classmate
Date
Page

-07-'19 # UNIT-I: EMBEDDED COMPUTING

- Introduction
- Complex Systems and Microprocessors
-

Characteristics we need to tackle care of deadlines, improve performance
- → Complex Algorithms
- → Uses Interfaces (GPS), Complex Interface
- → Real Time (deadline)
- → Multivariate
- → Manufacturing Cost
- → Power and Energy

-

Challenges
- → How much h/w do we need?
- → How do we meet deadlines?
- → How do we minimize power consumption?
- → How do we design for upgradability?
- → Complex Testing?
- → Limited observability and controlability?
- → Restricted development environment?

-

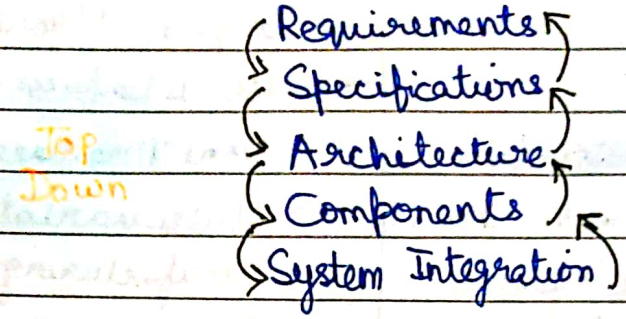
Performance
- → CPU
- → Platform
- → Program
- → Task
- → Multiprocessor

- The Embedded System Design Process

Normal Purpose - General Purpose

Embedded → which includes a microprocessor but not used for general purpose, i.e., specific purpose.

ES Design Process



- We will do all the programs using microcontrollers.
- A microcontroller is application specific: General Purpose, Only for one purpose, Only one operation
- A microprocessor is general purpose whereas a microcontroller is in built, one system contains all componenets.
- First Microprocessor - Intel 4004
  Calculator Operation - Calculator Application
- In a microprocessor, all connections are external.
- When we are upgrading something, we need to think about cost vs performance.
- We need to generate embedded system in such a way that programming should take less power and energy.
- Why do we use microprocessor for Embedded System?
  This is because any app can be run or implemented on microprocessor.
- Here: Components = Component Design

Challenges:

i) ES comes with hardware
- Optimize the hardware, in case also, we use ES (embedded system), many vehicles, application specific.

ii) Real Time
- Cost vs Performance

iii) Eg: Battery in Mobile Phones
Before it was very large, now it is small and large utilization

★ iv) ES installed software - we are using it with little changes. We need to make upgradability. It should execute a new program when it is upgraded.

v) vi) vii) } Same Concept — Write program on common desktop, compile and execute

How to improve performance of ES?
- It depends on processor (CPU)
- Depends on wires, buses, (platform), connection media
- Layer program into smaller one (program)
- Multitasking (single processor doing many tasks)
- Multiprocessor (it contains many processors)

07-19

In ES, Hexcode is not written manually.
- We will be writing C and C++ programs
- Hexfile will go to ES. Before dumping, we need to check all the possibilities when doing in ES (it is also one of the challenges).
- Testing on desktop.
- We cannot change it in ES, so before dumping, we ~~only~~ need to check. It is one of the disadvantages.
- In ES, we can read hardly 30 characters, but in other processors, it can read all characters, i.e., 1550.

