



University of British Columbia  
Electrical and Computer Engineering  
ELEC291/292

## Timers, Interrupts, and Pushbuttons

Dr. Jesús Calviño-Fraga P.Eng.  
Department of Electrical and Computer Engineering, UBC  
Office: KAIS 3024  
E-mail: [jesusc@ece.ubc.ca](mailto:jesusc@ece.ubc.ca)  
Phone: (604)-827-5387

January 19, 2024

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

## Objectives

- Configure and use the timers in the N76E003 microcontroller.
- Configure and use interrupts.
- Attach (and use) pushbuttons.
- Attach and use speaker with the N76E003 microcontroller.

Timers, Interrupts, and Pushbuttons

2

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

## Timing & machine cycles

- For the N76E003, one machine cycle takes 1 oscillator period. This year the oscillator is set to 16.6 MHz: One cycle takes 60.24 ns.
- If we use delay loops for timing, the processor is busy wasting valuable computing time!
- A better solution is to use dedicated hardware for timing and counting: Timers and Counters!
- The timers and counters of other processors maybe/are different. The idea is the same!

Timers, Interrupts, and Pushbuttons

3

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

## Timers/Counters

- Timers/Counters have advantages over timing loops:
  - The processor is not tied while counting.
  - Combined with interrupts, produces very efficient (small and fast) code.
  - They are usually independent on how many clocks per cycle the CPU takes.
  - Many timers/counters can be set to work concurrently.

Timers, Interrupts, and Pushbuttons

4

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

## 8051's Timers/Counters

- The original 8051 has only two timers/counters: 0 and 1.
- Newer 8051 microcontrollers usually have:
  1. The 8051 timers/counters: timers 0 and 1
  2. The 8052 timer/counter: timer 2 (slightly different to the standard in the N76E003)
  3. The Programmable Counter Array (PCA). Kind of available in the N76E003 via timer 2! Very powerful, other architectures have very similar devices
  4. Additional timer/counters: time 3, 4, 5, 6, etc. Timer 3 is available in the N76E003.
- Let us begin with timers 0 and 1:

The more timers the merrier!

Timers, Interrupts, and Pushbuttons

5

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

## Timer 0 and Timer 1 Operation Modes

- Timer 0 and 1 have four modes of operation:
  - Mode 0: 13-bit timer/counter (compatible with the 8048 microcontroller, the predecessor of the 8051). DO NOT USE THIS MODE!
  - Mode 1: 16-bit timer/counter.
  - Mode 2: 8-bit auto reload timer counter.
  - Mode 3: Special mode 8-bit timer/counter (timer 0 only).
- Timer 1 can be used as baud rate generator for the serial port. Some 8051/8052 microcontrollers have a dedicated baud rate generator.

Timers, Interrupts, and Pushbuttons

6

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

## TMOD timer/counter mode control register (Address 89H)

Timer 1				Timer 0			
GATE	C/T*	M1	M0	GATE	C/T*	M1	M0

Bit	Name		Description
7 & 3	GATE		1: uses either INT0 or INT1 pins to enable/disable the timer/counter
6 & 2	C/T*		0: timer; 1: counter (pins T0 and T1)
All the other pins!	M1	M0	
	0	0	13-bit timer/counter
	0	1	16-bit timer/counter
	1	0	8-bit auto-reload timer/counter
	1	1	Special mode

Timers, Interrupts, and Pushbuttons

7

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

## TCON: timer/counter control register. (Address 88H)

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
-----	-----	-----	-----	-----	-----	-----	-----

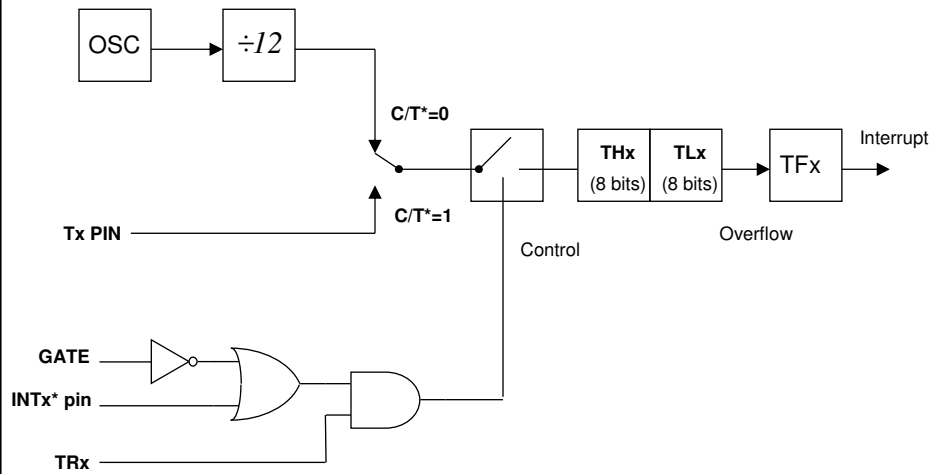
Bit	Name	Description
7	TF1	Timer 1 overflow flag.
6	TR1	Timer 1 run control.
5	TF0	Timer 0 overflow flag.
4	TR0	Timer 0 run control.
3	IE1	Interrupt 1 flag.
2	IT1	Interrupt 1 type control bit.
1	IE0	Interrupt 0 flag.
0	IT0	Interrupt 0 type control bit.

