Microprocessor
Ch.11 Interrupts

OUTLINE

- 8051 Interrupts
- Timer Interrupts
- External Hardware Interrupts
- Serial Port Interrupts
- Interrupt Priority
- Interrupt Programming in C

INTERRUPTS

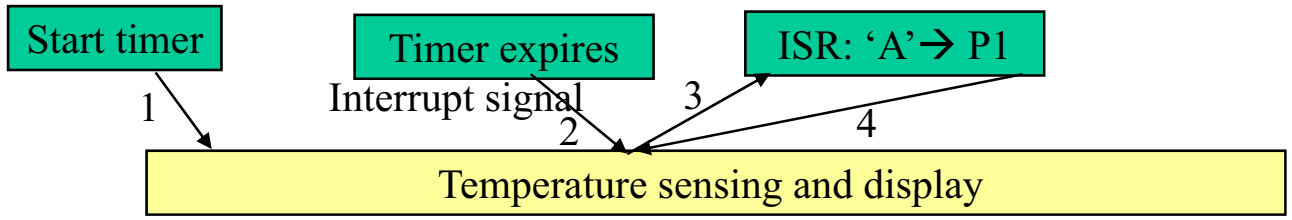
- **What is interrupt?**
  - Example: timer

```
        MOV TMOD, #02     ; 1. timer 0, mode 2 (8-bit auto reload)
HERE:   MOV TH0, #3EH     ; 2. load init value
        SETB TR0          ; 3. start timer
AGAIN:  JNB TF0, AGAIN    ; 4. monitor TF0 until timer overflows
        CLR TR0           ; 5. stop timer
        CLR TF0           ; 6. clear flag
```

  - Polling: the CPU continuously monitor the status of a device (e.g. check the status of TF0, wait until TF0 turns to 1)
    - The status monitoring process wastes a lot of MCU resources
  - Interrupt
    - Whenever any device needs service (e.g. timer expires), it will notify the MCU by sending an interrupt signal.
    - Upon receiving the interrupt signal, the CPU interrupts whatever it is doing, and serve the device (e.g. stop timer) by executing a special subroutine: Interrupt Service Routine (ISR)
    - Once ISR is done, CPU will go back to the location before interrupts.

INTERRUPTS

- **Interrupt service routine**
  - Special subroutines to be executed by CPU upon interrupt
  - Different interrupts have different subroutines



  - Example: a program needs to (1) send the letter 'A' to P1 every 100ms; (2) continuously sensing and display temperature
    - 1. start timer, then display temperature on display
    - 2. when timer expires, it sends an interrupt signal to CPU
    - 3. CPU stops temperature sensing and jump to ISR
    - 4. When ISR is over, the CPU returns to where it left and continue temperature sensing
  - Multiple devices can be connected to MCU, each of them has their unique ISRs. Whenever a device needs service, it will send interrupt to CPU and CPU can execute the corresponding ISR.

INTERRUPTS: SIX INTERRUPTS IN 8051

- **Six interrupts in 8051**
  - There are totally six different interrupts in 8051
    - Each interrupt has its own ISR
    - Each ISR has its unique starting address
    - When a particular interrupt is detected by CPU, PC will jump to the corresponding ISR starting address to execute the ISR.
  - Timer interrupts (2 interrupts)
    - 1 interrupt for timer 0 (ISR starting address: 000BH)
    - 1 interrupt for timer 1 (ISR starting address: 001BH)
  - Serial ports interrupt (1 interrupt)
    - 1 interrupt used to indicate Tx or Rx is done (ISR starting address: 0023H)
  - External hardware interrupts (2 interrupts)
    - P3.2: external interrupt 0 (INT0) (ISR starting address: 0003H)
    - P3.3: external interrupt 1 (INT1) (ISR starting address: 0013H)
  - Reset (1 interrupt)
    - when reset is activated, the 8051 jumps to address 0 (ISR address: 0H)