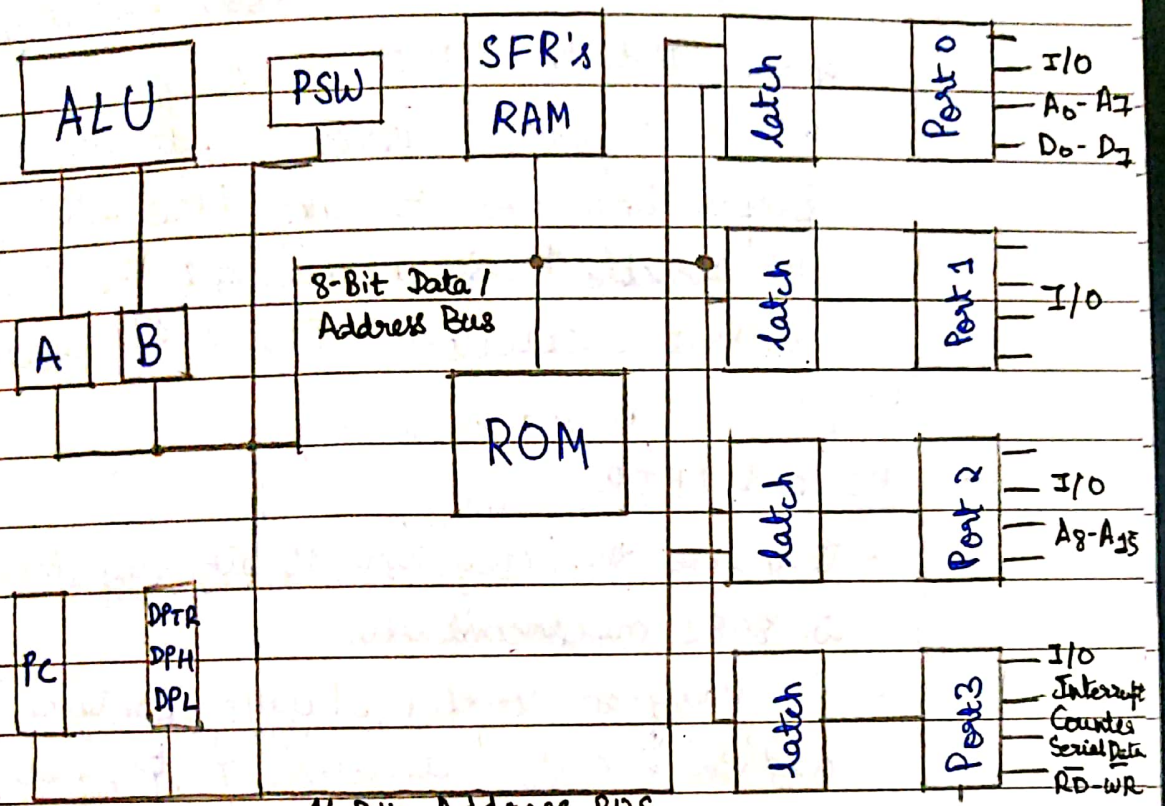
ES Design Process (5 mks, explain with applica

i) It is a 5 step process
It basically uses top-down approach.
I/P, O/P
Cost
Performance of System
ii) Specifications
iii) Architecture - what it does
Organization - How it does
(i/P, O/P, how many different hardware, how many sensors)
iv) Each and every component
(which component for what purpose)
v) Combining is System Integration

