

# Using the Timer/Counter in the HT48 Series of MCUs

HA0020E

## Introduction

This present application uses the HT48R10A-1 as an example, and shows how to use the timer/counter within the HT48 series of MCUs. Examples are given on how to use the timer mode, the pulse width measurement mode and the event counter mode along with additional points that should be considered when using the counters.

## Description

The HT48R10A-1 is one of Holtek's high function RISC 8-bit microcontrollers which contains an integrated programmable step-up 8-bit timer/event counter. This counter can be driven by either the microcontroller's internal system clock to implement a standard timing function or controlled by an external timer pin to measure external pulse widths or to count external logical events. Other devices in the series have similar timing functions to the HT48R10A-1.

The HT48R10A-1 has two registers associated with its operation, namely the TMR register and the TMRC register. The TMR register is where the actual timing value is stored. Here an initial value can be written which will be automatically refreshed by hardware when a timer overflow occurs. The TMR register is the control register for the timer in which the various modes of operation are defined.

The HT48R10A-1 can divide the system clock using its internal prescaler to provide the timer with a slower clock source. The division ratio is setup by the lowest 3 bits in the TMRC register. Note that not all timers in the range possess a prescaler function.

Bit name	Bit	Function
PS0~PS2	0-2	Division Ratio: 000: $f_{SYS}/2$ 001: $f_{SYS}/4$ 010: $f_{SYS}/8$ 011: $f_{SYS}/16$ 100: $f_{SYS}/32$ 101: $f_{SYS}/64$ 110: $f_{SYS}/128$ 111: $f_{SYS}/256$
TE	3	When TM1, TM0 = 01, the event counting mode is selected: 1: active on negative transition 0: active on positive transition When TM1, TM0 = 11, the pulse width measurement mode is selected: 1: measures the length of the high pulse (high edge starts timer, low edge stops timer) 0: measures the length of the low pulse (low edge starts timer, high edge stops timer)
TON	4	Timer on/off control bit. (1: ON; 0:OFF)
—	5	Not used, read as "0"
TM0 TM1	6 7	Defines timer mode TM1, TM0 01= Event counting mode 10= Timer mode $\bar{n}$ uses internal clock 11= Pulse width measuring mode 00= No function

**TMRC Timer Control Register**

It is important to note that if the RTC clock is to be used as the timer source clock then the Int RC+RTC must be chosen as the OSC option from the configuration options by selecting "tools/mask option" from the menu bar.

When a counter overflow occurs, the already defined value in the preload register will be automatically reloaded into the timer register. When this happens any new data already written to TMR will be immediately transferred into the timer register. However if the timer is stopped, when the timer is restarted, any data previously already into TMR will be written into the timer register. If data is written into TMR while the timer is running, this data will remain in the preload register until the next overflow occurs at which point it will be transferred into the timer register. When reading the timer/counter register, in order to avoid erroneous readings, the clock is stopped which may result in errors. It is important that programmers take this into account when writing application programs. For all modes, writing a "0" to the ETI will disable the timer interrupts.

## Program Examples

### Example 1

```

;Program Name: Basic internal timer mode (wave.asm)
;Objective: generate a signal on PA.0
;Duty Cycle 1:1 100ms square wave
;configuration options: first select mask option
;/osc/32krtc+bulitinrc, then select the 32k rtc as the timer clock
;source
include ht48r10a-1.inc
code    .section at 0h 'code'

;-----
data    .section 'data'
        pavalue db ?
        temp     db ?
        temp1    db ?
;-----
main    .section 'code'
org     00h
        jmp     start
org     08h
        jmp     intstart
start:
        mov     a,0feh
        mov     pac,a           ;pa.0 setup as an output

main:
        set     intc.0          ;enable master interrupt
        set     intc.2          ;enable timer interrupt
        mov     a,84h           ;use fSYS=32768Hz rtc, fINT=fSYS/32
        mov     tmrc,a
        mov     a,0ceh         ;setup timer intial value, fINT equal to 1ms,
                                ;50x1ms=50ms
        mov     tmr,a          ;start counting
        set     tmrc.4
        jmp     $

intstart:
        mov     temp,a         ;store the acc value
        mov     a,status       ;store the status register value
        mov     temp1,a
        mov     a,01h
        xorm    a,pa           ;invert value of pa.0
        mov     a,temp1
        mov     status,a       ;recover status register
        mov     a,temp         ;recover acc
        reti
end

```

- Program Description

In the timer mode, the timer clock source is the  $f_{INT}$  clock signal. When it is running, the I/O line PA.0 will produce a 1:1 square wave. This example demonstrates how the preloaded value is changed when the previous count finishes.

