CpE 213
Digital Systems Design

Lecture 24

Tuesday 11/18/2003

UMR UNIVERSITY OF MISSOURI-ROLLA
The Name. The Degree. The Difference.

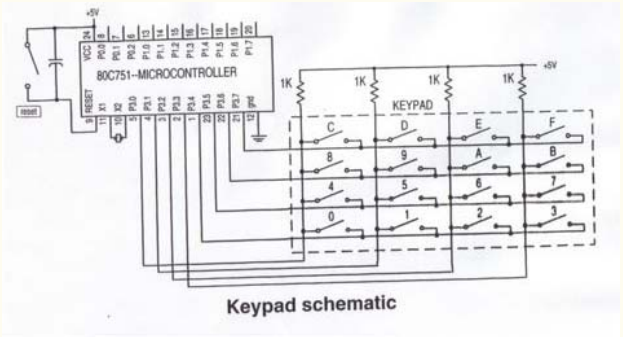
Announcements

- HW 8 has been posted.
  - Due date: Tuesday Nov. 25.
- Project peer review has been posted.
  - Please keep the ratings in mind while working on the project.
- Project demos are due on Monday 12/1.
  - Signup sheet is on my door. The spokesperson for each group should sign up for a slot on the schedule.
  - The sheet is posted on Blackboard for your reference.
  - All group members must be present at demonstration.
  - Deadline for signup is Tuesday 11/25.
  - Send me email by 11/21 if you have a valid schedule conflict all day on Monday 12/1.

Outline

- Discussion of exam question 6.
- Review of interrupt examples and timing.
- Introduction to timers and counters.

Exam 2 Question 6



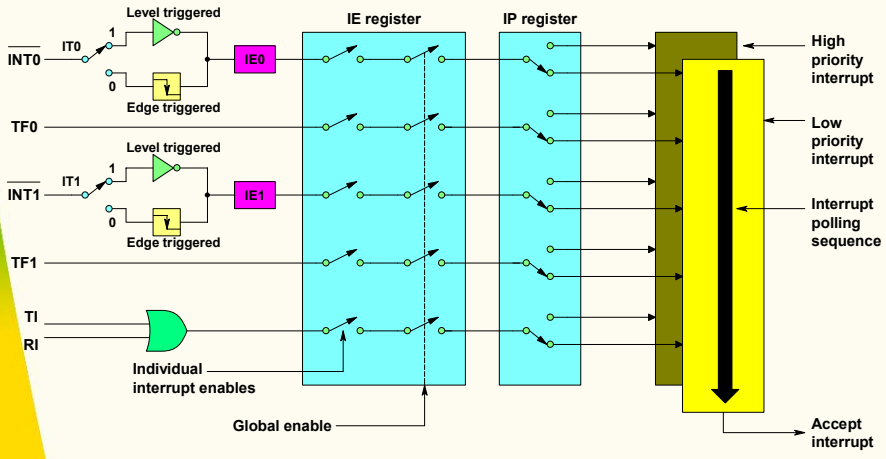
Keypad schematic

```
unsigned char pat;
unsigned int result;

result = 0x0000;

for (pat = 0x80 ; pat > 0x08 ; pat >> 1) {
   P3 = ~pat;
   result = (result | (P3 & 0x0F)) << 4;
}
```

What is the state of keys if result is 5280H after execution of program?

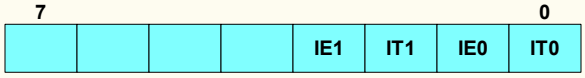
Review of Interrupts

Overview of 8051 Interrupt Structure



TCON's Interrupt-Related Bits

TCON (Timer Control) Register



Symbol	Function
IE1	External interrupt 1 edge flag. Set by hardware when a falling edge is detected on INT1; cleared by software or by hardware when CPU vectors to interrupt service routine.
IT1	External interrupt 1 type flag. Set/cleared by software for falling edge/low-level activated external interrupt.
IE0	External interrupt 0 edge flag.
IT0	External interrupt 0 type flag.