EA
ALE
PSEN
XTAL1
XTAL2
RESET
Vcc
GND

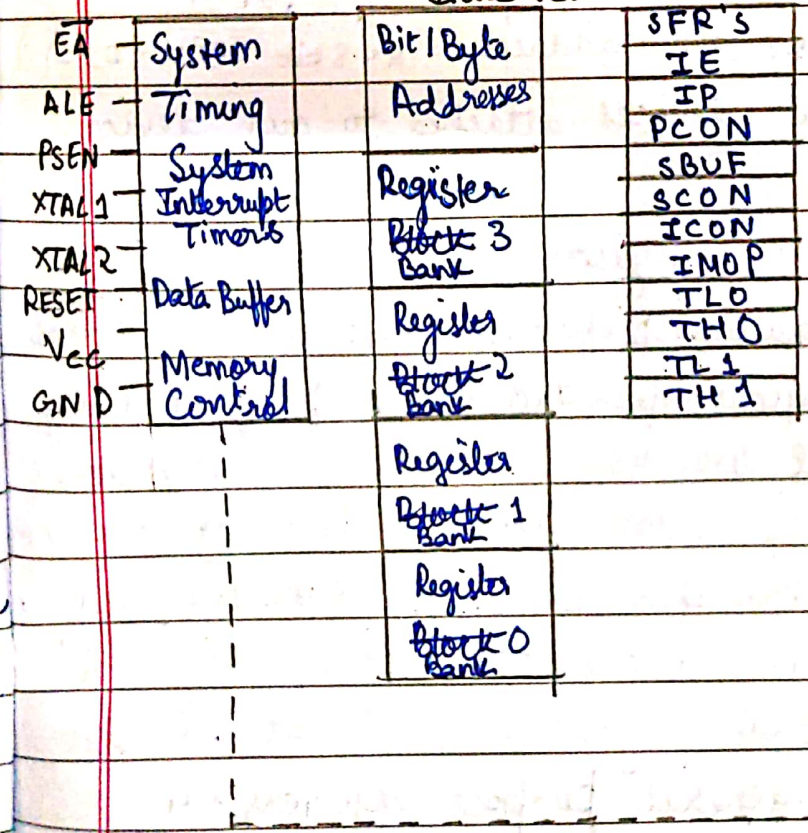
07-19

8051 Architecture



16 Bit Address BUS

Internal RAM Structure



→ **8051 Oscillator, Clock**

- 8051 consists of 2 internal oscillators and 3 external oscillators, whose frequency ranges from 1MHz to 16MHz.
- We fix crystal frequency to 12MHz, to use proper time; i.e., to make time 1µs. | 16MHz → < 1µs, <12mHz → > 1µs
- Not possible to synchronize all the tasks without oscillator. | Time α 1/frequency

→ **PC and DPTR**

- They are the only two 16 bit registers available in 8051 microcontroller.
- PC: Program counter always contains 16-bit address of next instruction to be executed.
- DPTR: Data Pointer is used to access external addresses (16bit address = 2× 8bit (DPH + DPL)) of external devices attached to our device.

→ **A and B CPU registers**

- All operations performed such as arithmetic and logical operations are performed by default we use 'A' and 'B' registers and also store data in them. They are CPU Registers
- Data transfer between external and internal devices are done using 'A' register
- A = 8 bit register. B = 8-bit register

★ 8 bit general purpose registers × 4
★ 2×16 bit registers : PC, DPTR

→ **Flags and PSW**

- Seven flags are present.
- Four special purpose flags, modified when arithemetic and logical operations are performed

i) Carry (CY) - Set when extra bit during addition, else reset

ii) Auxillary Carry (AC) - Used in BCD addition, set when carry is transferred from lower nibble to higher level, otherwise resel

iii) Overflow (OV) - Cannot store 256 in 8 bit register then overflow ~~oc~~ is set, else reset.

Overflow is detetted with: 9bit carry (carry into MSB) XOR 8th bit carry (carry out of MSB)

= 1 → overflow detected (set) = 0 → no overflow detected (not set)

9th bit ⊕ 8th bit

extra bit is generated — carry into MSB — carry out of MSB

1	1	0 → overflow not detected
0	0	
1	0	1 → overflow detected
0	1	

iv) Parity (P) - Always odd parity

No. of 1s in 8 bits + Parity bit = Odd no
↳ Parity Set

No. of 1s in 8 bits + Parity bit = Even no
↳ Parity reset

- Three special Purpose flags, can be access only by programmer, not by operations

i) GF0 (General Flag 0) ii) GF 1 (General Flag 1

iii) F0 (Flag 0)

- PSW is a special purpose register.

Applications:

- To check the status of flags.
- To modify the status of flags.
- To check which registers are under use.