INTERRUPTS: INTERRUPT VECTOR TABLE

- **Interrupt vector table**
  - The starting address for each ISR

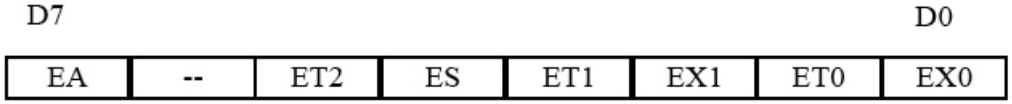
Interrupt	ROM Location (Hex)	Pin	Flag Clearing
Reset	0000	9	Auto
External hardware interrupt 0 (INT0)	0003	P3.2 (12)	Auto
Timer 0 interrupt (TF0)	000B		Auto
External hardware interrupt 1 (INT1)	0013	P3.3 (13)	Auto
Timer 1 interrupt (TF1)	001B		Auto
Serial COM interrupt (RI and TI)	0023		Programmer clears it.

  - If interrupts are enabled, ROM addresses 0003 – 0023 are taken by interrupt vector table
    - We need to bypass interrupt vector table

```
                ORG 0
                LJMP MAIN          ; bypass interrupt vector table
                ;--------------------
                ……                 ;
                                   ; ISRs can be written from 0003 – 0029 H
                ;--------------------
                ORG 30H
MAIN:           MOV A, #23H        ; main prgram
                …
```

INTERRUPTS: ENABLE INTERRUPT

- **Enable/disable interrupts**
  - Upon reset, all interrupts are disabled: the CPU will not respond to any interrupt.
  - In order to use an interrupt, it must be enabled by software
  - Register: IE (interrupt enable, bit addressable)



    - EA=0: disable all interrupts. EA=1: each interrupt is enabled individually.
    - ET2 = 1: enable interrupt for timer 2
    - ES = 1: enable interrupt for serial port
    - ET1 = 1: enable interrupt for timer 1
    - EX1 = 1: enable external interrupt 1
    - ET0 = 1: enable interrupt for timer 0
    - EX0 = 1: enable external interrupt 0
  - Example

```
MOV IE, #10010110B
MOV IE, #00010110B
```

OUTLINE

- 8051 Interrupts
- **Timer Interrupts**
- External Hardware Interrupts
- Serial Port Interrupts
- Interrupt Priority
- Interrupt Programming in C

TIMER INTERRUPTS

- **Roll-over timer flag and interrupt**
  - Recall: when timer roll-over, TF0 or TF1 is set to 1
  - Polling: HERE: JNB TF0, HERE
    - The entire CPU is tied down and waiting for TF to be set to 1
  - Interrupt
    - If timer interrupt is enabled (SETB IE.3 for timer 0, SETB IE.5 for timer 1),
      - whenever TF0 or TF1 is set to 1, a timer interrupt will be automatically generated

TIMER INTERRUPTS

- **Example**
  - Write a program that continuously (1) gets 8-bit data from P0 and sends it to P1 while simultaneously (2) create a square wave of 200 us period on P2.1

```
                ORG 0000H
                LJMP MAIN               ; by pass interrupt vector table
;-----------------timer 0 ISR ---------------
                ORG 000BH               ; timer 0 ISR
                CPL P2.1                ; toggle P2.1
                RETI                    ; return from ISR
;------------------ main program-------------
                ORG 0030H
MAIN:           MOV TMOD, #02H          ; timer 0, mode 2 (auto-reload)
                MOV P0, #0FFH           ; input
                MOV TH0, #-92           ; initial value
                MOV IE, #82H            ; IE = 10000010
                SETB TR0                ; start timer
                ;--- P0 → P1-----
BACK:           MOV A, P0
                MOV P1, A
                SJMP BACK
                END
```

