Interrupts

Interrupts Programming

- An *interrupt* is an external or internal event that interrupts the microcontroller to inform it that a device needs its service.

Interrupts vs. Polling

- A single microcontroller can serve several devices.
- There are two ways to do that:
  - interrupts
  - polling.
- The program which is associated with the interrupt is called the ***interrupt service routine*** (ISR) or ***interrupt handler***.

Steps in executing an interrupt

- Finish current instruction and saves the PC on stack.
- Jumps to a fixed location in memory depend on type of interrupt
- Starts to execute the interrupt service routine until RETI (return from interrupt)
- Upon executing the RETI the microcontroller returns to the place where it was interrupted. Get pop PC from stack

Interrupt Sources

- Original 8051 has 6 sources of interrupts
  - Reset
  - Timer 0 overflow
  - Timer 1 overflow
  - External Interrupt 0
  - External Interrupt 1
  - Serial Port events (buffer full, buffer empty, etc)
- Enhanced version has 22 sources
  - More timers, programmable counter array, ADC, more external interrupts, another serial port (UART)

Interrupt Vectors

Each interrupt has a specific place in code memory where program execution (interrupt service routine) begins.

External Interrupt 0:	**0003h**
Timer 0 overflow:	**000Bh**
External Interrupt 1:	**0013h**
Timer 1 overflow:	**001Bh**
Serial :	**0023h**
Timer 2 overflow(8052+)	**002bh**

Note: that there are only 8 memory locations between vectors.

ISRs and Main Program in 8051

```
        SJMP    main
        ORG     03H
        ljmp    int0sr
        ORG     0BH
        ljmp    t0sr
        ORG     13H
        ljmp    int1sr
        ORG     1BH
        ljmp    t1sr
        ORG     23H
        ljmp    serialsr
        ORG     30H
main:
        …
        END
```

Interrupt Enable (IE) register

- All interrupt are disabled after reset
- We can enable and disable them bye IE

D7							D0
EA	--	ET2	ES	ET1	EX1	ET0	EX0

EA	IE.7	Disables all interrupts.
--	IE.6	No implemented, reserved for future use
ET2	IE.5	Enables or disables timer 2 overflow interrupt
ES	IE.4	Enables or disables the serial port interrupt
ET1	IE.3	Enables or disables timer 2 overflow interrupt
EX1	IE.2	Enables or disables external interrupt 1
ET0	IE.1	Enables or disables timer 0 overflow interrupt
EX0	IE.0	Enables or disables external interrupt

Enabling and disabling an interrupt

- by bit operation
- Recommended in the middle of program

```
SETB   EA      SETB   IE.7      ;Enable All
SETB   ET0     SETB   IE.1      ;Enable Timer0 ovrf
SETB   ET1     SETB   IE.3      ;Enable Timer1 ovrf
SETB   EX0     SETB   IE.0      ;Enable INT0
SETB   EX1     SETB   IE.2      ;Enable INT1
SETB   ES      SETB   IE.4      ;Enable Serial port
```

- by mov instruction
- Recommended in the first of program

```
MOV IE, #10010110B
```

Example

- A 10khz square wave with 50% duty cycle

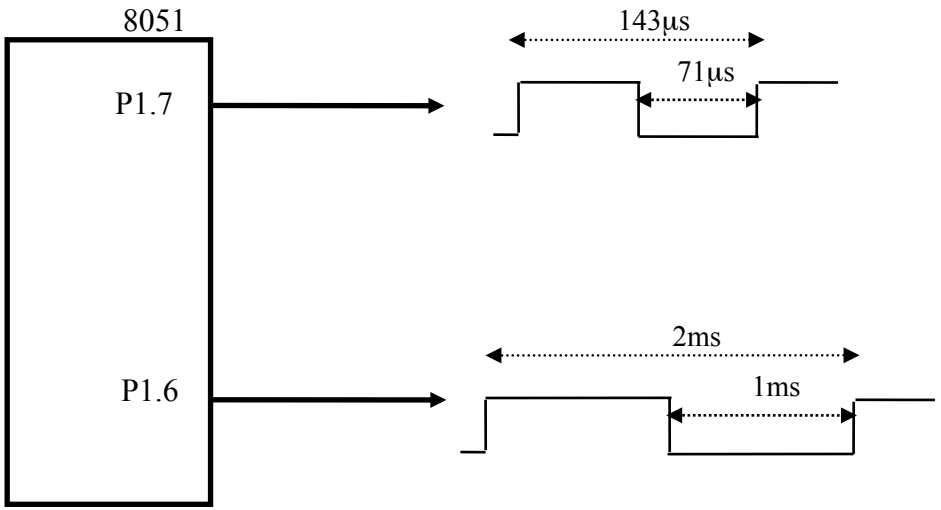
```
        ORG    0              ;Reset entry poit
        LJMP   MAIN           ;Jump above interrupt

        ORG    000BH          ;Timer 0 interrupt vector
T0ISR:  CPL    P1.0           ;Toggle port bit
        RETI                  ;Return from ISR to Main program

        ORG 0030H             ;Main Program entry point
MAIN:   MOV    TMOD,#02H      ;Timer 0, mode 2
        MOV    TH0,#-50       ;50 us delay
        SETB   TR0            ;Start timer
        MOV    IE,#82H        ;Enable timer 0 interrupt
        SJMP   $              ;Do nothing just wait
        END
```

