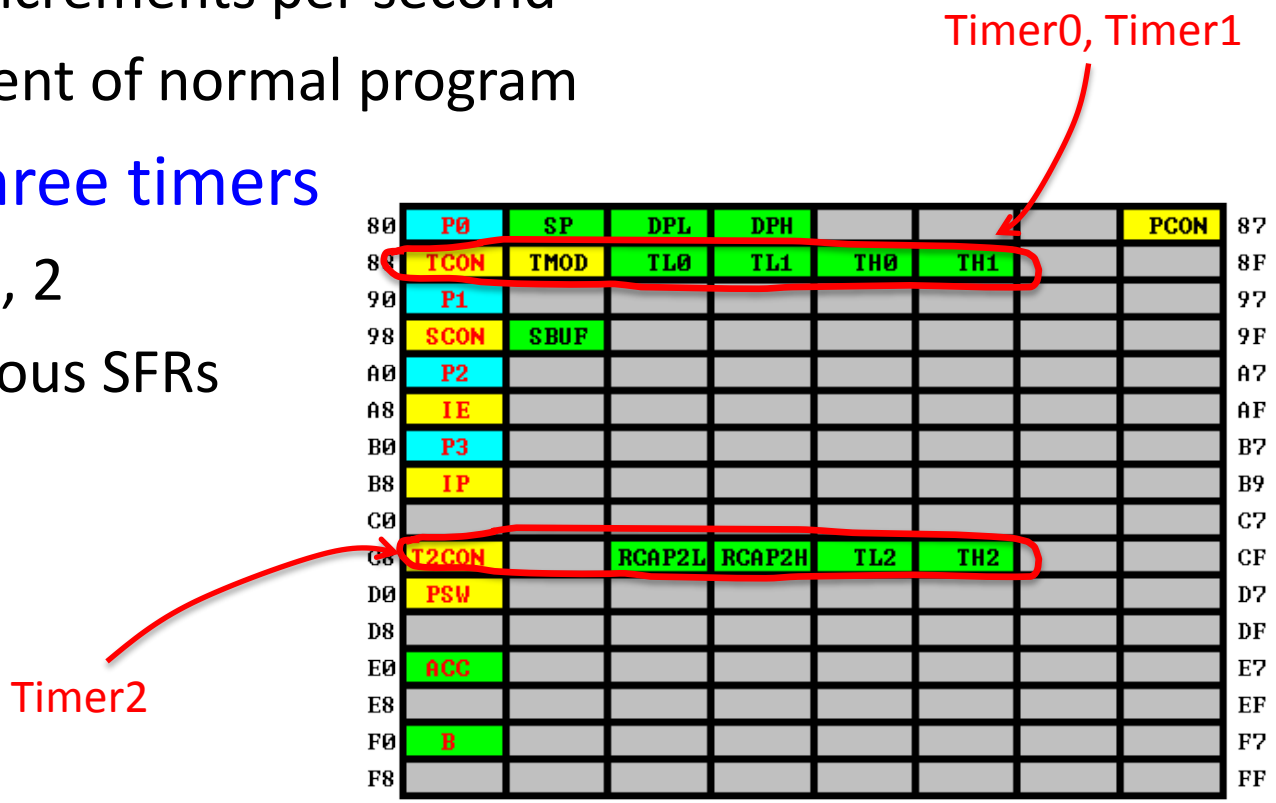
Timers and interrupts

Overview

- Timers
  - Creating fixed pauses
  - Calculate length of events
  - Counts events
  - Generate baud rate for serial communication
- Interrupts
  - Allow routine to interrupt normal execution
  - Can be triggered by timers

Timer basics

- Increment every instruction cycle
  - 11,059,200 ticks second / 12 ticks per cycle
  - 921,600 increments per second
  - Independent of normal program
- 8052 has three timers
  - Timer 0, 1, 2
  - Using various SFRs



Timer 0/1

- Timer0, Timer1
  - Provide exactly same functionality
  - Allows timing or counting two different things
  - Four different modes:
    - Mode 0 = 13 bit timer (legacy)
    - Mode 1 = 16-bit timer
    - Mode 2 = 8-bit auto-reload
    - Mode 3 = Split timer

