Programming the 8254 PIT: A Tech-Tip Example and Boilerplate

Thomas W. Jenkins
Associate Professor, New Mexico State University,
Department of Engineering Technology
PO Box 3001, MSC 3566, Las Cruces, NM 88003-8001, tjenkins@nmsu.edu

Abstract - The Intel® 8254 Programmable Interval Timer (PIT) is a common medium scale integrated electronic device whose many and varied applications include integrating it as a featured component into simple microcomputer based systems. This device is also frequently used in Engineering and Engineering Technology classes when discussing computer hardware, microcomputer system architectures, or microprocessor interfacing. Often this device is utilized in the application portion of these classes or in student projects. The author presents an example of the programming of this device in MASM – an assembly language. This can be used as a boilerplate for describing the device's actions or in the use of this device in electronic or computer hardware laboratories or other applications.

Index Terms – *Intel's Programmable Interval Timer 8254, PIT, MASM Programming Example*

Programming the 8254 PIT: A Tech-Tip Example and Boilerplate

The 8253/4 PIT

Introduction:

Designed to handle timing applications, one-shot applications, clock or frequency generation, and frequency division the Intel® 8254 Programmable Interval Timer (PIT) provides three programmable 16-bit internal counters/dividers and an 8-bit mode register in a single Integrated Circuit (IC) chip. Each of the three counters are fully user programmable to any of the following six different modes of operation via the mode register[1]:

0. Interrupt on terminal count condition
1. Hardware retriggerable one-shot
2. Rate generator
3. **Square wave mode (frequency divider) – the example mode for this paper**
4. Software Triggered Strobe
5. Hardware Triggered Strobe (retriggerable)

This versatile device is most commonly integrated into a microcomputer based system to generate frequencies for use by the CPU and its related component devices or peripherals such as the hard-drive or the internal speaker. Used in this type of system, the programming of the PIC is usually done by an operating system element as part of a microcomputer systems boot-strap or initialization routine. However, given the versatility of this device, it is often useful to modify the standard PIT initialization or it may be necessary to initialize the PIT to a specific specialized task for use in a stand-alone application or in a small user-defined applications separate from microcomputer systems where a commercial operating systems is not available or even unnecessary.

Computer system architecture, microprocessor interfacing, or computer hardware/software classes in Engineering and Engineering Technology programs often have students design (and build) simple microprocessor based systems to demonstrate basic microcomputer architecture concepts or concepts relating to interfacing of the microprocessor to various devices such as memory, IO devices, sensors, or other similar applications. Programming of the internal devices is regularly part of the process. The PIT is regularly a main component of these systems and thus requires the instructor and students to understand the initialization of this device to function within this system. Intel technical manuals are seldom easy for students to understand at their level of experience and many textbooks give only superficial treatment of this device. Therefore, the use of

[1] See the 8254 PIT specifications for a complete discussion of these modes of operation

an example boilerplate and accompanying in-line documentation is offered as a supplement to these resources and as an aid in these activities.

The following is figures from the Intel 8254 specification document [1] which shows the logical parts of the device and the chip template. Users should refer to this document when implementing this device via hardware.