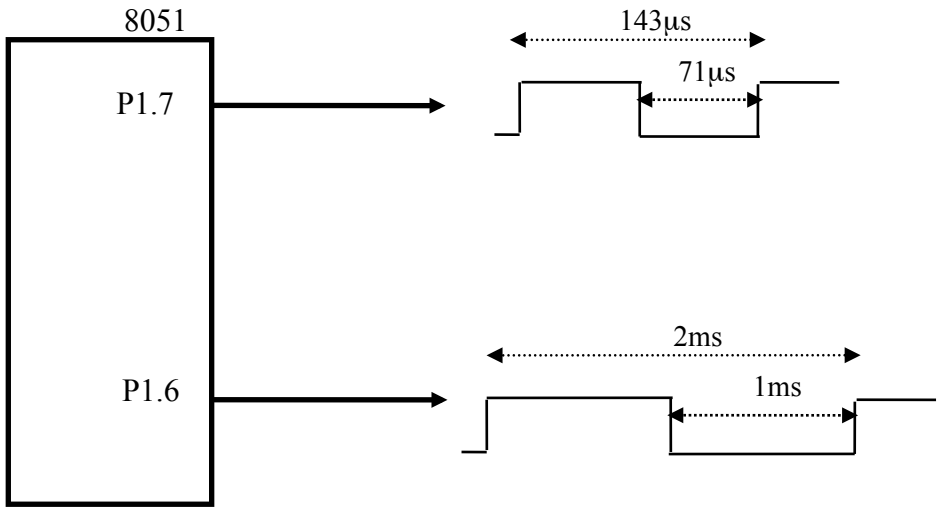
Example

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



Solution

```
        ORG     0
        LJMP    MAIN
        ORG     000BH
        LJMP    T0ISR
        ORG     001BH
        LJMP    T1ISR
        ORG     0030H
MAIN:   MOV     TMOD,#12H
        MOV     TH0,#-71
        SETB    TR0
        SETB    TF1
        MOV     IE,#8AH
        MOV     IE,#8AH
        SJMP    $
T0ISR:  CPL     P1.7
        RETI
T1ISR:  CLR     TR1
        MOV     TH1,#HIGH(-1000)
        MOV     TL1,#LOW(-1000)
        SETB    TR1
        CPL     P1.6
        RETI
        END
```



Timer ISR

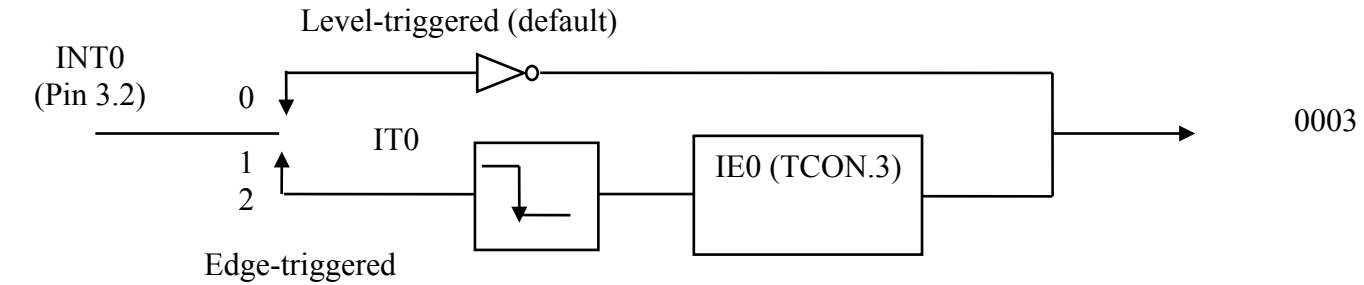
- Notice that
  - There is no need for a “CLR TFx” instruction in timer ISR
  - 8051 clears the TF internally upon jumping to ISR