Timers, Interrupts, and Pushbuttons

8

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

## Timer/Counter 0 or 1 in Mode 1 Original 8051



Timers, Interrupts, and Pushbuttons

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

9

## N76E003 Timer/Counter 0/1.

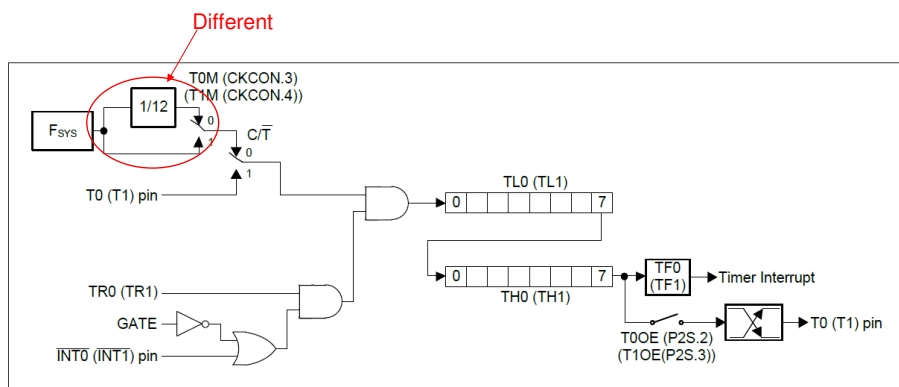


Figure 8.2-1. Timer/Counters 0 and 1 in Mode 1

Timers, Interrupts, and Pushbuttons

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

10

# CKCON Register

CKCON – Clock Control

7	6	5	4	3	2	1	0
-	PWMCKS	-	T1M	T0M	-	CLOEN	-
-	R/W	-	R/W	R/W	-	R/W	-

Address: 8EH

Reset value: 0000 0000b

Bit	Name	Description
6	PWMCKS	<b>PWM clock source select</b> 0 = The clock source of PWM is the system clock $F_{SYS}$ . 1 = The clock source of PWM is the overflow of Timer 1.
4	T1M	<b>Timer 1 clock mode select</b> 0 = The clock source of Timer 1 is the system clock divided by 12. It maintains standard 8051 compatibility. 1 = The clock source of Timer 1 is direct the system clock.
3	T0M	<b>Timer 0 clock mode select</b> 0 = The clock source of Timer 0 is the system clock divided by 12. It maintains standard 8051 compatibility. 1 = The clock source of Timer 0 is direct the system clock.
1	CLOEN	<b>System clock output enable</b> 0 = System clock output Disabled. 1 = System clock output Enabled from CLO pin (P1.1).

Timers, Interrupts, and Pushbuttons

11

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

## Timer/Counter 2 block diagram for the N76E003

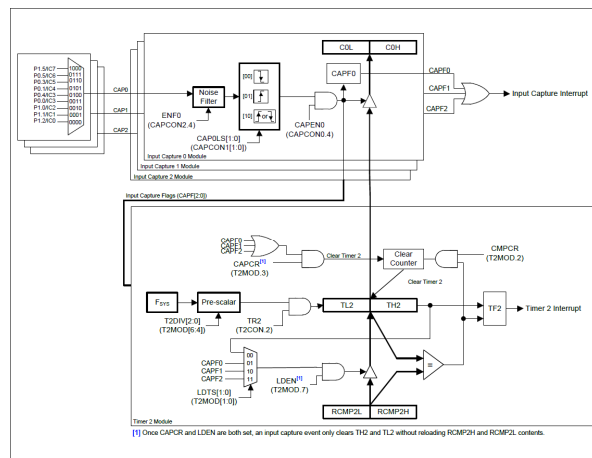


Figure 8.4-1. Timer 2 Block Diagram

Timers, Interrupts, and Pushbuttons

12

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

## T2CON: timer/counter 2 control register. (Address C8H)

T2CON – Timer 2 Control

7	6	5	4	3	2	1	0
TF2	-	-	-	-	TR2	-	CM/RL2
R/W	-	-	-	-	R/W	-	R/W

Address: C8H

Reset value: 0000 0000b

Bit	Name	Description
7	TF2	<b>Timer 2 overflow flag</b> This bit is set when Timer 2 overflows or a compare match occurs. If the Timer 2 interrupt and the global interrupt are enable, setting this bit will make CPU execute Timer 2 interrupt service routine. This bit is not automatically cleared via hardware and should be cleared via software.
2	TR2	<b>Timer 2 run control</b> 0 = Timer 2 Disabled. Clearing this bit will halt Timer 2 and the current count will be preserved in TH2 and TL2. 1 = Timer 2 Enabled.
0	CM/RL2	<b>Timer 2 compare or auto-reload mode select</b> This bit selects Timer 2 functioning mode. 0 = Auto-reload mode. 1 = Compare mode.

