connected to the counter

- After each count up to 255, timer resets its value to zero and starts with a new cycle of counting to 255.
- During each transition from 255 to zero, T0IF bit in INTCON register is set.
  - If interrupts are allowed to occur, this can be taken advantage of in generating interrupts and in processing interrupt routine.
  - It is up to programmer to reset T0IF bit in interrupt routine, so that new interrupt, or new overflow could be detected.

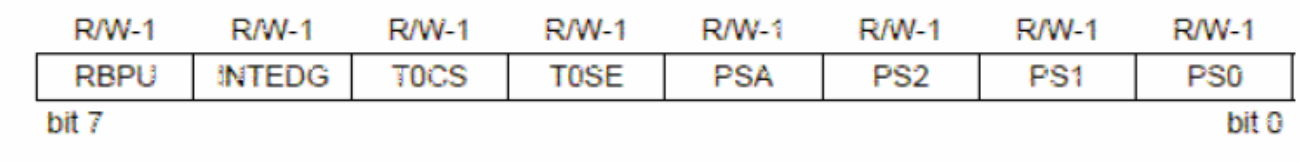


Time diagram of interrupt occurence with TMR0 timer

- The timer is controlled by a number of bits in the Option Register.

Option Register

- The option register is used to control a number of processor features.
- The least significant six bits of the option register control the timer logic.
- The OPTION_REG register is a readable and writable register which contains various control bits to configure
  - the TMR0/WDT prescaler,
  - the External INT Interrupt,
  - TMR0, and
  - the weak pull-ups on PORTB.



The meaning of those bits is as follows:

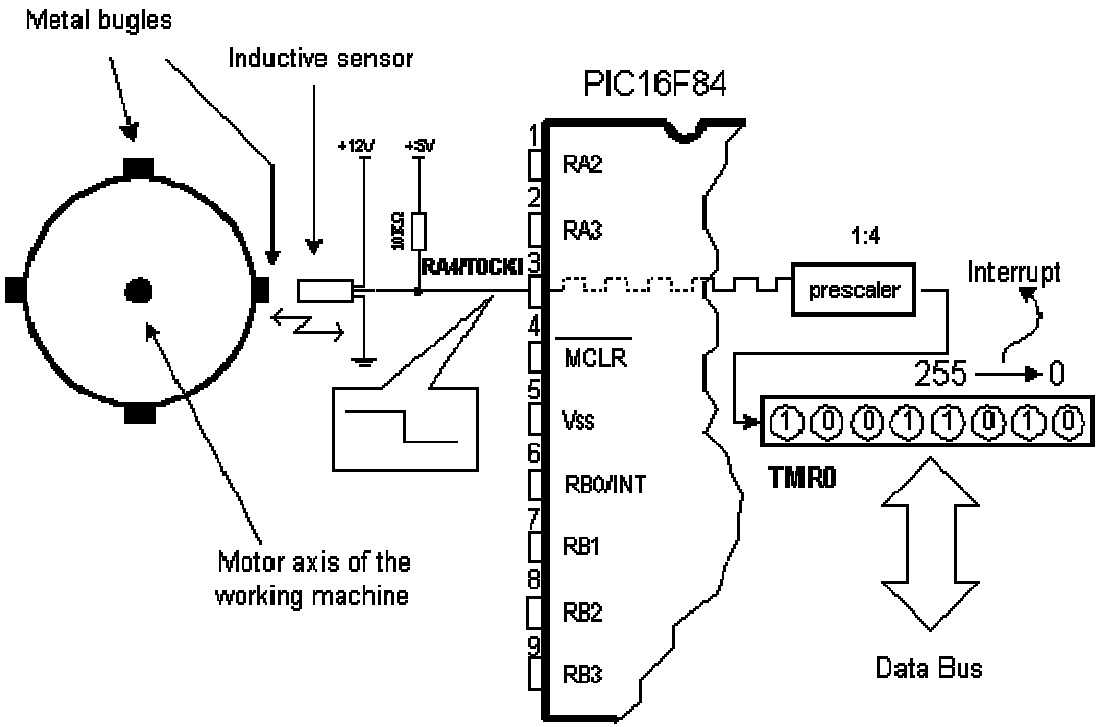
bit	name	purpose
0-2	**PS0-PS2**	These three bits together determine the division ratio of the prescaler.
3	**PSA**	This bit determines whether the prescaler will be used for the TMR0 register or for the watchdog timer
4	T0SE	This bit determines whether the rising or falling edge will trigger a transition when RA4 is used as the input to TMR0
5	**T0CS**	This bit determines whether the processor clock or RA4 will be used as the input to TMR0
6	INTEDG	The PIC can be programmed such that a transition on RB0 causes an interrupt. This bit determines whether that interrupt occurs on the leading or trailing edge
7	RBPU	Clearing this bit enables weak pull-up resistors on all the PORTB inputs. For low current applications like reading switches, this can eliminate the need for external pull-up resistors.