Another Example (review)

```
#include <reg51.h>
static unsigned char count=0;
void counter(void) interrupt 0 {
   count++;
}
main(){
   unsigned char x = 0;
   int i;

   /* set up external interrupt 0 */
   IT0=0;      /* interrupt on falling edge of INT0 */
   INT0=1;     /* configure for input */
   PX0=0;      /* set priority level */
   EX0=1;      /* enable external interrupt 0 */
   EA=1;       /* global interrupt enable */
```

(continued on next slide)

Example adapted from Dr. Beetner

Example continued

```
while(1){
   for(i=1;i<=10;i++) {
       x++;         /* dummy background operation */

       /* simulate a falling edge on interrupt */
       if(x==128){
             x=0;
             INT0=1;
             INT0=0;
             INT0=1;
       }
   }
}
```

Example in ASM

```
;declare segments
mydata segment data
mycode segment code

;declare variables
rseg mydata
   x: DS 1
   count: DS 1
   stack: DS 20

;place jump to main at 1st code mem location
CSEG AT 0000H
   JMP main ; will jump to main function
```

Example continued

```
; interrupt service routine
; place to respond to external interrupt 0
CSEG AT 0003H              ; start ISR at correct location
       PUSH ACC            ; save state of uC
       PUSH PSW
       MOV A,count
       ADD A,#1            ; count++
       MOV count, A
       POP PSW             ; restore state of uC
       POP ACC
       RETI                ; REQUIRED!

; main function
rseg mycode
 main: MOV x,#00H          ; initialize x
       mov count,#00H      ; initialize count
       MOV SP, #stack      ; initialize stack pointer
```

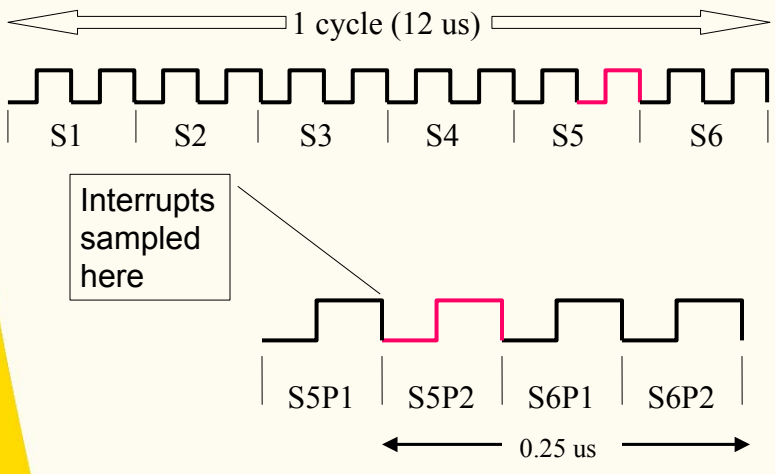
Example continued

```
  SETB IT0
  SETB INT0
  CLR PX0
  SETB EX0
  SETB EA

 while: INC x              ; x++
  MOV A,#128
  CJNE A,x,while           ; if x==128 …
  SETB INT0                ; simulate interrupt
  CLR INT0
  SETB INT0
  JMP while                ; end of while loop
END
```

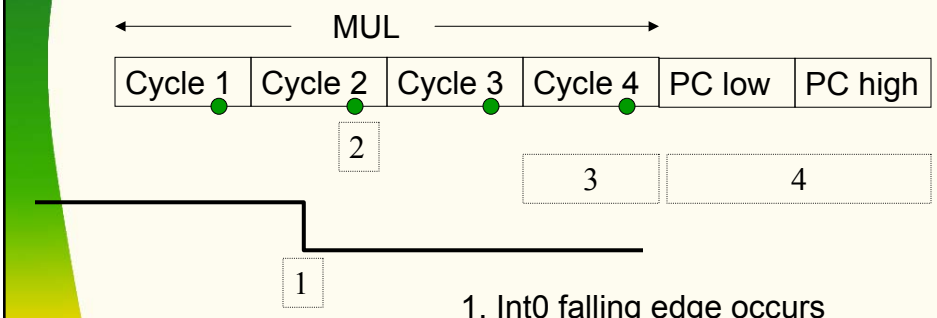
Interrupt Timing

- Interrupts sampled during S5P2
- Interrupts polled during last cycle of instruction
- PC pushed onto stack (2 cycles)
- Fetch first instruction of ISR
- A RETI or access to IE or IP lets one more instruction execute before pushing PC

Interrupt Timing



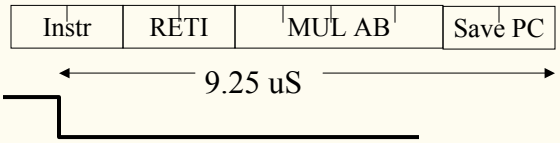
Interrupt Timing



- Interrupts Sampled

1. Int0 falling edge occurs
2. Ex0 interrupt detected
3. Interrupts polled. Start Ex0ISR
4. Push PC on stack (LCALL 3)

Worst Case Interrupt Timing



1. Level 1 event occurs just after sample in second to last cycle of instruction.
2. Interrupt sampled in last cycle of Instr
3. Interrupt polled in last cycle of RETI
4. RETI allows one more instruction: 4 cycle MUL
5. Plus two cycles to save PC
6. Equals 9.25 uS worst case (or is it really?)