Timers, Interrupts, and Pushbuttons

13

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

## T2CON: timer/counter 2 control register. (Address C8H)

T2MOD – Timer 2 Mode

7	6	5	4	3	2	1	0
LDEN	T2DIV[2:0]			CAPCR	CMPCR	LDS[1:0]	
R/W	R/W			R/W	R/W	R/W	

Address: C9H

Reset value: 0000 0000b

Bit	Name	Description
7	LDEN	<b>Enable auto-reload</b> 0 = Reloading RCMP2H and RCMP2L to TH2 and TL2 Disabled. 1 = Reloading RCMP2H and RCMP2L to TH2 and TL2 Enabled.
6:4	T2DIV[2:0]	<b>Timer 2 clock divider</b> 000 = Timer 2 clock divider is 1/1. 001 = Timer 2 clock divider is 1/4. 010 = Timer 2 clock divider is 1/16. 011 = Timer 2 clock divider is 1/32. 100 = Timer 2 clock divider is 1/64. 101 = Timer 2 clock divider is 1/128. 110 = Timer 2 clock divider is 1/256. 111 = Timer 2 clock divider is 1/512.
3	CAPCR	<b>Capture auto-clear</b> This bit is valid only under Timer 2 auto-reload mode. It enables hardware auto-clearing TH2 and TL2 counter registers after they have been transferred in to RCMP2H and RCMP2L while a capture event occurs. 0 = Timer 2 continues counting when a capture event occurs. 1 = Timer 2 value is auto-cleared as 0000H when a capture event occurs.
2	CMPCR	<b>Compare match auto-clear</b> This bit is valid only under Timer 2 compare mode. It enables hardware auto-clearing TH2 and TL2 counter registers after a compare match occurs. 0 = Timer 2 continues counting when a compare match occurs. 1 = Timer 2 value is auto-cleared as 0000H when a compare match occurs.

Timers, Interrupts, and Pushbuttons

14

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

## T2CON: timer/counter 2 control register. (Address C8H) Continuation...

Bit	Name	Description
1:0	LDTS[1:0]	<b>Auto-reload trigger select</b> These bits select the reload trigger event. 00 = Reload when Timer 2 overflows. 01 = Reload when input capture 0 event occurs. 10 = Reload when input capture 1 event occurs. 11 = Reload when input capture 2 event occurs.

Timer 2 is mostly compatible with the standard 8051 timer 2. Look at the subroutine 'Timer2\_Init' in 'ISR\_example.asm'.

Timers, Interrupts, and Pushbuttons

15

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

## Example: Time Delay Using a Timer

- To use a timer to implement a delay we need to:
  - Initialize the timer: use TMOD SFR.
  - Load the timer: use THx and TLx.
  - Clear the timer overflow flag: TFX=0;
  - Start the timer: Use TRx.
  - Check the timer overflow flag: Use TFX.

For the registers above 'x' is either '0' for timer 0, or '1' for timer 1.

Timers, Interrupts, and Pushbuttons

16

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.



## Time Delay Using a Timer

- Implement a 1 ms delay subroutine using timer 0. Assume the routine will be running in a N76E003 microcontroller with a 16.6 MHz oscillator.

First, we have to find the divider (TH0, TL0) needed for a 1 ms delay...

## N76E003 Timer/Counter 0/1 in Mode 1

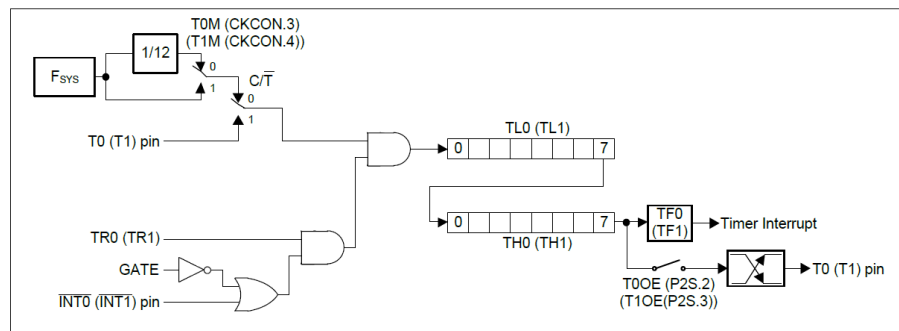


Figure 8.2-1. Timer/Counters 0 and 1 in Mode 1

## Calculating TH0 and TL0

$$\text{Rate} = \frac{\text{CLK}}{2^{16} - [\text{THn}, \text{TLn}]} = \frac{16.6\text{MHz}}{65536 - [\text{THn}, \text{TLn}]}$$

$$[\text{THn}, \text{TLn}] = 65536 - \frac{16.6\text{MHz}}{\text{Rate}} = 65536 - \frac{16.6\text{MHz}}{(1/1\text{ms})} = 48936$$