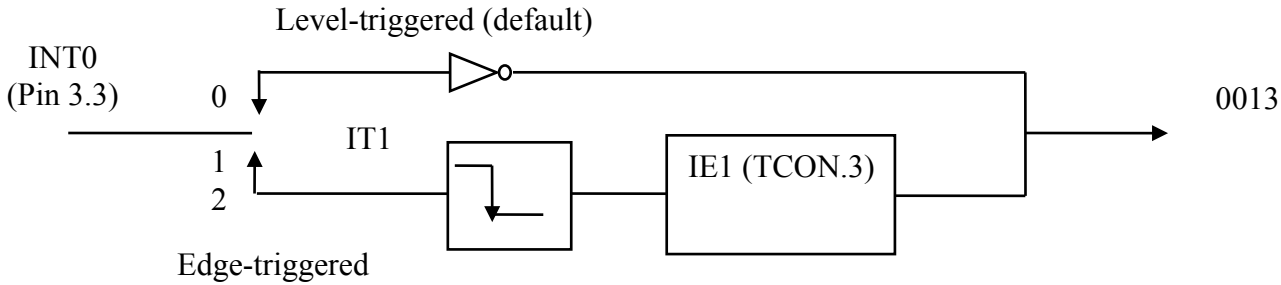
- Notice that
  - We must reload timer in mode 1
  - There is no need on mode 2 (timer auto reload)

External interrupt type control

- By low nibble of Timer control register TCON
- IE0 (IE1): External interrupt 0(1) edge flag.
  - set by CPU when external interrupt edge (H-to-L) is detected.
  - Does not affected by H-to-L while ISR is executed(no int on int)
  - Cleared by CPU when RETI executed.
  - does not latch low-level triggered interrupt
- IT0 (IT1): interrupt 0 (1) type control bit.
  - Set/cleared by software
  - IT=1 edge trigger
  - IT=0 low-level trigger

External Interrupts





Example of external interuupt

```
        ORG 0000H
        LJMP MAIN
;
;interrupt service routine (ISR)
;for hardware external interrupt INT1
;
        ORG 0013H
        SETB P1.1
        MOV R0,200
WAIT:   DJNZ R0,WAIT
        CLR P1.1
        RETI
;
;main program for initialization
;
        ORG 30H
MAIN:   SETB IT1              ;on negative edge of INT1
        MOV IE,#10000100B
WAIT2:  SJMP WAIT2
        END
```