Timers and Counters for the 8051

Slides adapted from Mr. K. T. Ng

What Is a Timer/Counter ?

- A counter is a device that generates binary numbers in a specified count sequence when triggered by an incoming clock pulse.
- Timers are counters that count pulses. If the pulses are “clock” pulses, then the timers count time.

Timers/Counter on the 8051

- The 8051 comes up with two 16-bit timers/counters, both of which may be controlled, set, read, and configured individually.
- The 8051 timers have three general functions:
  - Keeping time and/or calculating the amount of time between events (Timer Mode).
  - Counting the events themselves (Event Counter Mode)
  - Generating baud rates for the serial port.
- Always count UP irrespective of function as a timer or a counter.

Difference between a Timer and a Counter

- When the incoming clock frequency is known, we can generate a fixed period of time known to the designer by setting a preloaded value. This is called a "timer".
- It is also called an "interval timer" since it is measuring the time of the interval between two events.
- When the incoming clock is irregular and we are only interested in the number of occurrences of the pulse, this is called a "counter". Since we are counting events, is also known as an "event counter".
- In 8051, timer gets its clock from the oscillator (crystal that connected to the CPU) frequency (1/12 of it).
- Counter gets the clock from an external pin:
  - P3.4 for timer 0 and
  - P3.5 for timer 1.

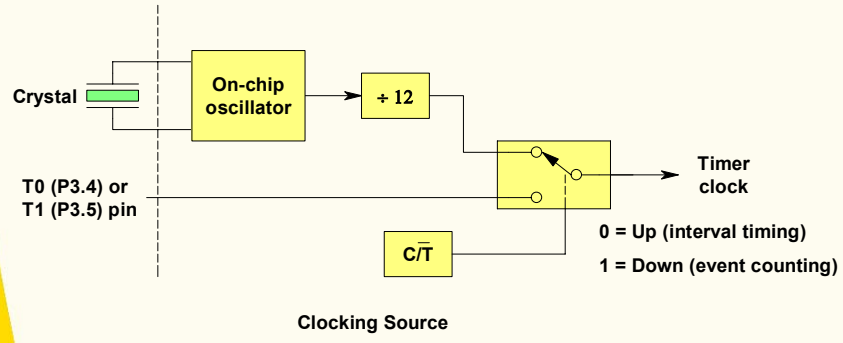
Difference between a Timer and Counter (Cont.)

- Timer mode (if the timer control bit is set to timer, the timer uses the system clock)
  - Increments by 1 every machine cycle.
  - A single machine cycle consists of 12 crystal pulses.
- Event mode (if the timer control bit is set to event, the timer/counter uses a port bit)
  - Increments by 1 for each pulse on P3.4 for Timer 0, and P3.5 for Timer 1, respectively.

Applications of Timers/Counters

- Generating time delays.
- Measuring pulse duration or timing intervals.
- Counting pulses or events.
- Generating baud rate clock for the internal 8051 serial I/O port.
- Generating interrupts.

Timer/Counter Clocking Source



Clocking Source

Group Exercise

- Find the timer's clock frequency and period for various 8051-based systems with the following crystal frequencies:
  - 12 MHz
  - 16 MHz

Timer Special Function Registers (SFRs)

- As mentioned before, the 8051 has two timers that essentially function the same way.
- One timer is called TIMER0 and the other is TIMER1.
- The two timers share two SFRs (TMOD and TCON) that control the timers.
- Each timer also has two SFRs dedicated solely to itself (TH0/TL0 and TH1/TL1).
- It is common practice to use given SFR names to refer to them, but in reality an SFR has a numeric address.
- When you enter the name of an SFR into an assembler, it internally converts it to the specified SFR address.

Location of Timer SFRs

SFR Name	Description	Address
TH0	Timer 0 High Byte	8CH
TL0	Timer 0 Low Byte	8AH
TH1	Timer 1 High Byte	8DH
TL1	Timer 1 Low Byte	8BH
TCON	Timer Control	88H
TMOD	Timer Mode	89H

Timer Control Register (TCON)

- TCON has the following functions:
  - Turns Timer 0 and Timer 1 on or off.
  - Sets trigger type (edge or level) for interrupt 0 and interrupt 1.
  - Sets flags when timer 0 or timer 1 overflow.