Maximum delay achievable?

$$\text{Rate} = \frac{16.6\text{MHz}}{2^{16} - [\text{THn}, \text{TLn}]} = \frac{16.6\text{MHz}}{65536 - [\text{THn}, \text{TLn}]}$$

$$[\text{THn}, \text{TLn}] = 0$$

$$\text{Rate} = \frac{16.6\text{MHz}}{65536} = 253.3\text{Hz} \rightarrow 3.95\text{ms}$$

Timers, Interrupts, and Pushbuttons

19

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

## Time Delay Using Timer 0

```
Wait1ms:
; Initialize the timer
mov a, TMOD
anl a, #11110000B ; Clear bits for timer 0
orl a, #00000001B ; GATE=0, C/T*=0, M1=0, M0=1: 16-bit timer
mov TMOD, a
clr TR0 ; Disable timer 0
; Load the timer [TH0, TL0]=65536-(16600000/(1/0.001))
mov TH0, #high(48936)
mov TL0, #low(48936)
clr TF0 ; Clear the timer flag
setb TR0 ; Enable timer 0
Wait1ms_L0:
jnb TF0, Wait1ms_L0 ; Wait for overflow
ret
```

Not bad, but we can do better:

Timers, Interrupts, and Pushbuttons

20

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

## Time Delay Using Timer 0

```
; Let the Assembler do the calculation for us!
XTAL equ 16600000
FREQ equ 1000 ; 1/1000Hz=1ms
RELOAD_TIMER0_1ms equ 65536-(XTAL/FREQ)

Wait1ms:
    ; Initialize the timer
    mov a, TMOD
    anl a, #11110000B ; Clear bits for timer 0
    orl a, #00000001B ; GATE=0, C/T*=0, M1=0, M0=1: 16-bit timer
    mov TMOD, a
    clr TR0 ; Disable timer 0
    mov TH0, #high(RELOAD_TIMER0_1ms)
    mov TL0, #low(RELOAD_TIMER0_1ms)
    clr TF0 ; Clear the timer flag
    setb TR0 ; Enable timer 0
Wait1ms_L0:
    jnb TF0, Wait1ms_L0 ; Wait for overflow
    ret
```

Timers, Interrupts, and Pushbuttons

21

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

## Interrupts

- Interrupt uses:
  - Handshake I/O thus preventing CPU from being tied up.
  - Providing a way to handle some errors: illegal opcodes, dividing by 0, power failure, etc.
  - Getting the CPU to perform periodic tasks: generate square waves, keep time of day, measure frequency, etc.

Timers, Interrupts, and Pushbuttons

22

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

# Interrupts

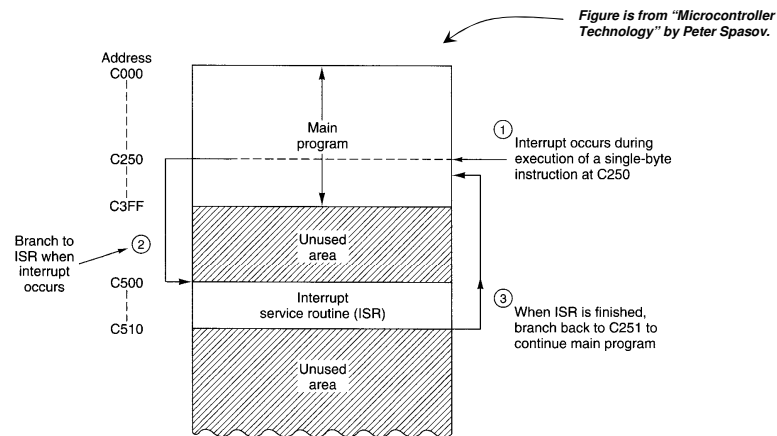


FIGURE 8.11 Example of how an interrupt causes a change in the execution of a program.

Timers, Interrupts, and Pushbuttons

23

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

# Interrupts

- Most processors provide a way of enabling / disabling all maskable interrupts. For the 8051:

**clr EA** ;Disable interrupts

**setb EA** ;Enable interrupts

- Some other interrupts are non-maskable and MUST be serviced. For example, the X86 has the "Non-Maskable Interrupt" NMI.
- Maskable interrupts can be enabled/disabled individually. For the 8051 use register IE:

Timers, Interrupts, and Pushbuttons

24

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

## IE: INTERRUPT ENABLE REGISTER. (Address A8H) (Original 8051)

EA	--	ET2	ES	ET1	EX1	ET0	EX0
----	----	-----	----	-----	-----	-----	-----

Bit	Name	Description
7	EA	Interrupt Enable Bit: EA = 1 interrupt(s) can be serviced, EA = 0 interrupt servicing disabled.
6	--	Reserved
5	ET2	Timer 2 Interrupt Enable. (8052)
4	ES	Serial Port Interrupt Enable
3	ET1	Timer 1 Overflow Interrupt Enable.
2	EX1	External Interrupt 1 Enable.
1	ET0	Timer 0 Overflow Interrupt Enable.
0	EX0	External Interrupt 0 Enable.

