SAKARYA ÜNİVERSİTESİ
Bilgisayar ve Bilişim Bilimleri Fakültesi
Bilgisayar Mühendisliği Bölümü

Mikroişlemcili Sistemler ve Laboratuvarı

KESME ÖRNEKLER

Örnek-1 Karedalga Örneği (Tek Dalga)

Example 6–1 A Square Wave Using Timer Interrupts

Write a program using Timer 0 and interrupts to create a 10 kHz square wave on P1.0.

Timer interrupts occur when the timer registers (TLx/THx) overflow and set the overflow flag (TFx). This example appears in Chapter 4 without using interrupts (see example). The bulk of the program is the same except it is now organized into the framework for interrupts. *Here's the program:*

```
0000              5            ORG     0           ;reset entry point
0000 020030       6            LJMP    MAIN        ;jump above interrupt vectors
000B              7            ORG     000BH       ;Timer 0 interrupt vector
000B B290         8    T0ISR:  CPL     P1.0        ;toggle port bit
000D 32           9            RETI
0030             10            ORG     0030H       ;Main program entry point
0030 758902      11    MAIN:   MOV     TMOD,#02H   ;timer 0, mode 2
0033 758CCE      12            MOV     TH0,#-50    ;50 us delay
0036 D28C        13            SETB    TR0         ;start timer
0038 75A882      14            MOV     IE,#82H     ;enable timer 0 interrupt
003B 80FE        15            SJMP    $           ;do nothing
                 16            END
```

Örnek-2 Karedalga Örneği (Çift Dalga)

Example 6–2: Two Square Waves Using Interrupts

Write a program using interrupts to simultaneously create 7 kHz and 500 Hz square waves on P1.7 and P1.6.

The hardware configuration with the timings for the desired waveforms is shown in Figure 6–4.

This combination of outputs would be extremely difficult to generate on a non-interrupt-driven system. Timer 0, providing synchronization for the 7 kHz signal, operates in mode 2, as in the previous example; and timer 1, providing synchronization for the 500 Hz signal, operates in mode 1, 16-bit timer mode. Since 500 Hz requires a high-time of 1 ms and low-time of 1 ms, mode 2 cannot be used. (Recall that 256 μs is the maximum timed interval in mode 2 when the 8051 is operating at 12 MHz.) Here's the program:

Örnek-2 Karedalga Örneği (Çift Dalga) - Devam

```
0000              5                 ORG     0
0000 020030       6                 LJMP    MAIN
000B              7                 ORG     000BH       ;Timer 0 vector address
000B 02003F       8                 LJMP    T0ISR
001B              9                 ORG     001BH       ;Timer 1 vector address
001B 020042      10                 LJMP    T1ISR
0030             11                 ORG     0030H
0030 758912      12     MAIN:       MOV     TMOD,#12H   ;Timer 1 = mode 1
                 13                                     ;Timer 0 = mode 2
0033 758CB9      14                 MOV     TH0,#-71    ;7 kHz using timer 0
0036 D28C        15                 SETB    TR0
0038 D28F        16                 SETB    TF1         ;force timer 1 interrupt
003A 75A88A      17                 MOV     IE,#8AH     ;enable both timer intrrpts
003D 80FE        18                 SJMP    $
                 19     ;
003F B297        20     T0ISR:      CPL     P1.7
0041 32          21                 RETI
0042 C28E        22     T1ISR:      CLR     TR1
0044 758DFC      23                 MOV     TH1,#HIGH(-1000)   ;1 ms high time &
0047 758B18      24                 MOV     TL1,#LOW(-1000)    ; low time
004A D28E        25                 SETB    TR1

004C B296        26                 CPL     P1.6
004E 32          27                 RETI
                 28                 END
```

Örnek-2 Karedalga Örneği (Çift Dalga) - Devam

Again, the framework is for a complete program that could be installed in EPF or ROM on an 8051-based product. The main program and the ISRs are located a the vector locations for the system reset and interrupts. Both waveforms are create "CPL bit" instructions; however, the timed intervals necessitate a slightly differen proach for each.

Since the TL1/TH1 registers must be reloaded after each overflow (i.e., after interrupt), Timer 1 ISR (a) stops the timer, (b) reloads TL1/TH1, (c) starts the timer,