Example of external interuupt

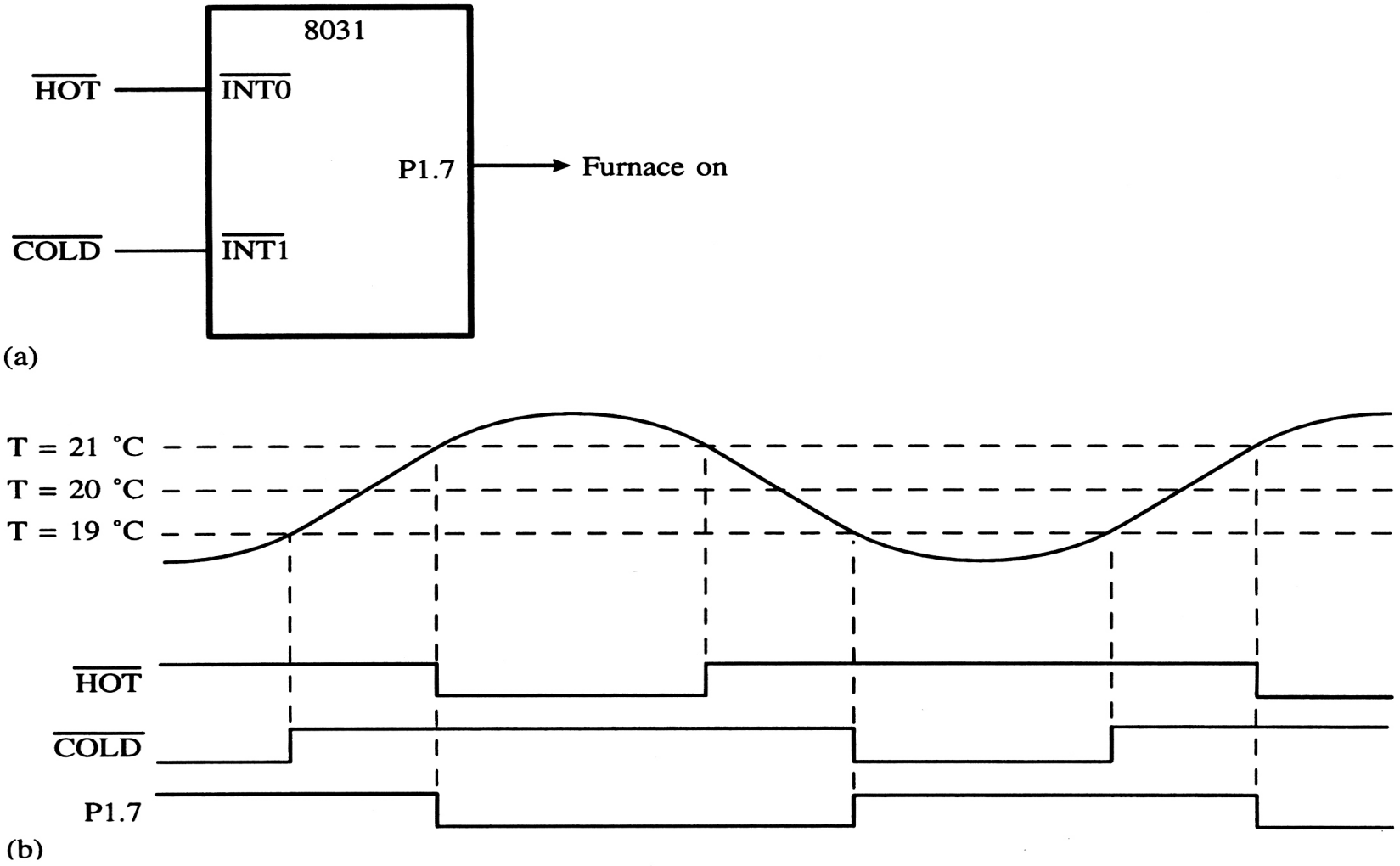


FIGURE 6–5
Furnace example. (a) Hardware connections (b) Timing.

Example of external interuupt

```
        Org 0000h
        Ljmp main

        Org 0003h
x0isr:  clr p1.7
        Reti

        Org 0013h
x1isr:  setb p1.7
        Reti

        Org 0030h
Main:   mov ie,#85h
        Setb it0
        Setb it1
        Setb p1.7
        Jb p3.2,skip
        Clr p1.7
Skip:   Sjmp $
end
```

Interrupt Priorities

- What if two interrupt sources interrupt at the same time?
- The interrupt with the highest PRIORITY gets serviced first.
- All interrupts have a power on default priority order.
  1. External interrupt 0 (INT0)
  2. Timer interrupt0 (TF0)
  3. External interrupt 1 (INT1)
  4. Timer interrupt1 (TF1)
  5. Serial communication (RI+TI)
- Priority can also be set to “high” or “low” by IP reg.

Interrupt Priorities (IP) Register

---	---	PT2	PS	PT1	PX1	PT0	PX0

IP.7: reserved

IP.6: reserved

IP.5: timer 2 interrupt priority bit(8052 only)

IP.4: serial port interrupt priority bit

IP.3: timer 1 interrupt priority bit

IP.2: external interrupt 1 priority bit

IP.1: timer 0 interrupt priority bit

IP.0: external interrupt 0 priority bit

Interrupt Priorities Example

---	---	PT2	PS	PT1	PX1	PT0	PX0

- `MOV IP , #00000100B` **or** `SETB IP.2` **gives priority order**
  1. Int1
  2. Int0
  3. Timer0
  4. Timer1
  5. Serial
- `MOV IP , #00001100B` **gives priority order**
  1. Int1
  2. Timer1
  3. Int0
  4. Timer0
  5. Serial

Interrupt inside an interrupt

---	---	PT2	PS	PT1	PX1	PT0	PX0

- A high-priority interrupt can interrupt a low-priority interrupy
- All interrupt are latched internally
- Low-priority interrupt wait until 8051 has finished servicing the high-priority interrupt