Timer mode 1

- Mode 1, 16-bit timer
  - Counts from 0 up to 65535, rolls around to 0
  - High byte in TH0/TH1, low byte TL0/TL1
    - e.g. TH0 = 3, TL0 = 232, value = 256 * 3 + 232 = 1000
  - Set mode 1 using two bits in TMOD

TMOD	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	GATE1	C/T1	T1M1	T1M0	GATE0	C/T0	T0M1	T0M0

0 1 = mode 1 for timer 1

0 1 = mode 1 for timer 0

Starting and overflow

- Timer control
  - Must set bit to start running
  - Timer sets a bit whenever it overflows
    - You must clear overflow bit to watch for next overflow

TCON	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Set to 1 when timer 1 overflows

1 = Timer 1 running
0 = Timer 1 stopped

Set to 1 when timer 0 overflows

1 = Timer 0 running
0 = Timer 0 stopped

Mode 1 example

```
    MOV TMOD, #00000001b ; Set timer0 mode to 01 = mode 1
    SETB TR0             ; Start timer0 running

Loop:
    ; Toggle all the LEDs every time P2.0 is pressed
    JB P2.0, $
    JNB P2.0, $
    CPL A
    MOV P0, A
    JMP Loop
```

Shorthand for code address of this line

TMOD	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	GATE1	C/T1	T1M1	T1M0	GATE0	C/T0	T0M1	T0M0

TCON	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Mode 1 overflow example

```
    MOV TMOD, #00000001b ; Set timer0 mode to 01 = mode 1
    SETB TR0             ; Start timer0 running

Loop:
    JNB TF0, $      ; Wait for overflow bit to be set
    INC P0          ; Increment P0 so LEDs change
    CLR TF0         ; Clear the overflow bit
    JMP Loop
```

TMOD	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	GATE1	C/T1	T1M1	T1M0	GATE0	C/T0	T0M1	T0M0

TCON	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Precise timing

- Goal: create 1 second pause using timer
  - Each timer increment = 1 cycle
  - 0.05 sec * 921,600 cycles/sec = 46080 cycles
  - Set timer to overflow this often
    - Start timer off part way through its 16-bit cycle
    - 65536 - 46080 = 19456
  - Repeat 20 times

Pause 1 second

```
Start:
    MOV TMOD, #00000001b   ; Put timer 0 in 16-bit mode

    MOV R0, #20            ; Loop, 0.05 * 20 = 1.0 sec
Loop:
    CLR TF0                ; Clear overflow bit
    MOV TH0, #76           ; High byte 19,456 = 76 * 256
    MOV TL0, #00           ; Low byte of 0

    SETB TR0               ; Start timer0 running
    JNB TF0, $             ; Wait for overflow
    CLR TR0                ; Stop timer0 running

    DJNZ R0, Loop
```

TMOD	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	GATE1	C/T1	T1M1	T1M0	GATE0	C/T0	T0M1	T0M0

TCON	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Reading 16-bit timer

- Problem: timer continues to increment while we read high and low byte
  - Timer value 3839 = 14 * 256 + 255
  - High byte 14, low byte 255
  - If we read high byte then low byte
    - MOV R0, TL0 (1 cycle)
    - MOV R1, TH0 (1 cycle)
  - Actual result:
    - High byte 15, low byte 255

Reading 16-bit timer

- Solution 1: stop the timer
  - Will lose a small amount of time

```
CLR TR0                 ; Stop timer0
MOV A, TH0              ; Copy high byte of timer
MOV R0, TL0             ; Copy low byte of timer
SETB TR0                ; Restart timer0
```

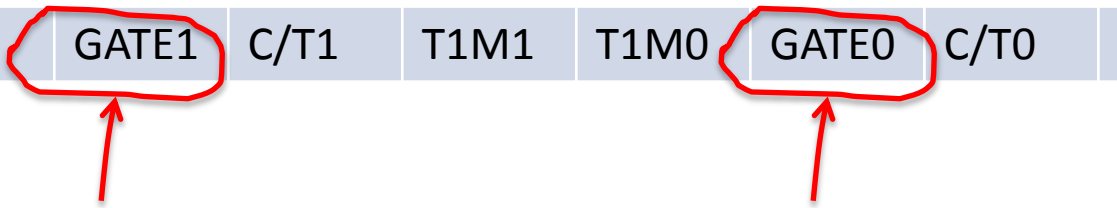
- Solution 2: read until no change in high byte

```
Repeat:
    MOV  A, TH0           ; Copy high byte of timer
    MOV  R0, TL0          ; Copy low byte of timer
    CJNE A, TH0, Repeat   ; Repeat until no change in high
```