Timers, Interrupts, and Pushbuttons

25

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

## () Bit addressable registers

- If the location address of an special function register (SFR) is a multiple of 8, then the register is bit addressable and you can use the **setb** and **clr** instructions.
- IE is bit addressable! Then you can access the bits like “setb EA”.

Timers, Interrupts, and Pushbuttons

26

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

# Interrupts in the N76E003

IE – Interrupt Enable (Bit-addressable)

7	6	5	4	3	2	1	0
EA	EADC	EBOD	ES	ET1	EX1	ET0	EX0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Address: A8H

Reset value: 0000 0000b

Bit	Name	Description
7	EA	<b>Enable all interrupt</b> This bit globally enables/disables all interrupts that are individually enabled. 0 = All interrupt sources Disabled. 1 = Each interrupt Enabled depending on its individual mask setting. Individual interrupts will occur if enabled.
6	EADC	<b>Enable ADC interrupt</b> 0 = ADC interrupt Disabled. 1 = Interrupt generated by ADCF (ADCCON0.7) Enabled.
5	EBOD	<b>Enable brown-out interrupt</b> 0 = Brown-out detection interrupt Disabled. 1 = Interrupt generated by BOF (BODCON0.3) Enabled.
4	ES	<b>Enable serial port 0 interrupt</b> 0 = Serial port 0 interrupt Disabled. 1 = Interrupt generated by TI1 (SCON.1) or RI (SCON.0) Enabled.
3	ET1	<b>Enable Timer 1 interrupt</b> 0 = Timer 1 interrupt Disabled. 1 = Interrupt generated by TF1 (TCON.7) Enabled.
2	EX1	<b>Enable external interrupt 1</b> 0 = External interrupt 1 Disabled. 1 = Interrupt generated by INT1 pin (P1.7) Enabled.
1	ET0	<b>Enable Timer 0 interrupt</b> 0 = Timer 0 interrupt Disabled. 1 = Interrupt generated by TF0 (TCON.5) Enabled.
0	EX0	<b>Enable external interrupt 0</b> 0 = External interrupt 0 Disabled. 1 = Interrupt generated by INT0 pin (P3.0) Enabled.

Timers, Interrupts, and Pushbuttons

27

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

# Interrupts in the N76E003

EIE – Extensive Interrupt Enable

7	6	5	4	3	2	1	0
ET2	ESPI	EFB	EWDT	EPWM	ECAP	EPI	EI2C
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Address: 9BH

Reset value: 0000 0000b

Bit	Name	Description
7	ET2	<b>Enable Timer 2 interrupt</b> 0 = Timer 2 interrupt Disabled. 1 = Interrupt generated by TF2 (T2CON.7) Enabled.
6	ESPI	<b>Enable SPI interrupt</b> 0 = SPI interrupt Disabled. 1 = Interrupt generated by SPIF (SPSR.7), SPIOVF (SPSR.5), or MODF (SPSR.4) Enabled.
5	EFB	<b>Enable Fault Brake interrupt</b> 0 = Fault Brake interrupt Disabled. 1 = Interrupt generated by FBF (FBD.7) Enabled.
4	EWDT	<b>Enable WDT interrupt</b> 0 = WDT interrupt Disabled. 1 = Interrupt generated by WDTF (WDCON.5) Enabled.
3	EPWM	<b>Enable PWM interrupt</b> 0 = PWM interrupt Disabled. 1 = Interrupt generated by PWMF (PWMCON0.5) Enabled.
2	ECAP	<b>Enable input capture interrupt</b> 0 = Input capture interrupt Disabled. 1 = Interrupt generated by any flags of CAPF[2:0] (CAPCON0[2:0]) Enabled.
1	EPI	<b>Enable pin interrupt</b> 0 = Pin interrupt Disabled. 1 = Interrupt generated by any flags in PIF register Enabled.
0	EI2C	<b>Enable I<sup>2</sup>C interrupt</b> 0 = I <sup>2</sup> C interrupt Disabled. 1 = Interrupt generated by SI (I2CON.3) or I2TOF (I2TOC.0) Enabled.