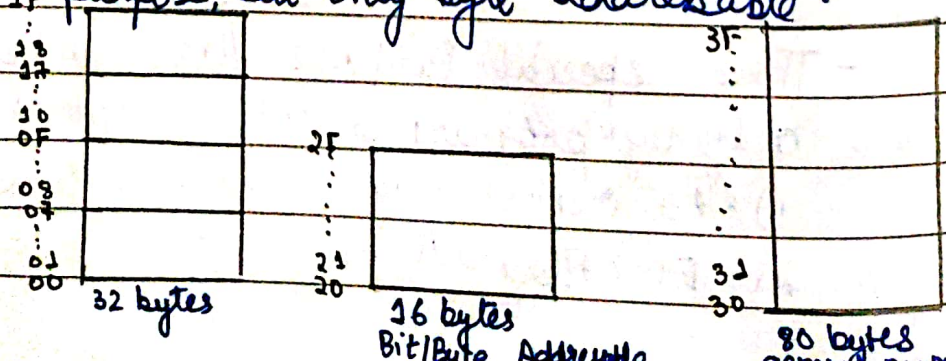
CY	AC	F0	RS1	RS0	OV	-	P

RS0	RS1		Register Bank
0	0	-	R.B.0
0	1	-	R.B.1
1	0	-	R.B.2
1	1	-	R.B.3

→ Internal Memory

Internal RAM:

- 128 bytes memory given to RAM.
- First 32 bytes given to register banks, each consists 8 registers (4×8=32) bit addressable
  (RB0 + RB1 + RB2 + RB3 = 8 + 8 + 8 + 8)
- After that, 16 bytes memory are accessed by bit addresses / byte addresses.
- Remaining 80 bytes are used for general purpose, but only byte addressable.



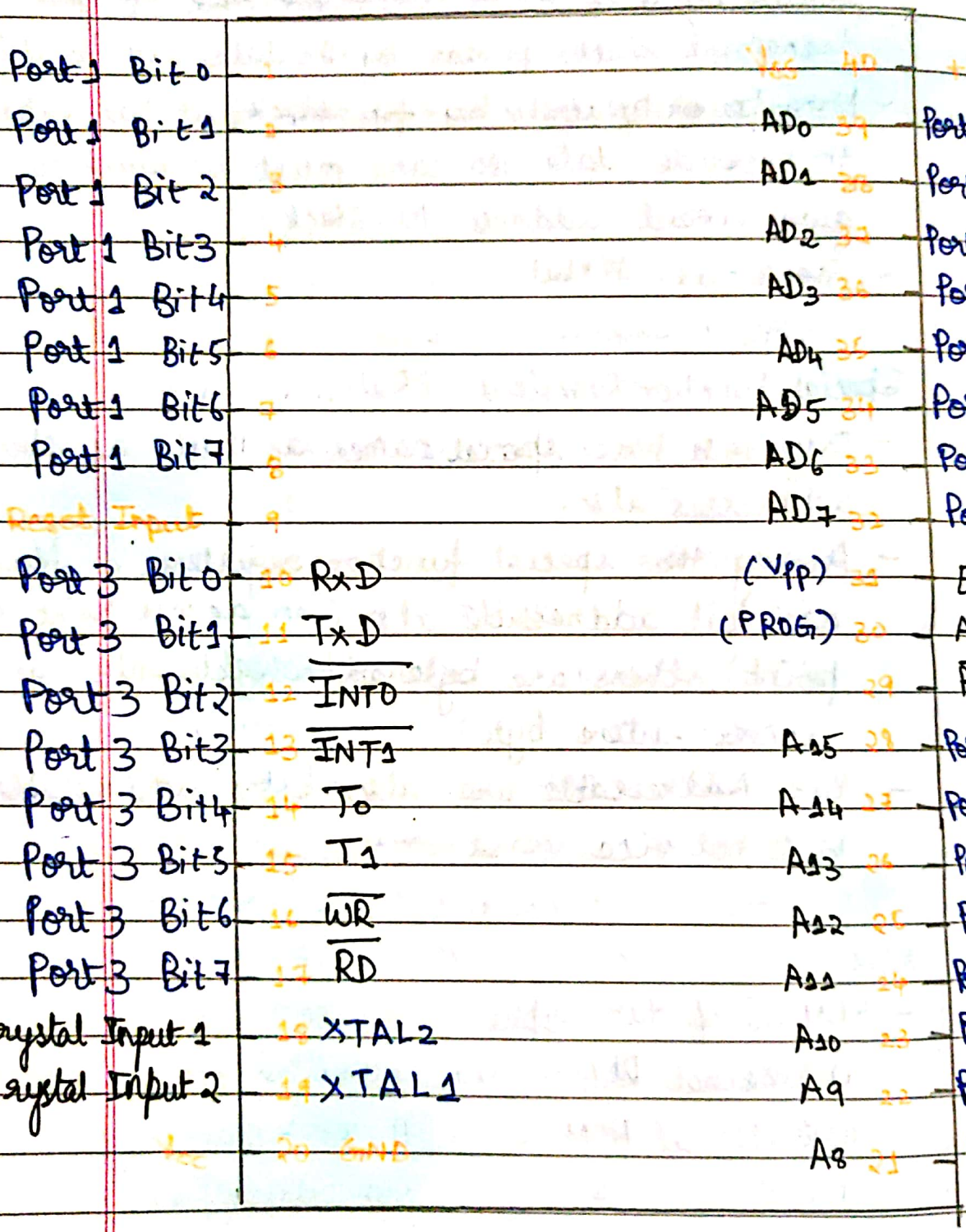
-Stack Memory : From internal memory, some part of RAM is used for stack pointer, increament the stack pointer by 1 if data is inserted, and if you decrement stack pointer by 1, data will be deleted.
- More priority will be for stack, it has capacity to override data, so care must be taken to give initial address to stack.
- Stack is 8 bit.