1. Due to interrupt, we don't need to monitor TF0
2. No need to clear TF0!

TIMER INTERRUPTS

- **Example**
  - Write a program to generate a square wave of 50Hz frequency on P1.2.
    - 50Hz → timer delay = 10ms → initial value: 10ms/1.085us = 9216 ticks → 65535-init_value+1 = 9216 → init_value = DC00
    - We need to use mode 1 (16-bit timer, but not auto-reload)

```
            ORG 0
            LJMP MAIN
;-----------ISR for Timer 0-----------------------------
            ORG 000BH
            LCALL ISR_T0        ; store ISR elsewhere
            RETI
;-------------------------------------------------------------
            ORG 30H
MAIN:       MOV TMOD, #01H  ; timer 0, mode 1
            MOV TL0, #00H
            MOV TH0, #0DCH
            MOV IE, #82H        ; enable timer 0 interrupt
            SETB TR0            ; start timer
            SJMP $
            END
```

```
ISR_T0:
            CPL P1.2
            MOV TL0, #00
            MOV TH0, #0DCH
            RET
```

OUTLINE

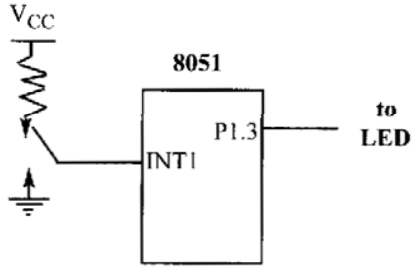
- 8051 Interrupts
- Timer Interrupts
- **External Hardware Interrupts**
- Serial Port Interrupts
- Interrupt Priority
- Interrupt Programming in C

EXTERNAL INTERRUPT

- **External interrupts INT0 and INT1**
  - INT0: P3.2, ISR starting address: 0003H
  - INT1: P3.3, ISR starting address: 0013H
  - Upon activation of these pins, the 8051 gets interrupted in whatever it is doing and jumps to the corresponding ISR
  - Two types of interrupts activation signals
    - Level triggered interrupt (default interrupt mode)
      - INT0 and INT1 is normally high
      - If a low level signal is applied to them, interrupt is triggered.
      - If the low level signal is not removed after the execution of ISR, it will be interpreted as a new interrupt.
    - Edge triggered interrupt
      - If a high-to-low signal is applied to P3.2 or P3.3, interrupt is triggered

EXTERNAL INTERRUPT: LEVEL TRIGGERED

- **Level triggered interrupt is the default mode for external interrupt**
- **Example**
  - Assume INT1 is connected to a switch that is normally high. Whenever it goes low, it should turn on an LED. The LED should stay on for a fraction of second.

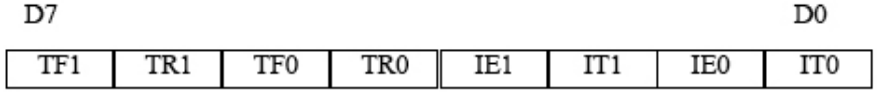



```
                ORG 0H
                LJMP MAIN
;---------------ISR for INT1-------------------
                ORG 0013H
                SETB P1.3
                MOV R3, #255
BACK:           DJNZ R3, BACK
                CLR P1.3
                RETI
;----------------Main Program---------------
                ORG 30H
MAIN:           MOV IE, #10000100B
HERE:           SJMP HERE
                END
```

D7							D0
EA	--	ET2	ES	ET1	EX1	ET0	EX0

EXTERNAL INTERRUPT: EDGE TRIGGERED

- **Edge triggered**
  - Level triggered interrupt is the default interrupt mode.
  - In order to change to edge triggered mode, we need to set the TCON register



    - TF1: timer over flow flag. (HERE: JNB TF1, HERE)
    - TR1: start or stop timer. (SETB TR1, CLR TR1)
    - IT1: interrupt mode selection. Default is 0
      - IT1 = 0: level triggered (triggered by low level)
      - IT1 = 1: edge triggered (triggered by H-to-L)
    - IE1: external interrupt 1 edge flag.
      - It's set to 1 when H-to-L is detected
      - It's automatically cleared when the interrupt is processed (after RETI is executed).
        - » When IE1 = 1, no more interrupt will be recognized → avoid interrupt in an interrupt.
      - If IT1 = 0 (level triggered), IE1 is not used at all.