Bits	TMR0	WDT
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

- Beside the internal oscillator clock, timer status can also be increased by the external clock on RA4/T0CKI pin.
  - Choosing one of these two options is done in OPTION register through T0CS bit.
  - If this option of external clock was selected, it would be possible to define the edge of a signal (rising or falling), on which timer would increase its value.

Example

- In practice, one of the typical example that is solved via external clock and a timer is counting full turns of an axis of some production machine, like transformer winder for instance.
- Let's wind four metal screws on the axis of a winder. These four screws will represent metal convexity.
- Let's place now the inductive sensor at a distance of 5mm from the head of a screw.
- Inductive sensor will generate the falling signal every time the head of the screw is parallel with sensor head.
- Each signal will represent one fourth of a full turn, and the sum of all full turns will be found in TMR0 timer.
- Program can easily read this data from the timer through a data bus.



Determining a number of full axis turns of the motor

how to initialize timer to signal falling edges from external clock source with a prescaler

```
      clrf TMR0          ;TMR0=0
      clrf INTCON        ;Interrupts and TOIF=0 disallowed
      bsf STATUS,RP0     ;Bank1 because of OPTION_REG
      movlw B'00110001'  ;prescaler 1:4, falling edge selected external
                         ;clock source and pull up ;selected resistors
                         ;on port B activated
      movwf OPTION_REG ;OPTION_REG <- W
TO_OVFL
      btfss INTCON, TOIF       ;testing overflow bit
      goto TO_OVFL             ;interrupt has not occured yet, wait
;
; (Part of the program which processes data regarding a number of turns)
;
goto TO_OVFL                   ;waiting for new overflow
```

- Prescaler can be assigned either timer TMR0 or a watchdog.
  - Watchdog is a mechanism which microcontroller uses to defend itself against programs getting stuck.
- Prescaler is accorded to timer TMR0, or to watchdog timer trough PSA bit in OPTION register.
  - By clearing PSA bit, prescaler will be accorded to timer TMR0. When prescaler is accorded to timer TMR0, all instructions of writing to TMR0 register (CLRF TMR0, MOVWF TMR0, BSF TMR0,...) will clear prescaler.
  - Prescaler change is completely under programmer's control, and can be changed while program is running.

PIC Timers / Timer 1

- 16-bit timer/counter with prescaler
- Readable and writeable
- 1, 2, 4, 8 programmable prescaler
- Internal or external clock select
- External clock can be syn. or asyn.
- Interrupt on overflow
- Second crystal permitted

PIC Timers / Timer 2

- 8-bit timer/counter with prescaler and postscaler
- Readable and writeable
- 1,4 or 16 programmable prescaler
- 4-bit programmable postscaler
- Interrupt on overflow
- Output to port pin