### Example 2

```
;program name: pulse width measurement mode (wavetest.asm)
;Author: Huang Shan Yun
;Objective: this example demonstrates how to measure the width
;of an input square wave
;Configuration Options: Choose a 4MHz crystal oscillator as the system
;oscillator. The timer prescaler has a minimum value of /2 giving the
;timer  $f_{INT}$  a value of 2MHz
;-----
include ht48r10a-1.inc
data .section 'data'
    count db ?
    over db ?
code .section at 0h 'code'
org 00h
jmp start
    org 08h
jmp timeint
start:
    clr over
    clr acc
    clr count ;initialize program
    clr intc
    mov a,05h
    mov intc,a ;Enable master interrupt and timer interrupt
                ;bits
    mov a,11000000B
    mov tmrc,a ;set pulse width measurement mode, start
                ;counting on falling edge
    mov a,00h
    mov tmr,a ;setup initial value of counter
    set tmrc.4

loop:
    snz tmrc.4 ;check timer enable bit to see if pulse
                ;width measurement has finished
    jmp count1 ;measurement finished (timer enable bit reset
                ;to zero) jump to count1
    jmp loop ;measurement not finished (timer enable bit
                ;equal to "1") jump back and wait for
                ;measurement to finish
```

```

count1:
    mov  a, tmr
    jmp  $
timeint:
    inc  count           ;if the pulse width exceeds the value of 0FFH
                        ;the counter will overflow, and the value of
                        ;this register will increase by 1
    sz   count           ;count overflow situation
    jmp  $1              ;add extra bit to expand measure range
    set  over.0
$1:
    reti
    end

```

- Program Description

The HT48R10A-1 microcontroller has a timer input pin with the name TMR, which is pin-shared with the I/O pin PC1. It is only in this mode that the TON pin will be automatically reset to zero, in the other modes, the TON can only be reset to zero under program control. In the pulse width measuring mode, when both the TON and TE bits are set to "1", when a rising edge appears on the TMR pin the timer/counter will start to run and will continue running until a falling edge appears on the TMR pin. When this happens the TON bit will be automatically cleared to zero. Note that if the TE bit is "0" the counter will start to run when a falling edge appears on the TMR pin and stop when a rising edge appears, at which point the TON bit will also be cleared to zero. When the timer/counter stops counting, the measured value will be stored in the timer register. In other words, when in the pulse width measuring mode the timer/event counter can only measure one pulse width. Only when the TON bit is set again to a high level can the timer begin to measure the width of additional pulses on the TMR pin. When in the pulse width measurement mode, the timer/counter is not activated by the level of the input signal but is rather triggered by the input edge. Once a timer/event counter overflow occurs, the previously setup preloaded value will be again reloaded into the timer register. At the same time a timer interrupt request signal will be generated, a situation which is similar to that of the event counter mode and the timer mode. When in this mode the accuracy of the measured pulse width is related to the accuracy of the timer source clock. Before a measurement takes place, by estimating the general frequency range that is required, more efficient measurements can be made. By choosing the correct frequency and memory it should be possible to meet the demands for accuracy of most measurements. In this example a 1K square wave is measured. The program uses Data Memory to expand the range of the timer and increase its accuracy.

- Measurement results:

This program can measure waveforms within a range of 16Hz~600kHz

### Example 3

```

;Program Name: External event count mode (COUNT.ASM)
;Author: Huang Shan Yun
;Objective: Within a fixed time of 100ms, measure the number
;of external input transitions
;Configuration Options: 4MHz system clock, 1us instruction period
#include ht48R10A-1.inc
;-----
data    .section 'data'
        R1    db ?
        R2    db ?
        count db ?
        over  db ?
code    .section at 0h 'code'
org     00h
        jmp   start
org     08h
        jmp   timeint
start:
        clr   over
        clr   acc
        clr   count    ;initialisation
        clr   intc
        mov   a,05h
        mov   intc,a    ;Enable master interrupt and timer
                        ;interrupt bits
        mov   a,01000000B
        mov   tmrc,a    ;Setup event count mode, measure rising
                        ;edge
        mov   a,00h
        mov   tmr,a    ;setup initial value of counter
        set   tmrc.4
MS:
        ;use instructions to get 100ms time
        ;timing error is 2µs
        mov   a,0c8h    ;1µ
        mov   r1, a    ;1µ
LP2:    mov   a,0a5h    ;1µ
        mov   r2,a    ;1µ
LP1:    sdz   r2    ;1µ
        jmp  LP1    ;2µ
        sdz   R1    ;1µ
        jmp  LP2    ;2µ
        clr   TMRC.4    ;stop timer
        mov   A,TMR    ;get timer value
        jmp  $

```

```
timeint:
inc  count           ;timer overflow counts, increase by 1
sz   count           ;check for count overflow
jmp  $1              ;overflow bit expands range
set  over.0
$1:
reti
end
```

- Program Description

It should be noted that the TON bit in this example can only be cleared to zero under program control. It is only in the pulse width measurement mode that the TON bit will be automatically cleared to zero, which is done after a single pulse measurement has been made. However among the three timer modes of operation, the TON bit can always be cleared under program control. After the timer/counter overflows, the previously preloaded initial value will be again reloaded into the timer register.

- Calculation Method

$$2 + [2 + 3 \times R2(165) + 3] \times R1(200) = 100002\mu s$$

- Timer/Counter result

1kHz input waveform gives timer value TMR of 64H  
100Hz input waveform gives timer value TMR of 0AH  
150Hz input waveform gives timer value TMR of 0FH