EXTERNAL INTERRUPT: EDGE TRIGGERED

- **Example**
  - Assuming P3.3 (INT1) is connected to a pulse generator. Write a program in which the falling edge of the pulse will send a high pulse to P1.3, which is connected to an LED.

```
                ORG 0000H
                LJMP MAIN
    ;-----------ISR for INT1----------
                ORG 0013H
                SETB P1.3
                MOV R3, #255
BACK:           DJNZ R3, BACK
                CLR P1.3
                RETI
    ;-----------Main--------------------
MAIN:           SETB TCON.2          ; make INT1 edge-triggered interrupt
                MOV IE, #10000100    ; enable external INT1
HERE:           SJMP HERE
                END
```

TCON

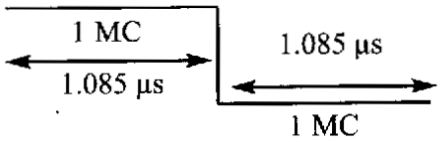
D7							D0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

IE

D7							D0
EA	--	ET2	ES	ET1	EX1	ET0	EX0

EXTERNAL TRIGGER: EDGE TRIGGERED

- **Additional notes for edge triggered interrupt**
  - Minimum pulse duration to detect edge-triggered interrupts (XTAL = 11.0592MHz)



  - Once H-to-L edge is detected, IEn will be set to 1 (IEn = IE0 or IE1)
    - IEn is called “Interrupt in service” flags
  - IEn will stay high inside ISR
    - When IEn = 1, no more H-to-L interrupts will be accepted.
  - RETI has two functions
    - Pop up the top of the stack to PC (same as RET)
    - Clear IEn to 0 (we do not need to clear IEn manually)

OUTLINE

- 8051 Interrupts
- Timer Interrupts
- External Hardware Interrupts
- **Serial Port Interrupts**
- Interrupt Priority
- Interrupt Programming in C

SERIAL PORT INTERRUPT

- **Review: serial port communication**

```
                MOV TMOD, #20H          ; timer 1, mode 2 (8-bit auto-reload)
                MOV TH1, #-3            ; 9600 baud
                MOV SCON, #50H          ; 0101 0000, initialize SCON register
                SETB TR1                ; start timer
                ;--------- Tx ----------
                MOV SBUF, #'Y'          ; store 'A' in SBUF
HERE1:          JNB TI, HERE1           ; polling TI
                CLR TI
                ;--------- Rx ----------
HERE2:          JNB RI, HERE2           ; polling RI
                MOV A, SBUF
                CLR RI
```

  - Tx: when Tx is successful, TI is set to 1
  - Rx: when a new byte arrives, RI is set to 1
  - Constantly polling TI and RI to check the status of Tx and Rx

SERIAL PORT INTERRUPT

- **Serial port interrupt**
  - Only 1 interrupt and 1 ISR assigned to serial port communication for both Tx and Rx
  - ISR entrance address: 0023H
  - When TI or RI is raised, an interrupt is generated, and PC will jump to 0023H to execute the corresponding IRS
  - How do we tell if the interrupt is generated by TI or RI flag?
    - In the ISR, we must examine the RI and TI flag to see which one triggered the interrupt.
    - Before executing RETI, we must clear RI or TI flag
      - So in the next serial port interrupt we will still be able to distinguish Tx interrupt from RI interrupt

SERIAL PORT INTERRUPT

- **Example**
  - Write a program in which the 8051 gets data from P1 and sends it to P2 continuously while incoming data from the serial port is sent to P0. Assume that XTAL = 11.0592MHz. Set the baud rate at 9600.

```
        ORG 0
        LJMP MAIN
        ORG 23H
        LJMP SER            ; jump to serial ISR
        ORG 30H
MAIN:   MOV P1, #0FFH       ; make P1 an input port
        MOV TMOD, #20H      ; timer 1, mode 2
        MOV TH1, #-3        ; 9600 baud
        MOV SCON, #50H
        MOV IE, #10010000B  ; enable serial interrupt
        SETB TR1            ; start timer 1
BACK:   MOV A, P1           ; read data from P1
        MOV P2, A           ; send it to P2
        SJMP BACK
```

```
        ORG 100H
SER:    JB TI, TRANS
        MOV A, SBUF
        MOV P0, A
        CLR RI
        RETI
TX:     CLR TI
        RETI

        END
```