Timing event length

- Increment only in presence of external signal
  - Hook something up to P3.2
  - Timer only increments if pin high
  - Microcontroller handles the check of P3.2
  - Set GATE0/1 bit in TMOD

TMOD	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	GATE1	C/T1	T1M1	T1M0	GATE0	C/T0	T0M1	T0M0



if GATE1 = 1, P3.2 must be high for timer1 to increment

if GATE0=1, P3.2 must be high for timer0 to increment

Counting events

- Use timer to count discrete events
  - Hook something up to P3.4
  - Timer increments if it detects "event"
  - Event = one-to-zero transition
    - 1/24 * 11.0592 Mhz = 460,800 events/second or less
  - Set C/Tx bit in TMOD

TMOD	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	GATE1	C/T1	T1M1	T1M0	GATE0	C/T0	T0M1	T0M0

if 1, counts in timer1
events occurring on P3.4

if 1, counts in timer0
events occurring on P3.4

Timer mode 2

- Mode 1, 8-bit auto-reload
  - Counts from 0 up to 255, in low byte: TLx
  - Resets to fixed value in high byte: THx
  - Useful for establishing baud rate in serial comm
  - Set mode 2 using two bits in TMOD

TMOD	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	GATE1	C/T1	T1M1	T1M0	GATE0	C/T0	T0M1	T0M0

1 0 = mode 1 for timer 1

1 0 = mode 1 for timer 0

Auto-reload example

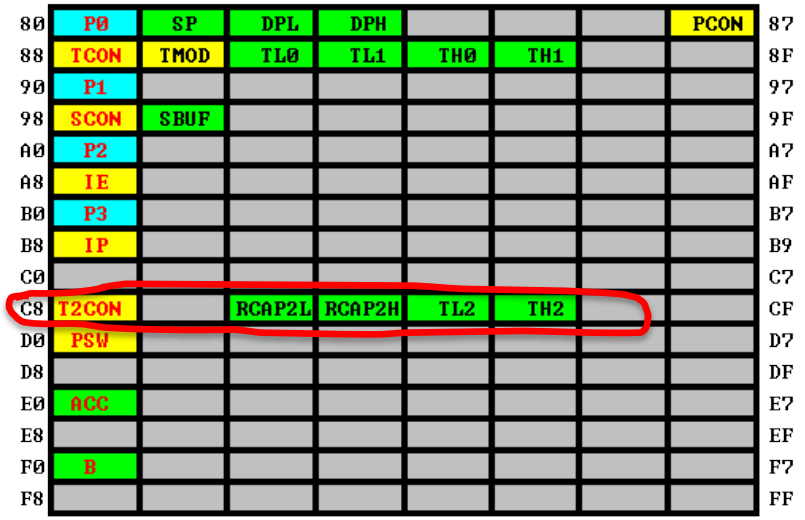
```
Start:
     MOV TMOD,#00100000b    ; Timer 1, 8-bit auto reload
     MOV TH1, #253          ; Set reload value 253 (9,600 baud)
     SETB TR1               ; Start timer1 running
Loop:
     MOV P0, TL1            ; Copy timer value to P0
     JMP Loop
```

TMOD	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	GATE1	C/T1	T1M1	T1M0	GATE0	C/T0	T0M1	T0M0

TCON	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Timer 2

- Timer 2
  - 8052 adds a third timer
    - 16-bit auto-reload mode
    - Capture mode
  - Controlled by T2CON SFR
  - Two extra SFRs for reload/capture values
    - Low byte RCAP2L, high byte RCAP2H