Special Function Registers (SFR's)
- They will have special names as well as special addresses also.
- Among the special function registers, a few are bit addressable also (can access point by point), others are byte addressable only (can access entire byte)
- Bit Addressable are also byte addressable, but not vice versa.

ROM
- ROM is of two types:
  i) Internal ROM
  ii) External ROM

★ Internal Memory
* Internal RAM
* Stack and SP
* Special Function Registers (SFR's)
* Internal ROM

8051 Microcontroller Architecture - 40 Pin Diagram



- Port 0, Port 1, Port 2, Port 3 - All pins can be used as i/p pins or o/p pins.
- Port 0, Port 2 can be used for dual purpose, i.e., accessing address and accessing data.
- Port 0 is used to access lower byte address [A0-A7]
- Port 2 is used to access higher byte address [A8 - A15]
- Port 3 has control pins Read, Write, Receive Data, Transmit Data.
- $\overline{RD}$ = 0 → Performing Read Operation
  $\overline{WR}$ = 0 → Performing Write Operation
- 8 x 4 = 32 Port Pins
- Other Pins:
  - Vcc - Ground Control
  - Vss - Frequency / Voltage
  - Crystal input 1 - } To provide frequency
  - Crystal Input 2 - } to device
  - Reset Input - To reset the microcontroller
  - EA
  - ALE
  - $\overline{PSEN}$

  } Access external RAM, ROM

- Addresses of Special Function Registers

A - 0E0	P0 - 80	PCON - 87	TCON - 88
B - 0F0	P1 - 90	PSW - 0D0	TL0 - 8A
DPH - 83	P2 - A0	SCON - 98	TH0 - 8C
DPL - 82	P3 - 0B0	SBUF - 99	TL1 - 8B
IE - 0A8		SP - 81	TH1 - 8D
IP - 0B8		TMOD - 89	

10-07-19

09

Counters and Timers

→ TCON

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
7	6	5	4	3	2	1	0
001B		000B		0013 P3.3 INT1		0003 P3.3 INT0	

- Timers are used to calculate / count internal clock pulses.
- Clocks are used to calculate / count external clock pulses.
- Processor can also do all these, but if a processor has to do its own tasks and also couting, burden on processor is more.
- So, to reduce burden on processors, two hardware chips, externally added 16 bit registers, timers and counters are used.
- They are used as 2×8 bit registers
  TL0 + TH0 = Timer 0
  TL1 + TH1 = Timer 1
  L = Lower, H = Higher
- To find 'n' number of clock pulses, timer must be initialized with max − n + 1 value.
  Eg: Maximum = 10, we wan to count 5 events, timer must be initialized with 10 − 5 + 1 = 6

  (6) →1 (7) →2 (8) →3 (9) →4 (10) →5 (11)

  When overflow occurs, flag is set.
  When count comes to 0000, flag is set.