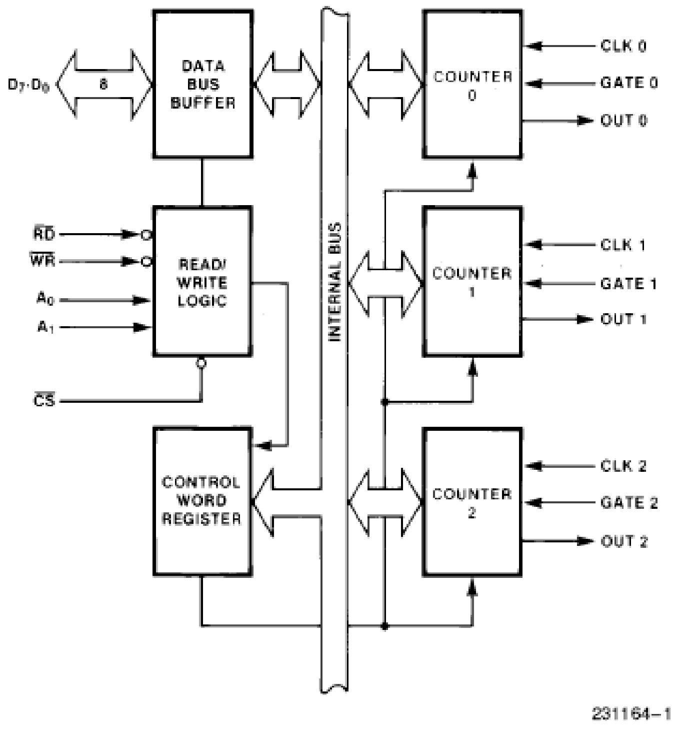

Figure 1. 8254 Block Diagram

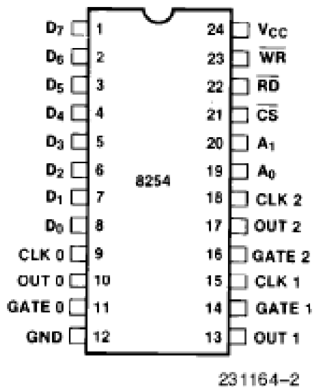

Figure 2. Pin Configuration

The author has used this boiler plate in a senior Hardware Design and Microcomputer Architecture class that has students design, build, integrate, test, and document a single board microprocessor based computer system. Small assembly language routines are constructed to permit the hardware to do specified tasks such as display a real-time clock and function as a four-function calculator. The boiler-plate provided below is given to the students as a supplement to technical manuals, lectures, and text material over these devices. The boiler-plate approach gives good documentation on the function of the device and relieves the student of much of the burden of the coding, which permits more investment in learning interfacing and functionality.

The following provides an example of assembly language code which initializes the PIT into an example configuration and ***provides a boiler-plate for users to modify the code to meet their specific requirements.*** The following is written as MASM code and the author assumes a degree of assembly language familiarity by the readers.

Finally, the author assumes that the device is connected correctly in the system and all hardware connections (including address decoding, gate enables, etc.) are valid as per the Intel specification document [1].

Programming the 8253/4 PIT

```
;*********************************************************************
;*********************************************************************
;                  8253/4 Programmable Interval Timer (PIT)
;*********************************************************************
;*********************************************************************
;In this example, we plan to program the PIT’s internal counters 0 and 1
;(and ignore counter 2 in this exmple)to have both act as repeating
;frequency dividers i.e. mode 3. The hardware input to the PIT counters
;will be a frequency of a specific value and the hardware output from the
;PIT will be smaller frequencies, that is, the divided down frequencies.
;
;In counter 0, we will divide an input signal by the largest possible value
;in 16 bits (64,535+1 – more on this latter) with counter 0, then divide that
;signal by 36 with counter 1.
;
;Therefore, we want the two counter operations to both be configured into
;a mode 3 – repeating square wave frequency divider mode of operation
;
;The process of programming the PIT counters is a two step
;process for each counter – 1) program the mode register which provides
;information as to the mode and counter, then 2)program the counter with
;a numeric value – the count or division value.
;So, with this example in mind:
;
;             Initialize the 8254 PIT controller
;into a continuous counter or divider (mode 3 of the six possible modes),
;which will divide a hardware input signal input to counter 0 by 65536,
;i.e. it acts as a frequency divider.
;            For example:
;If an input is 2.38MHz – a common P_clk signal used in microcomputer
;systems, then this first division produces an output of about 36.4Hz.
; 2.8MHz / 65536 = 36.4Hz approximately
;Note: the P_clk signal is a common clock signal in microcomputers often generated
;by a clock generator circuit and has a well defined freq. and shape.
;
;We can then take this 36Hz output signal and input this to counter 1 by
;tying the output of channel/counter 0 to the input pin of channel/counter 1
;of the PIT. With counter 1 programmed to be a divide by 36 configuration,
;we would obtain a final frequency of 1Hz – one pulse per second – perhaps
;useful for timing of a process or real-time clock frequency applications.
;
;The control words defined below as – PIT_Cx_M3, are sent to the PIT’s mode
;register (commonly located at I/O port 43H) of the 8254.
;The byte value describes how counter register x (0/1 for counter 0 or 1
;respectively) will be utilized i.e. the mode, and how
;the counter will perform. Each counter that we will use, must be
;initialized via the mode register, so for counter 0 we should have the
;following initialization bit pattern or byte length code value:
```