D7							D0
EA	--	ET2	ES	ET1	EX1	ET0	EX0

SERIAL PORT INTERRUPT

- **Example**
  - Write a program
    - (1) Make timer 0 generate a square wave of 5KHz on P0.1
    - (2) Receive data serially and send it to P0 at 4800 baud

```
        ORG 0
        LJMP MAIN
        ORG 000BH
        LJMP TIMER_ISR
        ORG 0023H
        LJMP SER_ISR
        ORG 30H
MAIN:   MOV TMOD, #22H
        MOV TH1, #-6            ; 4800
        MOV TH0, #-92           ; delay = 100 us
        MOV SCON, #50H
        MOV IE, #10010010B      ; serial port and timer 0
        SETB TR1
        SETB TR0
        SJMP $
```

D7 D0

EA	--	ET2	ES	ET1	EX1	ET0	EX0

Interrupt	ROM Location (Hex)	Pin	Flag Clearing
Reset	0000	9	Auto
External hardware interrupt 0 (INT0)	0003	P3.2 (12)	Auto
Timer 0 interrupt (TF0)	000B		Auto
External hardware interrupt 1 (INT1)	0013	P3.3 (13)	Auto
Timer 1 interrupt (TF1)	001B		Auto
Serial COM interrupt (RI and TI)	0023		Programmer clears it.

SERIAL PORT INTERRUPT

- **Example**
  - Write a program. Once 8051 receives a character from serial port, it will send back (1) the received character (2) carriage return (ASCII: 0DH) (3) line break (ASCII: 0AH) to serial port to display them on Hyper terminal. Use interrupt for Rx and polling for Tx.

```
                ORG 0
                LJMP MAIN
                ORG 23H
                LJMP SER_ISR        ; jump to serial ISR
                ORG 30H
MAIN:
                MOV TMOD, #20H      ; timer 1, mode 2
                MOV TH1, #-3        ; 9600 baud
                MOV SCON, #50H
                MOV IE, #10010000B  ; enable serial interrupt
                SETB TR1            ; start timer 1
                SJMP $
SER_ISR:
                …….
```

Question: what will happen if we also use interrupt for Tx?

OUTLINE

- 8051 Interrupts
- Timer Interrupts
- External Hardware Interrupts
- Serial Port Interrupts
- **Interrupt Priority**
- Interrupt Programming in C

INTERRUPT PRIORITY

- **Interrupt priority**
  - What will happen if two or more interrupts are activated at the same time?
  - The 6 interrupts have different priorities
  - When two or more interrupts are activated simultaneously, interrupt with higher priority will be served first.

Highest to Lowest Priority	
External Interrupt 0	(INT0)
Timer Interrupt 0	(TF0)
External Interrupt 1	(INT1)
Timer Interrupt 1	(TF1)
Serial Communication	(RI + TI)
Timer 2 (8052 only)	**TF2**

INTERRUPT PRIORITY

- **Setting interrupt priority with the IP register**
  - The interrupt priority can be changed by programming the IP register
  - IP register (Interrupt Priority Register)

D7							D0
--	--	PT2	PS	PT1	PX1	PT0	PX0

    - When power up, all bits in IP register are 0 → default priority order
    - To assign a high priority to an interrupt, set the corresponding bit to 1 → The interrupt with IP bit set to 1 has the highest priority among all interrupts.
    - If more than one bit are set to 1, their relative priorities are determined by the default priority order
    - Interrupts with priority bits set to 1 have higher priority than interrupts with priority bits being 0.