- We use '1' in max-n+1 because when the counter changes from FF to 00, the timer/counter counts that also, and we use that as interrupt.
- Whenever timer becomes 00, we use that as interrupt and can perform a particular task.
- Eg. 60 sec green light in traffic signals.
- Two special function registers
  Timer Control Register (TCON) and
  Timer Mode Register (TMOD)
  two 16 bit register control two timers.

TCON:

i) Timer Flag 1 (TF1)
ii) Timer Run 1 (TR1)
iii) Timer Flag 0 (TF0)
iv) Timer Run 0 (TR0)
v) Interrupt Edge 1 (IE1)
vi) Interrupt Signal Timer (IT1)
vii) Interrupt Edge 0 (IE0)
viii) Interrupt Signal Timer (IT0)

TCON is bit addressable, can change bit by bit in programming and also byte addressable.

* Timer 0 to count 'n', whenever Timer 0 moves from FF to 00, i.e., it overflows, TF0 becomes 1. Indicates counting completed.
* Timer 1 to count 'n', whenever Timer 1 moves from FF to 00, i.e., it overflows, TF1 becomes 1. Indicates counting completed.

* Both are reset / become 0 whenever microprocessor executes an interrupt service routine (whenever interrupts occur, automatically jump to particular address to execute task) present in address 001B (TF1) and 000B (TF0).

* TR1 is going to be set by programmer if we want to calculate internal clock pulses, i.e. (using T1 as timer) used for timing purpose.
* TR1=0 indicates we are not using timing, T1 does not work as timer.

* TR0 is going to be set by programmer if we want to calculate interal clock pulses, i.e., using T0 as timer, used for timing purpose.
* TR0=0 indicates we are not using timing, T0 does not work as timer.

* PCON Register
  Higher Nibble = Timer Flags
  Lower Nibble = Interrupt Flags

* IE1 - Wherever on P3.3 pin high to low (⎍) pin encountered, i.e., $\overline{INT1}$ pin of port 3 encounters change of high to low edge, it indicates internal interrupt occured, and IE1 is set, ~~otherwise reset~~..

* IE0 - Whenever on P3.3 pin high to low (↓) pin is encountered, i.e., $\overline{INT0}$ pin of port 3 encounters change of high to low edge, it indicates external interrupt has occured, IE0 is set.

* IE1=0 when process jumps to execute interrupt service routine at address 0013.
* IE0=0 when process jumps to execute interrupt service routine at address 0003.

* IT1 and IT0 are internal interrupts. User can set or reset them based on the task to be performed.

TMOD:

Gate	C/$\overline{T}$	M1	M0	Gate	C/$\overline{T}$	M1	M0
7	6	5	4	3	2	1	0
Timer 1				Timer 0			

- TMOD has two sets of four flags each, one set for Timer 0 and another set for Timer 1.
- The flags are duplicate ones. We can write only 4 bits.
- ~~The~~ TMOD is byte addressable, not bit addressable.
- Flags:
  i) C/$\overline{T}$: Counter/Timer ii, iii) M0, M1 - Mode Selection Bits
  iv) Gate - Gate Bit.

* $C/\bar{T}$ = Counter / Timer } Used to know whether we use it as timer / counter.

$C/\bar{T} = 0$ → Used as a timer

↳ Bit 6 = 0 => Timer 1 used as timer

↳ Bit 2 = 0 => Timer 0 used as timer

$C/\bar{T} = 1$ → Used as a counter

↳ Bit 6 = 1 => Timer 1 used as clock

↳ Bit 2 = 1 => Timer 1 used as clock

* M1 and M0 → Mode selection bits

Used when we use timers

Important when $C/\bar{T} = 0$, does not matter when $C/\bar{T} = 1$.

M1	M0	Timer Mode
0	0	0
0	1	1
1	0	2
1	1	3

- ~~Timer~~ Mode

* Gate Bit

- This device acts as a timer when we encounter conditions.
- When T0/T1 works in timer mode, i.e., used to clock internal clock pulses:

i) Timer 1/0 bit in TCON register must be 1 AND

ii) (a) Either gate bit in TMOD = 0 or

(b) Interrupt 1/0 must be 1.