8051 - Interrupts

EE4380 Fall 2001

Class 9

Pari vallal Kannan

Center for Integrated Circuits and Systems

University of Texas at Dallas

Polling Vs Interrupts

- Polling:
  - MCU monitors all served devices continuously, looking for a “service request flag”
  - Whenever it sees a request, it serves the device and then keeps polling
  - MCU is always “busy” with polling doing the “while any request” loop
- Interrupts
  - If and when a device is ready and needs attention, it informs the MCU
  - MCU drops whatever it was doing and serves the device
  - MCU is always “free”, when not serving any interrupts

Interrupt Service Routine

- MCUs have fixed number of interrupts
- Every interrupt has to be associated with a piece of code called “Interrupt Service Routine”, or ISR.
  - If interrupt-x is received by MCU, the ISR-x is executed
- MCU architecture defines a specific “code address” for each ISR, which is stored in the,
  - “Interrupt vector Table IVT”
- ISRs are basically “subroutines”, but they end with the ***RETI,*** instruction instead of RET
- When an interrupt occurs, the MCU fetches its ISR code address from the IVT and executes it.

Interrupt Execution

1. MCU finishes the instruction it is currently executing and stores the PC on the stack
2. MCU saves the current status of all interrupts internally
3. Fetches the ISR address for the interrupt from IVT and jumps to that address
4. Executes the ISR until it reaches the RETI instruction
5. Upon RETI, the MCU pops back the old PC from the stack and continues with whatever it was doiing before the interrupt occurred

8051 Interrupts

- Vendors claim 6 hardware interrupts. One of them is the reset. So only 5 real interrupts in the 8051. Clones may differ.
- Two external interrupts – INT0 and INT1, two timer interrupts – TF0 and TF1 and one serial port interrupt – S0
- Interrupts can be individually enabled or disabled. This is done in the IE (Interrupt Enable Register)
- External interrupts (INT0 and INT1) can be configured to be either level or edge triggered.

8051 - IVT

- Each Interrupt has 8 bytes for its ISR.
- If ISR is too big to fit in 8bytes, then use a ljmp

```
                ORG 0
rom_start:      LJMP main_code
                ORG 13H
int1_vec:       LJMP int1_isr
                ORG 30H
main_code:                  ;bla bla
; ….
int1_isr:                   ;bla bla
```

Interrupt	ROM Location	Pin
Reset	0000H	9
INT0	0003H	P3.2
TF0	000BH	
INT1	0013H	P3.3
TF1	001BH	
S0	0023H	

IE Register

- EA = 0, disable all interrupts

EA	--	ET2	ES	ET1	EX1	ET0	EX0

- Other bits if set to 1, enable the corresponding interrupt, if set to 0, disable it.
- EX0 = enable INT0
- ET0 = enable Timer0
- EX1 = enable INT1
- ET1 = enable Timer1
- ES = enable serial port interrupt
- ET2 = (for 8052 clones only) enable Timer2
- Interrupts can be triggered by software by setting the bits in IE
  - setb IE.1

Simple Example

- INT1 pin is connected to a switch that is normally high. Whenever it goes low, an LED should be turned on. LED is connected to port pin P1.3 and is normally OFF

```
            org 0H
            ljmp MAIN
            org 13H              ;INT1 ISR
INT1_ISR:   setb P1.3            ;turn on LED
            mov r3, #255
BACK:       djnz r3, BACK        ;keep the led ON for a while
            clr P1.3             ;turn OFF the LED
            RETI                 ;use RETI,  ***NOT RET***
            org 30H
MAIN:       mov IE, #1000 0100B  ;enable INT1, EA=1, EX1=1
HERE:       sjmp HERE            ;stay here until interrupted
            end
```

External Interrupts

- INT0 and INT1
  - Level triggered : a low level on the pin causes interrupt – Default mode
  - Edge triggered : a high-to-low transition on the pin causes interrupt
- Configuration in TCON register
  - (IT1) TCON.2 = 1 ➔ INT1 is edge triggered
  - (IT0) TCON.0 = 1 ➔ INT0 is edge triggered
- IE0 (TCON.1) and IE1 (TCON.3)
  - In edge triggered mode, if interrupt INTx occurs, the MCU sets the IEx bit, which is cleared only aftera RETI is executed
  - Prevents interrupt within interrupt
- Setup and Hold times for Edge triggered external interrupts
  - One machine cycle each

Interrupt Priority

- Default Priority
  - INT0 > TF0 > INT1 > TF1 > S0
- The ISR of an interrupt can be “interrupted” by a higher priority interrupt.
- The Default Priority can be changed by programming the IP register

--	--	PT2	PS	PT1	PX1	PT0	PX0

- To set higher priority to an interrupt, set its bit in IP to 1
- If more than one 1 in IP, the default priority is used for all the interrupts that have 1 in IP

8051 - Timers

- Two 16-bit timers T0 and T1
  - Timer - calculate timing, time etc
  - Event counter – Count the occurrence of an event
- T0 = TH0:TL0
- T1 = TH1:TL1
- Timer mode is controlled by TMOD register
  - Gate, C/T, M0, M1
- Timers are controlled by TCON register (upper 4 bits)
  - TR0, TR1, TF0, TF1

8051 Timer : TMOD Register

Gate1	C/T1	M1	M0	Gate0	C/T0	M1	M0

- Gate = 0, software gate of Timer (TRx bit in TCON)
- Gate = 1, hardware gate of Timer (INTx) pin
- C/T = 0 ➔ Timer operation
- C/T = 1 ➔ Counter operation
- M1:M0 = 00 → Mode 0 (13bit timer)
- M1:M0 = 01 → Mode 1 (16 bit timer)
- M1:M0 = 10 → Mode 2 (8 bit timer, with auto-reload)
- M1:M0 = 11 → Mode 3 (split timer)
- Clock source for the timer is sys_clk/12

Timer – Mode 1

- 16 bit timer
- Operation
  1. Load TMOD register to set mode
  2. Load TLx and THx with the initial count values
  3. Start timer (setb TRx)
  4. Keep monitoring TFx flag (jnb TFx, target)
  5. Stop timer (clrb TRx) and clear TFx flag
  6. Go back to step 2 to load again
- Time spent
  - Telapsed = (65536 - intial_value)*cycle_time
- Instead of polling for TFx flag, an ISR could be used

Timer : Mode –1 Example

- Generate a 50% duty cycle square wave on P1.5, with Timer0

Timer0 sequence
FFF2 TF=0
FFF3 TF=0
FFF4 TF=0
….
….
FFFE TF=0
FFFF TF=1

```
        mov TMOD, #01       ;Timer 0, mode 1
Here:   mov TL0, #0F2H
        mov TH0, #0FFH      ;Initial Value = FFF2H
        cpl P1.5
        acall delay
        sjmp Here
Delay:  setb TR0            ;start Timer0
Again:  jnb TF0, Again      ;poll for TF0 (timer overflow)
        clr TR0             ;stop timer
        clr TF0             ;clear TF0 flag
        RET
```