Timers, Interrupts, and Pushbuttons

28

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

# Interrupts in the N76E003

EIE1 – Extensive Interrupt Enable 1

7	6	5	4	3	2	1	0
-	-	-	-	-	EWKT	ET3	ES_1
-	-	-	-	-	R/W	R/W	R/W

Address: 9CH

Reset value: 0000 0000b

Bit	Name	Description
2	EWKT	<b>Enable WKT interrupt</b> 0 = WKT interrupt Disabled. 1 = Interrupt generated by WKTF (WKCON.4) Enabled.
1	ET3	<b>Enable Timer 3 interrupt</b> 0 = Timer 3 interrupt Disabled. 1 = Interrupt generated by TF3 (T3CON.4) Enabled.
0	ES_1	<b>Enable serial port 1 interrupt</b> 0 = Serial port 1 interrupt Disabled. 1 = Interrupt generated by TI_1 (SCON_1.1) or RI_1 (SCON_1.0) Enabled.

Shouldn't it be 'Extended Interrupt Enable 1'?

Timers, Interrupts, and Pushbuttons

29

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

## Two external Interrupts

- Connected to pins P3.2 (INT0) and P3.3 (INT1) in the standard 8051. In the N76E003 they are assigned to P1.7 (INT1) and P3.0 (INT0).
- Can be configured to be edge sensitive or level sensitive. Use bits IT0 and IT1 in SFR TCON to specify falling edge or low level sensitivity.
- There is an application note (somewhere) on how to use the timer inputs T0 and T1 as additional external interrupts.
- The N76E003 has the “pin interrupt” that allows for selected pins to generate an interrupt.

Timers, Interrupts, and Pushbuttons

30

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

## Interrupts and the stack

- Interrupts in the 8051 make use of the stack.
- The stack is an area of memory where variables can be stacked. It is a LIFO memory: the last variable you put in is the first variable that comes out.
- Register SP (stack pointer) points to the beginning of the stack. SP in the 8051 is incremented **before** is used (**push**), or used and then decremented (**pop**).
- After reset, SP is set to 07H. If you have variables in internal RAM, any usage of the stack is likely to **OVERWRITE/CORRUPT** them. Therefore, at the beginning of your program set the SP:

**mov** SP, #7FH ; Set the stack pointer to idata start

Timers, Interrupts, and Pushbuttons

31

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

## Interrupts and the stack

- Additionally, these two instructions can be used to push/pull registers to/from the stack: **push & pop**
- After an interrupt is asserted the CPU:
  - Pushes the address of the next instruction into the stack (two bytes). Some processors also push some or all of the registers into the stack as well (not the 8051 though!).
  - All interrupts of equal or lower priority are disabled.
  - Then the program counter (PC) is set to the Interrupt Service Routine (ISR) vector.
  - The PC will be restored to the interrupted point once the **reti** instruction is executed in the ISR and all interrupts of equal or lower priority are re-enabled.

Timers, Interrupts, and Pushbuttons

32

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.



## Interrupt Service Routines (ISR) Vectors

- The 8051 will **lcall** to an specific memory location when an interrupt occurs. The may be different for different 8051 variants. For the standard 8051:

Interrupt source	Address
External 0	0003H
Timer 0	000BH
External 1	0013H
Timer 1	001BH
Serial port	0023H
Timer 2	002BH

- For the modern N76E003 there are many more interrupt vectors!  
See next slide:

Timers, Interrupts, and Pushbuttons

33

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

## Interrupts in the N76E003

Table 20-1. Interrupt Vectors

Source	Vector Address	Vector Number	Source	Vector Address	Vector Number
Reset	0000H	-	SPI interrupt	004BH	9
External interrupt 0	0003H	0	WDT interrupt	0053H	10
Timer 0 overflow	000BH	1	ADC interrupt	005BH	11
External interrupt 1	0013H	2	Input capture interrupt	0063H	12
Timer 1 overflow	001BH	3	PWM interrupt	006BH	13
Serial port 0 interrupt	0023H	4	Fault Brake interrupt	0073H	14
Timer 2 event	002BH	5	Serial port 1 interrupt	007BH	15
I <sup>2</sup> C status/timer-out interrupt	0033H	6	Timer 3 overflow	0083H	16
Pin interrupt	003BH	7	Self Wake-up Timer interrupt	008BH	17
Brown-out detection interrupt	0043H	8			

Timers, Interrupts, and Pushbuttons

34

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

## Interrupt Service Routines (ISR) Vectors

- Notice that there are only 8 bytes available between vectors. Not enough for a decent ISR, but more than enough for a *jmp* instruction!
- IF you enable a particular interrupt, there **MUST** be an ISR, or your program **WILL** crash. A fool proof code technique is to setup all the ISR vectors and place a *reti* (return from interrupt) instruction for those that are not used (next example).
- In assembly language you can use the “*org*” directive to set an ISR vector.
- To return from an ISR use the *reti* instruction. To return from a normal routine use the *ret* instruction.

Timers, Interrupts, and Pushbuttons

35

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

## Saving and Restoring Registers in the Stack

- If your ISR routine uses a register, you must make sure that it will remain **unmodified** before returning to the interrupted program. Example, if register “A” was 33 when the ISR was called, it must be set back to 33 before the *reti*.
- Use the instructions *push/pop* to save/restore registers to/from the stack.
- Additionally, you could use one of four available register banks in your ISR.