Selecting Parameters

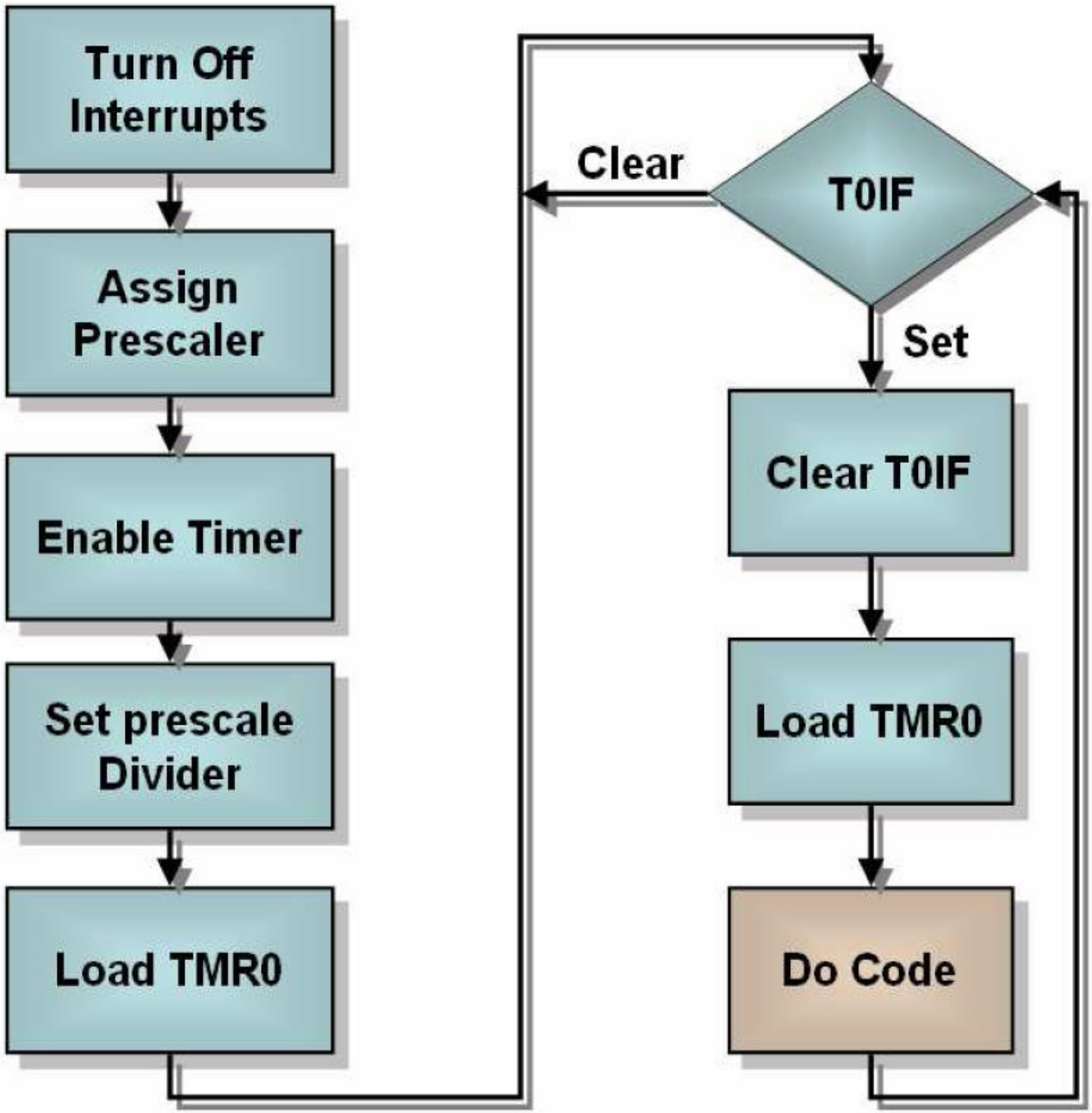
- In order to set up the timer, it is necessary to first decide the time interval needed.
- The basic timer rate is one microsecond (with a 4 MHz crystal).
- This one microsecond clock is divided by the prescaler, which can be set to divide by 2, 4, 8, 16, 32, 64, 128 or 256.
- The timer register itself has 8 bits, so it can count to 256.
- Thus, it is necessary to service the timer with software at least every 256*256 microseconds, or 65.536 microseconds (assuming a 4 MHz clock).

Setting up the Timer

- To set up the timer, one must first disable interrupts so that an interrupt doesn't occur when the timer expires.
- Then, enable the timer and assign the prescaler to the timer.
- Establish the prescaler value, and finally, load the timer register.

Setting up the Timer

- Whenever the timer expires, the T0IF bit in the INTCON register will be set.
- We must clear this bit, reload the timer register, and then execute the code that is to be done at this time.


Turn Off Interrupts
Assign Prescaler
Enable Timer
Set prescale Divider
Load TMR0
Clear
T0IF
Set
Clear T0IF
Load TMR0
Do Code

- In code, the setup portion might look something like:

```
banksel INTCON
bcf INTCON,T0IE ; Mask timer interrupt
banksel OPTION_REG
bcf OPTION_REG,T0CS ; Enable timer
bcf OPTION_REG,PSA ; Prescaler to timer
bcf OPTION_REG,PS2 ; \
bsf OPTION_REG,PS1 ; >- 1:16 prescale
bsf OPTION_REG,PS0 ; /
movlw D'100' ; Timer will count
movwf TMR0 ; 156 (256-100) counts
```

Timer0 Initialization (Internal Clock Source)

```
            CLRF TMR0 ; Clear Timer0 register
            CLRF INTCON ; Disable interrupts and clear T0IF
            BSF STATUS, RP0 ; Bank1
            MOVLW 0xC3 ; PortB pull-ups are disabled,
            MOVWF OPTION_REG ; Interrupt on rising edge of RB0
            ; Timer0 increment from internal clock
            ; with a prescaler of 1:16.
            BCF STATUS, RP0 ; Bank0
;**         BSF INTCON, T0IE ; Enable TMR0 interrupt
;**         BSF INTCON, GIE ; Enable all interrupts
;
; The TMR0 interrupt is disabled, do polling on the overflow bit
;
T0_OVFL_WAIT
            BTFSS INTCON, T0IF
            GOTO T0_OVFL_WAIT
; Timer has overflowed
```

Watchdog Timers

- Watchdog timers are used to guard a system against lock-up due to software errors or soft failures in hardware.
  - Often included in CPU supervisor circuits.
- Retriggering usually done in the main program loop.