INTERRUPT PRIORITY

- **Example**
  - Order the interrupt based on their priorities from high to low after the instruction: MOV IP, #00001100B

Highest to Lowest Priority	
External Interrupt 0	(INT0)
Timer Interrupt 0	(TF0)
External Interrupt 1	(INT1)
Timer Interrupt 1	(TF1)
Serial Communication	(RI + TI)
Timer 2 (8052 only)	**TF2**

D7 D0

--	--	PT2	PS	PT1	PX1	PT0	PX0

  - Program the IP register to assign the highest priority to INT1, then discuss what happens if INT0, INT1, and TF0 are activated at the same time

INTERRUPT PRIORITY

- **Interrupt inside an interrupt**
  - What happens if another interrupt happens during the execution of ISR?
    - If the new interrupt has a higher priority than the interrupt that is currently being responded, the CPU will
      - 1. immediately jump to the ISR of the new interrupt.
      - 2. Upon finishing the new ISR, come back to the original ISR being served.
    - If the new interrupt has a lower priority than the current interrupt, the CPU will
      - 1. finish the current interrupt,
      - 2. then jump to the ISR of the new interrupt

INTERRUPT PRIORITY

- **Triggering the interrupt by software**
  - We can manually activate an interrupt by setting the corresponding interrupt flag with software
    - By this way we can test the operation of ISR without actually incurring interrupt.
    - E.g. if the IE bit of Timer 1 is set, SETB TF1 will generate an interrupt for timer 1
  - Example

```
                ORG 0000H
                LJMP MAIN               ; by pass interrupt vector table
    ;-----------------timer 0 ISR ---------------
                ORG 000BH               ; timer 0 ISR
                CPL P2.1                ; toggle P2.1
                RETI                    ; return from ISR
    ;----------------- main program-------------
                ORG 0030H
MAIN:           MOV IE, #82H            ; IE = 10000010
                SETB TF1                ; the ISR will be executed
                END
```

OUTLINE

- 8051 Interrupts
- Timer Interrupts
- External Hardware Interrupts
- Serial Port Interrupts
- Interrupt Priority
- **Interrupt Programming in C**

PROGRAMMING IN C

- **Interrupt programming in C**
  - A unique number is assigned to each interrupt

Interrupt	Name	Numbers used by 8051 C
External Interrupt 0	(INT0)	0
Timer Interrupt 0	(TF0)	1
External Interrupt 1	(INT1)	2
Timer Interrupt 1	(TF1)	3
Serial Communication	(RI + TI)	4
Timer 2 (8052 only)	(TF2)	5

  - How to write an ISR in C

```
void timer0ISR(void) interrupt 1
{
        .......
}
```

    - The keyword “interrupt 1” followed by the name of the subroutine indicates
      - 1. this is an ISR
      - 2. It will be activated by interrupt 1 (timer 0)

PROGRAMMING IN C

- **Example**
  - Write a C program that continuously gets a single bit of data from P1.7 and sends it to P1.0, while simultaneously creating a square wave of 200 us period on pin 2.5

```
#include <reg51.h>
sbit SW = P1^7;
sbit IND = P1^0
sbit WAVE = P2^5;
void timer0(void) interrupt 1
{
        WAVE = ~WAVE;
}
void main(void)
{
        SW = 1;
        TMOD = 0x02;
        TH0 = 0XA4;
        IE = 0x82;
        while(1)
        {
                IND = SW;
        }
}
```

D7							D0
EA	--	ET2	ES	ET1	EX1	ET0	EX0

PROGRAMMING IN C

- **Example: write a C program using interrupts to do the following**
  - (1) Rx data serially and send it to P0
  - (2) Read port P1, transmit data serially, and give a copy to P2
  - (3) Make timer 0 generte a square wave of 5KHz on P0.1

```
#include <reg51.h>
sbit WAVE = P0^1;
void timer0( ) interrupt 1
{   WAVE = ~WAVE;
}
void serial0( ) interrupt 4
{
    if (RI = = 1)
    {           P0 = SBUF;
                RI = 0;
    }
}
```

```
void main(void)
{   unsigned char x;
    P1 = 0xFF;
    TMOD = 0x22;
    TH1 = 0xF6;             // 4800 baud
    SCON = 0x50;
    TH0 = 0xA4;             // 5KHz, delay = 100us
    IE = 0x92;
    TR1 = 1;
    TR0 = 1;
    while(1)
    {           x = P1;
                SBUF = x;
                while (TI = = 0);
                TI = 0;
                P2 = x;
    }
}
```

D7 D0

EA	--	ET2	ES	ET1	EX1	ET0	EX0