# Chapter 09

# Programming in Assembly

# Lesson 07

# Programming Examples for Real Time Clock Interrupts

# Real time clock interrupts

- Used to perform real time tasks after regular periodic intervals
- The task which runs  on real time clock interrupts can be assigned highest priority
- Use of mode 2 and mode 0 for initiating the RTC tasks at the periodic intervals

# Program for the successive overflows and interrupts

- Overflows and interrupts After intervals of 4.096 ms
- Use mode 2 of T0
- Assume that Xtal frequency 12 MHz
- Also program for obtaining square-wave pulses at P2 pin 0 of time-period 8.192 ms and frequency 122 Hz

# Registers Used

- Program uses the registers R7, R6, A, TL0, TH0, TMOD and IE.
- Program affects the registers R7, R6, TL0, TH0, TMOD and IE.

# Program (Main Program)

(i) ANL TMOD, #F2 ; Define T0 mode 2, internal clock inputs and internal start/stop

(ii) MOV TH0, #00H ; TH0 reload into TL0 on overflow of TL0

(ii) MOV TL0, #00H ; TL0 overflows after 256 clock inputs

# Calculation for Number of Interrupts for actions

- Internal clock intervals = 1 μs
- Overflow required after 4.096 ms
- Means after 4.096 ms/ 1μs = 4096 internal clock inputs = 1000H = $2^{12}$ = 256 × 16
- When we are using the mode 2, there is only 8-bit timer TL0 required
- Therefore the ISR running at every 16th instance should take action of toggling port bit

# Program (Main Program)

- (iii) SETB EA ; Enable interrupts
- (iv) MOV R7, #10H ; Initial value = 16 for variable numISRT0.
- (v) MOV R6, #10H ; Initial value = 16 for constant numISRT0Init.

# Register Used

- Assume R6 used to save the value of constant numISRT0Init (number of ISR counts initial value)

# Registers Used

- R7 used as down-counter for the number of times the ISR T0 starts and runs
- It holds the variable numISRT0
- R7 written the numISRT0Init from R6 on successive 16th instances

# Program (Main)

(vi) SETB PT0 ; Define priority of T0 interrupts high

(vii) ANL IP, # 02H ; Define priority all low except PT0 IP bit 1for T0 interrupts

(viii) SETB ET0 ; Enable T0 interrupts

(ix) SETB TR0 ; Run timer T0

# Interrupt Service Routine for T0

- 8-byte instructions between 0x000B and 0x012

(i) DJNZ R7, End ; No action till 16th instance
;of the run of ISR T0; Decrement
; numISRT0 and jump if does not become =0

(ii) XCH A, R6; Exchange numISRT0Init in R6 with A

(iii) MOV R7, A ; Copy R7 from A. R7 = numISRT0Init

# Interrupt Service Routine for T0

- 8-byte instructions between 0x000B and 0x012

(iv) XCH A, R6 ; Exchange R6 with A.
This restores numISRT0Init in R6

(v) ACALL RTCtasks ; Call RTC tasks

(vi) End: RETI ; Return from Interrupt

# Real Time Clock (RTC) tasks

- RTCtasks: CPL P2^0 ; The instructions for RTC tasks
- RET ; Return from RTC tasks

# Summary

# We learnt

- Real Time Clock Interrupts generated
- Timer used for the program
- Number of interrupts counted and real time actions taken every $16^{th}$ interrupt for 4.096 ms.

# End of Lesson 07 on

## Programming Examples for Real Time Clock Interrupts