B7	B6	B5	B4	B3	B2	B1	Bit - 0
SC1	**SC0**	**RL1**	**RL0**	**M2**	**M1**	**M0**	**BCD vs Binary**
0 - 0 = Counter 0 0 - 1 = Counter 0 1 - 0 = Counter 0 1 - 1 = Control Register		0 - 0 = Counter Latching operations 0 - 1 = Read/Load LSB only 1 - 0 = Read/Load MSB only 1 - 1 = = Read/Load LSB first then MSB		0 - 0 - 0 = Mode 0 0 - 0 - 1 = Mode 1 0 - 1 - 0 = Mode 2 0 - 1 - 1 = Mode 3 1 - 0 - 0 = Mode 4 1 - 0 - 1 = Mode 5			0 = Binary Counter or 1 = BCD

```
;
; BITS are listed MSB(7) -> LSB(0):  PIT_C0_M3 = PIT_Channel_0_Mode_3
;
;0   0          1  1           0  1    1             0   =   36H
;{Select        RD/WR         { Mode 3 Operation}     Binary
;Counter 0}     LS byte                               Counter
;               then MS                                Mode

;We are selecting counter 0 (00), we will be sending to counter 0 a two
;byte(16 bit) divide number – the value of the frequency division. The
;least significant (LS) byte sent first, then followed by the most
;significant(MS) byte - this is indicated by the (11) bits. Then (011)
;indicates that we want counter 0 to operate in mode 3 and the last (0)
;indicates that the value we are sending is a binary (vs. BCD) number –
;our two numeric options for the counter values.
;
;NOTE: the use of equates (equ) is a common and useful form of assigning
;mnemonic names to constants and thus increasing the MASM program’s
;useful documentation.
;
PIT_C0_M3     equ   36h   ;8253 PIT timer channel 0 mode 3
                          ;(see above bit map)
PIT_C1_M3     equ   77h   ;8253 PIT timer channel 1 mode 3
                          ;see below bit map
;
; BITS are listed MSB(7) -> LSB(0) : PIT_C1_M3: PIT_Channel_1_Mode_3
;
;0   1         1    1          0  1  1                1   =   77H
;{Select        RD/WR         { Mode 3 Operation}     Binary
;Counter 1}     LS byte                               Counter
;               then MS                                Mode
;
;Here, we are selecting counter 1(01), sending to counter 1 two bytes -
;least significant (LS) byte first, followed by the most significant(MS)
;byte - this is indicated by the (11) bits. Then (011) indicates that we
;want counter 0 to operate in mode 3 and the last (1) indicates that
;the value we are sending is a BCD (vs binary) number.

;The PIT is commonly located in most Intel microcomputer systems at the
;standard IO port addresses 40-43H. These standard ports could be
;different for non-standard utilization of the PIT, i.e. user defined.
;
TIMER0        equ   40h   ;8254 channel/counter zero(0) port address
TIMER1        equ   41h   ;8254 channel/counter one(1) port address
TIMER2        equ   42h   ;8254 channel/counter two(2) port address (n/a)
```

```
TIMER_MODE    equ   43h   ;8254 PIT control_port for mode

TICS         equ    36H   ;8254 value for counter 1 – BCD value of 36 dec.
;
;********************** INIT_8254 **********************************
;The INIT_8254 procedure code which follows is used to initialize the 8254
;PIT controller into a continuous or repeating single counter, which
;divides the input frequency signal first by 65536 using counter 0.
;This produces an output of 36Hz. (given the the sample inputs above) for
;Channel zero(0) or counter 0. This is used for generation of a timer
;signal (or frequency divider). The single control word defined above
;(PIT_C0_M3) sets counter 0 to be a continuous repeating divider (mode 3)
;with a binary count divider.
;
;The output of channel 0 will then be directed to the input of the PIT
;channel/counter 1 to be divided by channel 1 count value which we will
;set to be a divide by 36 - yielding a final output (out1) of about 1Hz.