Timers, Interrupts, and Pushbuttons

36


Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

## Example: Saving and Restoring Registers in the Stack

```
timer0_ISR:
    ; The main loop is using both registers R0 and R1,
    ; so if we want to use them in this ISR we should push them
    ; into the stack and restore them before reti.
    push AR0
    push AR1

    .
    . ; Some code that uses R1 and R0
    .
    .

    ; Restore the register to their original values
    pop AR1
    pop AR0
    reti ; Return from interrupt
```



The 'A' stands for address...

Timers, Interrupts, and Pushbuttons

37

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

## Saving and Restoring Registers in the Stack

- Before using the stack make sure you set the SP register.
- Popular registers to push/pop in ISRs: ACC, DPL, DPH, PSW, R0 to R7. Of course, only if they are used in the ISR.
- Pop registers from the stack in the **REVERSE** order you pushed them! Remember the stack is a LIFO.

Timers, Interrupts, and Pushbuttons

38

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

## Setting the SP register

```
CLK            EQU 16600000 ; Oscillator frequency
TIMER0_RATE    EQU 4096     ; 2048Hz square wave
TIMER0_RELOAD  EQU ((65536-(CLK/TIMER0_RATE)))
myprogram:
    mov SP, 0x7f ; Do it once in your program!
    .
    .
    .
    .
```

Timers, Interrupts, and Pushbuttons

39

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

## Interrupt Programming with the 8051 in Assembly (summary)

- Set a ***ljmp*** to the ISR into the corresponding memory address for each interrupt source.
- Setup the stack in the main program. (Do this only once!)
- Setup (including priority) and Enable the interrupt to use.
- In the ISR use ***push/pop*** to save/restore used registers. You may also use a different register bank.
- Use a ***reti*** instruction to return from the ISR.

Timers, Interrupts, and Pushbuttons

40

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

## Reading Push Buttons

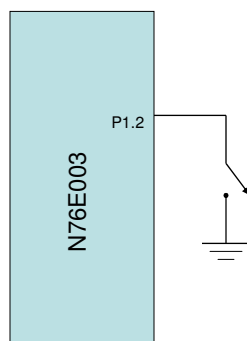
- Before using a pin for input we need to configure it:
  - Original 8051: Write '1' to the pin to be used as input.
  - Newer 8051s: configure the pin as input using designated SFRs .
- In the original 8051 any pin can be used as output or input. In newer 8051s some pins can be only input and/or outputs.
- In the original 8051 pins in the same port can be independently used as inputs or outputs. For example pin P0.0 can be used as input, while P0.1 can be used as output!

Timers, Interrupts, and Pushbuttons

41

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

## Reading Push Buttons



```
Setb P1.2; Make pin input
```

```
.  
. .  
. .  
. .
```

```
jnb P1.2, ButtonPressed
```

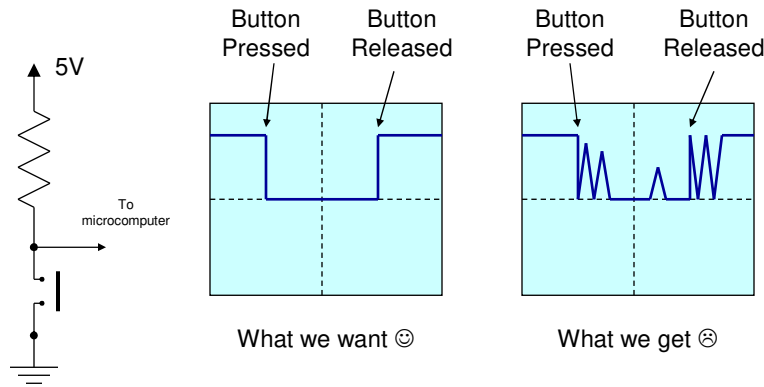
```
jb P1.2, ButtonNotPressed
```

Timers, Interrupts, and Pushbuttons

42

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

## Problem: Contact Bounce



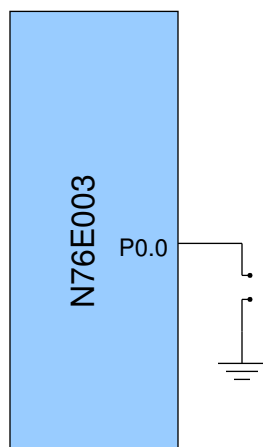
The time the contact bounces  
can be as long as 50ms!