T2CON	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2	CP/RL2

Precise timing redux

- Goal: create 1 second pause using timer 2
  - Timer2 can automatically reset 16-bit start value
  - Note: Keil AT89S8253, missing timer2 SFRs:

```
; Declare all the timer2 related SFRs
T2CON  EQU 0C8h
RCAP2L EQU 0CAh
RCAP2H EQU 0CBh
TL2    EQU 0CCh
TH2    EQU 0CDh
TF2    EQU T2CON.7
EXF2   EQU T2CON.6
RCLK   EQU T2CON.5
TCLK   EQU T2CON.4
EXEN2  EQU T2CON.3
TR2    EQU T2CON.2
C_T2   EQU T2CON.1
CP_RL2 EQU T2CON.0
ET2    EQU IE.5
```

Timer 2, 1 second pause

```
Start:
    MOV T2CON,  #00000000b ; Put timer 2 in 16-bit reload
    MOV RCAP2L, #0         ; Reload low byte 0
    MOV RCAP2H, #76        ; Reload high byte 7
    MOV TL2, #0            ; Start timer at low byte 0
    MOV TH2, #76           ; Start timer at high byte 7
    SETB TR2               ; Start timer2

Pause1Sec:
    MOV R0, #20            ; Loop, 0.05 * 20 = 1.0 sec
Loop:
    JNB TF2, $             ; Wait for overflow
    CLR TF2                ; Clear overflow flag
    DJNZ R0, Loop

    JMP Pause1Sec          ; Loop forever
```

Timer 2, capture mode

- Capture mode
  - On external event, copy timer value to RCAP2H/L
  - Allows precise recording of event time
    - Triggered on 1-0 transition on P1.1
  - Timer continues to run
  - Set CP/RL2 = 1, EXEN2 = 1

T2CON	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2	CP/RL2

Intro to interrupts

- Interrupts
  - Microcontroller interrupts your normal program
  - Interrupt on:
    - Timer0/1/2 overflow/reload
    - Reception/transmission serial data
    - External event (P3.3, P3.2)
  - Go to fixed code address
    - Run some code and then return

Interrupt example

- Goal:
  - Monitor button, if pressed buzz
  - Also increment binary counter every second
    - Build main program to handling buzzing
    - Build Interrupt Service Routine (ISR) for counter
    - Use timer2 auto-reload
    - Interrupt every 0.05 seconds
    - Every 20 interrupts:
      - Increment counter variable
      - Update the LEDs

Buzzing main program

```
BUZZER EQU P3.0
BUTTON EQU P2.0

Main:
    SETB BUZZER             ; Turn buzzer off
    JB   BUTTON, $          ; Loop if button not pressed
    CLR  BUZZER             ; Turn buzzer on
    JNB  BUTTON, $          ; Wait for button release
    JMP  Main
```

Setting up timer & interrupt

```
Start:
    MOV T2CON,  #00000000b ; Put timer 2 in 16-bit reload
    MOV RCAP2L, #0         ; Reload low byte 0
    MOV RCAP2H, #76        ; Reload high byte 7
    MOV TL2,    #0         ; Start timer at low byte 0
    MOV TH2,    #76        ; Start timer at high byte 7

    SETB EA                ; Enable global interrupts
    SETB ET2               ; Enable timer2 overflow interrupt
    SETB TR2               ; Start timer2
```

Interrupt service routine

```
; Timer 2 overflow ISR
ORG 002Bh                       ; Address of ISR
     PUSH ACC                   ; Save off accumulator
     PUSH PSW                   ; Save off PSW

     INC R0                     ; Another 0.05 has passed
     CLR TF2                    ; Clear overflow flag
     CJNE R0, #20, Done         ; Leave if not to 20 yet
     MOV R0, #0                 ; Reset count to 0

     INC R1                     ; Update the P0 LED counter
     MOV A, R1
     CPL A
     MOV P0, A

Done:
     POP PSW                    ; Restore PSW
     POP ACC                    ; Restore accumulator
     RETI                       ; Return from ISR
```

Summary

- Timers
  - Provide precise time delays
  - Time events
  - Count events
- Interrupts
  - Allow concurrent processing
  - Based on interrupt signal
    - From timer
    - From receiving/transmitting data
    - From external signal