;
;So, initialize the 8254 PIT for counter 0 input/output in mode 3
;and a divide by 65536 using the above equated (equ) mnemonic names.

 mov al,PIT_C0_M3  ;load time0 mode value
 out TIMER_MODE,al ;send it to 8254 mode reg. – port 43H
;
;Next, the counter divisor is set to 0000h so that it will cause the input
;signal to be divided down by the maximum number 64,536 (64K + 1)in order to
;achieve the 36.4Hz output signal. Note: the value in the counter of 0 gives
;us a division of 65536 due to the fact that the PIT subtracts from the
;counter BEFORE it tests the value for being 0 (its terminal condition).
;In mode 3 operation, upon reaching its terminal condition of 0, the PIT
;reloads the programmed counter value and continues, i.e. repeates – this
;methodology then allows a maximum count value of 64536 (64K+1).
;
;So, load the counter divisor.  It is EXTREMELY IMPORTANT that both
;outputs (LSB and MSB of the 16 bit value) are sent because the 8254 PIT
;will wait for both values (see the mode register value defined above)
;before it starts to function. The three counters are 16 bit registers.
;
 xor al,al               ;zero out al; i.e. the counte value of 00
 out TIMER0,al           ;send LSB of div. count, then

;You MUST perform this command as well (see note above).
 out TIMER0,al           ;send MSB
;
;Counter 0 now has the value of 0 giving a count-down division of 64536
;
;Next, initialize the PIT for counter 1 in mode 3 and divide by 36.
;The technique is same as counter 0....
;Do the mode register first, then the count to the counter1
;
 mov al,PIT_C1_M3        ;load time1 mode value
 out TIMER_MODE,al       ;send it to 8254 mode reg.
;
 mov al,TICS             ;load the counter value (36 decimal in BCD format)
 out TIMER1,al           ;send LSB counter value
 xor al,al                ;zero MSB
 out TIMER1,al            ;send counter 1 MSB value
;
;             Done!! –
;
;The PIT has two counters initialized to Mode 3!! And counter 0 is
;programmed to be a repeating frequency divider generating a square wave
;output of 50% with a frequency 1/64536th of the input frequency, while
;counter 1 is programmed to be a repeating frequency divider generating a
;50% duty cycle square wave with 1/36th output of the input frequency.
;***********************************************************************
;***********************************************************************
;                  8253/4 Programmable Interval Timer (PIT)
;************************************************************************
;************************************************************************
```

Conclusion:

The 8254 PIT is a versatile device that has many applications in electronics and microcomputer systems. Understanding the potential applications means understanding how to program the device to function in its diffent modes. An example boilerplate makes their programming an easier endeavor and permits the full potential implementation in applications and as teaching tools.

References:

1. Intel Hardware Design Homepage including specifications for the 8253/8254 Programmable Interval Timer; http://developer.intel.com/
2. **The Intel Microprocessors: 8086/8088, 80186, 80286, 80386, 80486, Pentium, Pentium Pro, and Pentium II, Pentium III, Pentium 4,** Seventh Edition, Barry B. Brey, Prentice-Hall ©2005
3. **Embedded Controllers: 80186/80188 and 80386EX**, Barry B. Brey, Prentice-Hall, ©1998
4. **The 8088 and 8086 Microprocessors**, WA Triebel and A. Singh, Prentice Hall, 2000.
5. **The 80386, 80486, and Pentium Processors: Hardware, Software, and Interfacing,** Walter Tribel, Prentice Hall, 1998