Timers, Interrupts, and Pushbuttons

43

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

## Software Debouncing



```

mov P0M1, #0 ; Configure P0
mov P0M2, #0
setb P0.0 ; Before using as input...
jb P0.0, not_pressed
lcall Wait50ms ; Wait and check again
jb P0.0, not_pressed
; Wait for the button to be released
L0: jnb P0.0, L0
sjmp pressed
    
```

This technique is called *wait-and-see*  
in many textbooks

Timers, Interrupts, and Pushbuttons

44

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

# Speaker



## Specifications

Rated voltage	3.5 V <sub>o-p</sub>	
Operating voltage	3.0 - 5.0 V <sub>o-p</sub>	
Mean current	35 mA max.	
Coil resistance	42 ±6.3 Ω	Applying rated voltage, 2048 Hz square wave, 1/2 duty
Sound output	Min. 85 (Typical 95) dBA	Distance at 10cm (A-weight free air). Applying rated voltage of 2048 Hz, square wave, 1/2 duty.
Rated frequency	2,048 Hz	
Operating temperature	-20 ~ +60° C	
Storage temperature	-30 ~ +70° C	
Dimensions	φ12.0 x H6.5 mm	See attached drawing
Weight	1.4 g	
Material	PPO (Black)	
Terminal	Pin type (AU Plating)	See attached drawing
RoHS	yes	

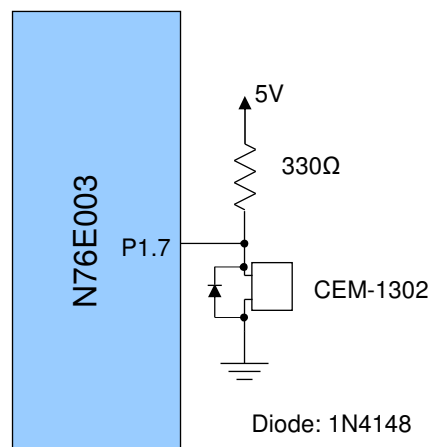
Louder at 2.048 kHz!

Timers, Interrupts, and Pushbuttons

45

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

# Attaching Speaker (simple way)

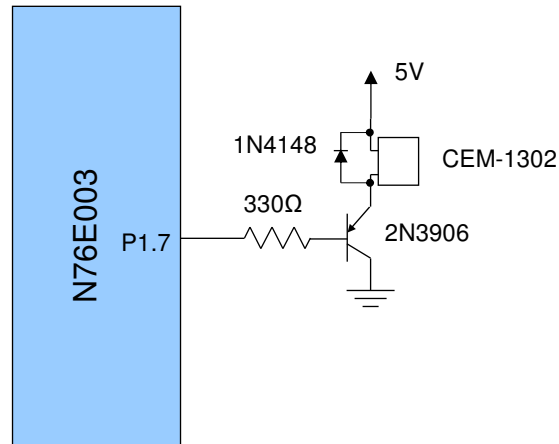


Timers, Interrupts, and Pushbuttons

46

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

## Attaching Speaker (louder!)

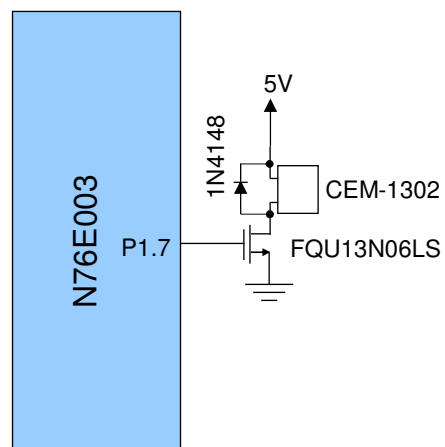


Timers, Interrupts, and Pushbuttons

47

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

## Attaching Speaker (loud and simple!)



Timers, Interrupts, and Pushbuttons

48

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.



## Lab 2

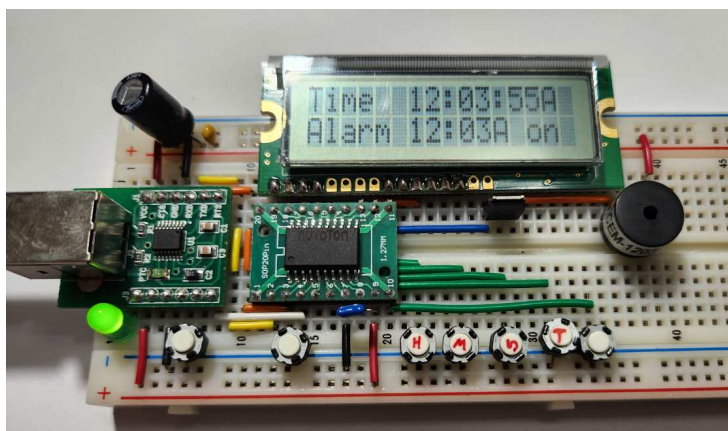
- 12h Alarm clock (AM/PM).
- Use LCD, speaker, and push buttons.
- Sample code provided:
  - ISR\_example.asm: interrupt programming example.
  - LCD\_4bit.inc: functions and macros to use the LCD.

Timers, Interrupts, and Pushbuttons

49

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.

## Lab 2



Timers, Interrupts, and Pushbuttons

50

Copyright © 2009-2024, Jesus Calvino-Fraga. Not to be copied, used, or revised without explicit written permission from the copyright owner.