Summary of TCON

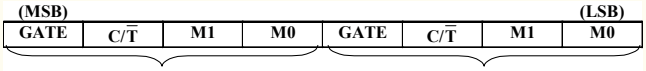
Bit	Symbol	Bit Address	Description
TCON.7	TF1	8FH	Timer 1 overflow flag. Set by hardware upon overflow; cleared by software, or by hardware when processor vectors to interrupt service routine
TCON.6	TR1	8EH	Timer 1 run-control bit. Set/cleared by software to turn timer on/off
TCON.5	TF0	8DH	Timer 0 overflow flag
TCON.4	TR0	8CH	Timer 0 run-control bit
TCON.3	IE1	8BH	External interrupt 1 edge flag. Set by hardware when a falling edge is detected on $\overline{\text{INT1}}$; cleared by software, or by hardware when CPU vectors to interrupt service routine
TCON.2	IT1	8AH	External interrupt 1 type flag. Set/cleared by software for falling edge/low-level activated external interrupt
TCON.1	IE0	89H	External interrupt 0 edge flag
TCON.0	IT0	88H	External interrupt 0 type flag

- Only the upper 4 bits are related to timers; the lower 4 bits have nothing to do with timers - they have to do with interrupts.
- TCON is a bit-addressable register.

Timer Mode Control (TMOD) Register

- TMOD controls timer 0 and timer 1 functions.
- The register's upper nibble controls Timer 1 and the lower nibble controls timer 0.
- The four bits for each timer control a gate function, selection of counter or timer, and timer mode.
- The timer or counter function is selected by control bits C/T; these two timers/counters have 4 operating modes, selected by appropriate M0 and M1 bits in the TMOD register.
- TMOD is not a bit-addressable register. It has to be manipulated as a byte.

Timer Mode (TMOD) Register Summary



Timer 1 | Timer 0

GATE If GATE = 1, timer x will run only when TRx = 1 and INTx = 1. If GATE = 0, timer x will run whenever TRx = 1.

C/T Timer mode select. If C/T = 1, timer x runs in counter mode taking its input from Tx pin. If C/T = 0 , timer x runs in timer mode taking its input from the system clock.

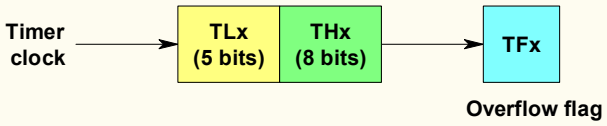
M1	M0	Operating Mode
0	0	13-bit timer mode (8048 mode).
0	1	16-bit timer mode.
1	0	8-bit auto-reload mode. "THx" holds a value which is to be reloaded into "TLx" each time it overflows.
1	1	(Timer 0) TL0 is an 8-bit Timer/Counter controlled by the standard Timer 0 control bits. TH0 is an 8-bit timer only controlled by Timer 1 control bits.
1	1	(Timer 1) Timer/Counter 1 is stopped.

Group Exercise

- Indicate which mode and which timer are selected for each of the following:
  - `MOV TMOD,#01H`
  - `MOV TMOD,#20H`
  - `MOV TMOD #12H`

Mode 0 - 13 Bit Counter

- TH, TL = XXXXXXXX, 000XXXXX
- 8 bits from TH
- 5 bits from LSB of TL
- For compatibility with older microcontroller (8048), not very useful now.
- Range = 0 to 8191.

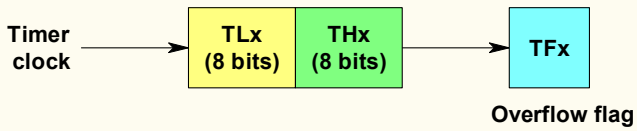


8051 Timer Mode 0

Timer Mode 1 - 16 Bit Counter

- 8 bits from TH, another 8 bits from TL.
- Each byte loaded separately
- Counts UP, so initialize with negative value.
- The counters start counting when enabled (by a bit in the SFR).
- Counts time (when acting as timer) until a preset number as specified by the THx and TLx registers is reached.
- Generates an overflow (interrupt) when timed out.
- In order to repeat the process the registers TH and TL must be reloaded with the original value and the timer overflow flag must be reset under the control of the program.

Timer Mode 1 (Cont.)



8051 Timer Mode 1

- Example

```
TH0= -1000 >> 8; //-1000. = 0xFC18
TL0= -1000 & 0xFF;
```

- Why doesn't -1000/256 work?