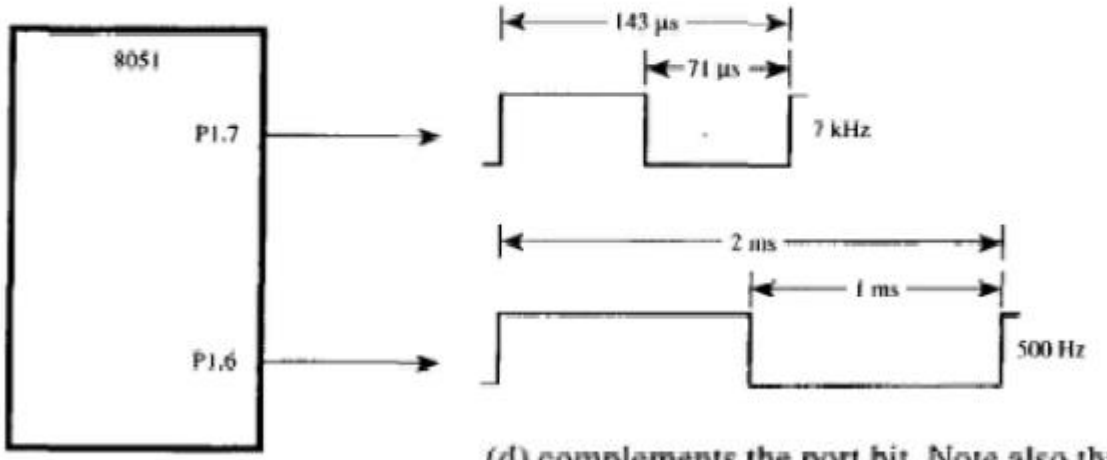

FIGURE 6–4
Waveform example

(d) complements the port bit. Note also that TL1/TH1 are *not* initialized at the beginning of the main program, unlike TH0. Since TL1/TH1 must be reinitialized after each overflow, TF1 is set in the main program by software to "force" an initial interrupt as soon as interrupts are turned on. This effectively gets the 500 Hz waveform started.

The Timer 0 ISR, as in the previous example, simply complements the port bit and returns to the main program. SJMP $ is used in the main program as the abbreviated form of HERE: SJMP HERE. The two forms are functionally equivalent. (See "Special Assembler Symbols" in Chapter 7.)

Örnek-3 Kombi Kontrolü – Harici Kesme

Example 6–4: Furnace Controller

Using interrupts, design an 8051 furnace controller that keeps a building at 20°C ± 1°C.

The following interface is assumed for this example. The furnace ON/OFF solenoid is connected to P1.7 such that

```
P1.7 = 1 for solenoid engaged (furnace ON)
P1.7 = 0 for solenoid disengaged (furnace OFF)
```

Temperature sensors are connected to $\overline{INT0}$ and $\overline{INT1}$ and provide $\overline{HOT}$ and $\overline{COLD}$ signals, respectively, such that

$\overline{HOT}$ = 0 if T > 21°C
$\overline{COLD}$ = 0 if T < 19°C

The program should turn on the furnace for T < 19°C and turn it off for T > 21°C. The hardware configuration and a timing diagram are shown in Figure 6–5.

Örnek-3 Kombi Kontrolü – Harici Kesme (Devam)

```
0000              5              ORG     0
0000 020030       6              LJMP    MAIN
                  7                                    ;EXT 0 vector at 0003H
0003 C297         8      EX0ISR: CLR     P1.7          ;turn furnace off
0005 32           9              RETI
0013             10              ORG     0013H
0013 D297        11      EX1ISR: SETB    P1.7          ;turn furnace on
0015 32          12              RETI
0030             13              ORG     30H
0030 75A885      14      MAIN:   MOV     IE,#85H       ;enable external interrupts
0033 D288        15              SETB    IT0           ;negative edge triggered
0035 D28A        16              SETB    IT1
0037 D297        17              SETB    P1.7          ;turn furnace off
0039 20B202      18              JB      P3.2,SKIP     ;if T > 21 degrees,
003C C297        19              CLR     P1.7          ; turn furnace off
003E 80FE        20      SKIP:   SJMP    $             ;do nothing
                 21              END
```

The first three instructions in the main program (lines 14–16) turn on external interrupts and make both $\overline{\text{INT0}}$ and $\overline{\text{INT1}}$ negative-edge triggered. Since the current state of the $\overline{\text{HOT}}$ (P3.3) and $\overline{\text{COLD}}$ (P3.3) inputs is not known, the next three instructions (lines 17–19) are required to turn the furnace ON or OFF, as appropriate. First, the furnace is turned ON (SETB P1.7), and then the $\overline{\text{HOT}}$ input is sampled (JB P3.2,SKIP). If $\overline{\text{HOT}}$ is high, then T < 21°C, so the next instruction is skipped and the furnace is left ON. If, however, $\overline{\text{HOT}}$ is low, then T > 21°C. In this case the jump does not take place. The next instruction turns the furnace OFF (CLR P1.7) before entering the do-nothing loop.

Once everything is set up properly in the main program, little remains to be done. Each time the temperature rises above 21°C or falls below 19°C, an interrupt occurs. The ISRs simply turn the furnace ON (SETB P1.7) or OFF (CLR P1.7), as appropriate, and return to the main program.

Note that an ORG 0003H statement is not necessary immediately before the EX0ISR label. Since the LJMP MAIN instruction is three bytes long, EX0ISR is certain to start at 0003H, the correct entry point for external 0 interrupts.