Finding Values to be Loaded into the Timer

- Assume that we know the desired amount of timer delay, t_{delay}. To calculate the values to be loaded into the TL and TH registers, using crystal frequency of $X_{crystal}$ MHz, the following steps are needed:
  - Step 1: find n, where n = $\frac{t_{delay} \bullet X_{crystal}}{12}$
  - Step 2: Perform 65536 - n
  - Step 3: Convert step 2 result to hex, assume this result is yyxx
  - Step 4: Set TL = xx and TH = yy.

Group Exercise

- Assuming a desired time delay of 5 ms and Xcrystal = 11.0592 MHz, determine the values of TL and TH in hex.

A Better Way

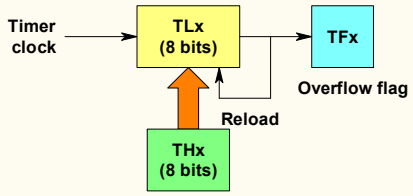
- A better way is to let the assembler do the arithmetic.
- Suppose the required delay count number, n, is 4608. When using timer0, we can load this value into TH0 and TL0 as follows:

```
MOV TH0,#-4068 SHR 8    ;MSB of -4068
MOV TL0,#-4068          ;LSB of -4068
```

- What does SHR 8 mean ?
  - It means the assembler would take the 16-bit number and shift it right by 8 bits. This gives us the MSB of -4068.
- The assembler automatically selects only the bottom 8 bits of the value, so the LSB of -4068 goes into TL0.

Mode 2 - 8 Bit Auto-Reload

- TL0 or TL1 is an 8 bit counter/timer
- TH0 or TH1 holds an initialization value
- TL reloaded from TH on 255 to 0 transition
- The reload leaves TH unchanged.
- Used to generate UART baud rate clock; a periodic flag or interrupt.

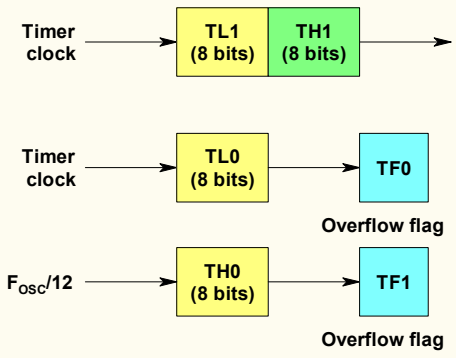


8051 Timer Mode 2

Timer Mode 3 - Three from Two

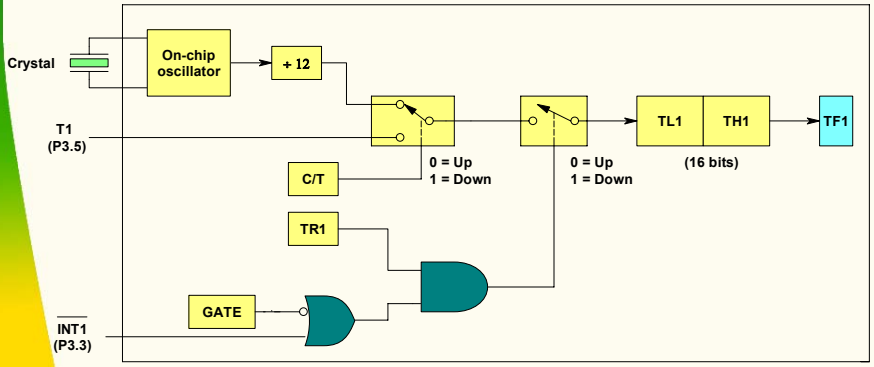
- T0 is split into two 8-bit counter/timers.
- First counter (TL0) act like mode 0.
- Second counter (TH0) counts CPU cycles.
  - Uses TR1 (timer 1 run bit) as enable.
  - Uses TF1 (timer 1 overflow bit) as flag.
  - The clock frequency is $F_{osc}/12$
- T1 becomes a 16-bit timer that can be started and stopped at any time by the mode operating bits M0 & M1, but cannot cause an interrupt.
- T1 is normally in mode 2 when T0 in mode 3
- T1 is used for baud rate clock while T0 provides two 8-bit counters.

Timer Mode 3 (Cont.)



8051 Timer Mode 3

C/T' and GATE Bits



Timer 1 operating in mode 1

C/T' and GATE Bits (Cont.)

- C/T' is used as the timer or counter select bit. It is cleared for timer operation (input from internal system clock), and set for counter operation (input from Tn input pin).
- When the Gate control is set, Timer/Counter n is enabled only while INTn pin is high and TRn control pin is set. When cleared, Timer n is enabled whenever TRn control pin is set.
- Setting Gate = 1 allows us a way of controlling the stop and start of the timer externally at any time via a simple switch.