Microcontroller

By

Prof.Mahesh Yanagimath

M.Tech(VLSI &ES), MIEEE, MISTE.

Assistant Professor,Dept of EEE
HIT Nidasoshi.

- What is Microcontroller?
- Why the name Microcontroller”?
- Why it is required?
- Where it is Used?

Why do we need to learn Microprocessors/controllers?

- The microprocessor is the core of computer systems.
- Many communication, digital entertainment, portable devices, are controlled by them.
- A designer should know what types of components he needs, ways to reduce production costs and product reliable.

Three criteria in Choosing a Microcontroller

1. Meeting the computing needs of the task efficiently and cost effectively
   - speed, the amount of ROM and RAM, the number of I/O ports and timers, size, packaging, power consumption
   - easy to upgrade
   - cost per unit
2. Availability of software development tools
   - assemblers, debuggers, C compilers, emulator, simulator, technical support
3. Wide availability and reliable sources of the microcontrollers.

Different aspects of a microprocessor/controller

- Hardware : Interface to the real world
- Software : order how to deal with inputs

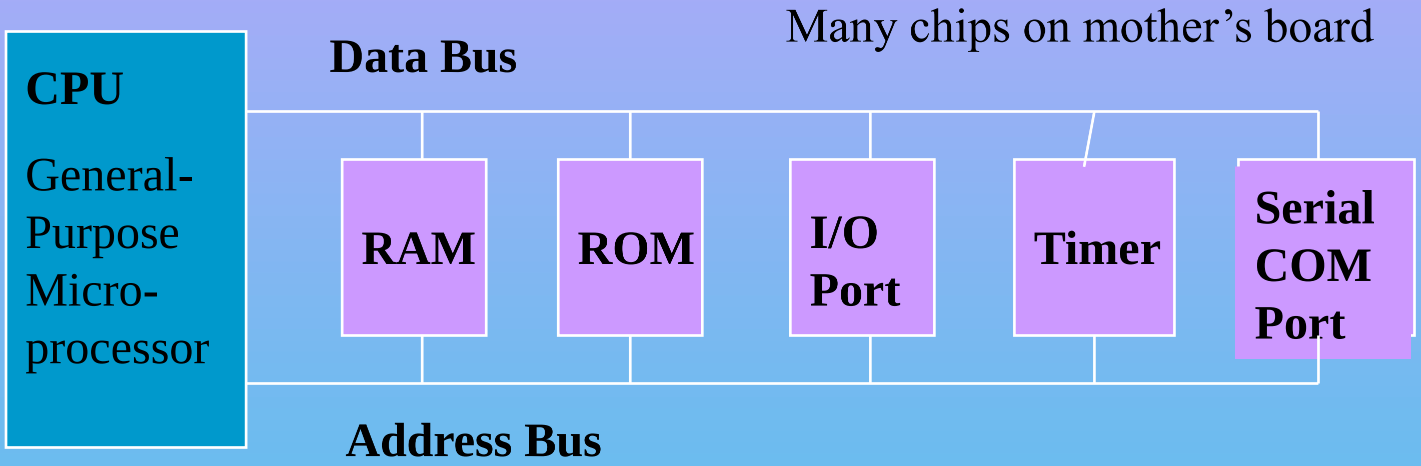
The necessary tools for a microprocessor/controller

- CPU: Central Processing Unit
- I/O: Input /Output
- Bus: Address bus & Data bus
- Memory: RAM & ROM
- Timer
- Interrupt
- Serial Port
- Parallel Port

Microprocessors:

General-purpose Microprocessor

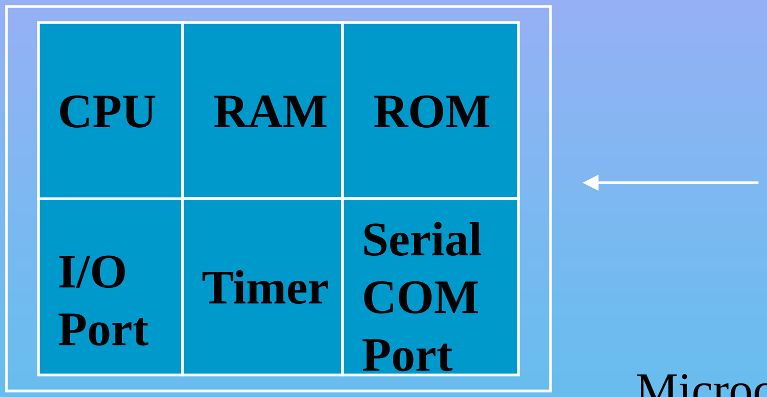
- CPU for Computers
- No RAM, ROM, I/O on CPU chip itself
- Example：Intel’s x86, Motorola’s 680x0



General-Purpose Microprocessor System

Microcontroller :

- A smaller computer
- On-chip RAM, ROM, I/O ports...
- Example：Motorola’s 6811, Intel’s 8051, Zilog’s Z8 and PIC 16X



CPU RAM ROM

I/O Port Timer Serial COM Port

A single chip

Microcontroller

Microprocessor vs. Microcontroller

Microprocessor

- CPU is stand-alone, RAM, ROM, I/O, timer are separate
- Designer can decide on the amount of ROM, RAM and I/O ports.
- Expensive
- Versatility
- General-purpose

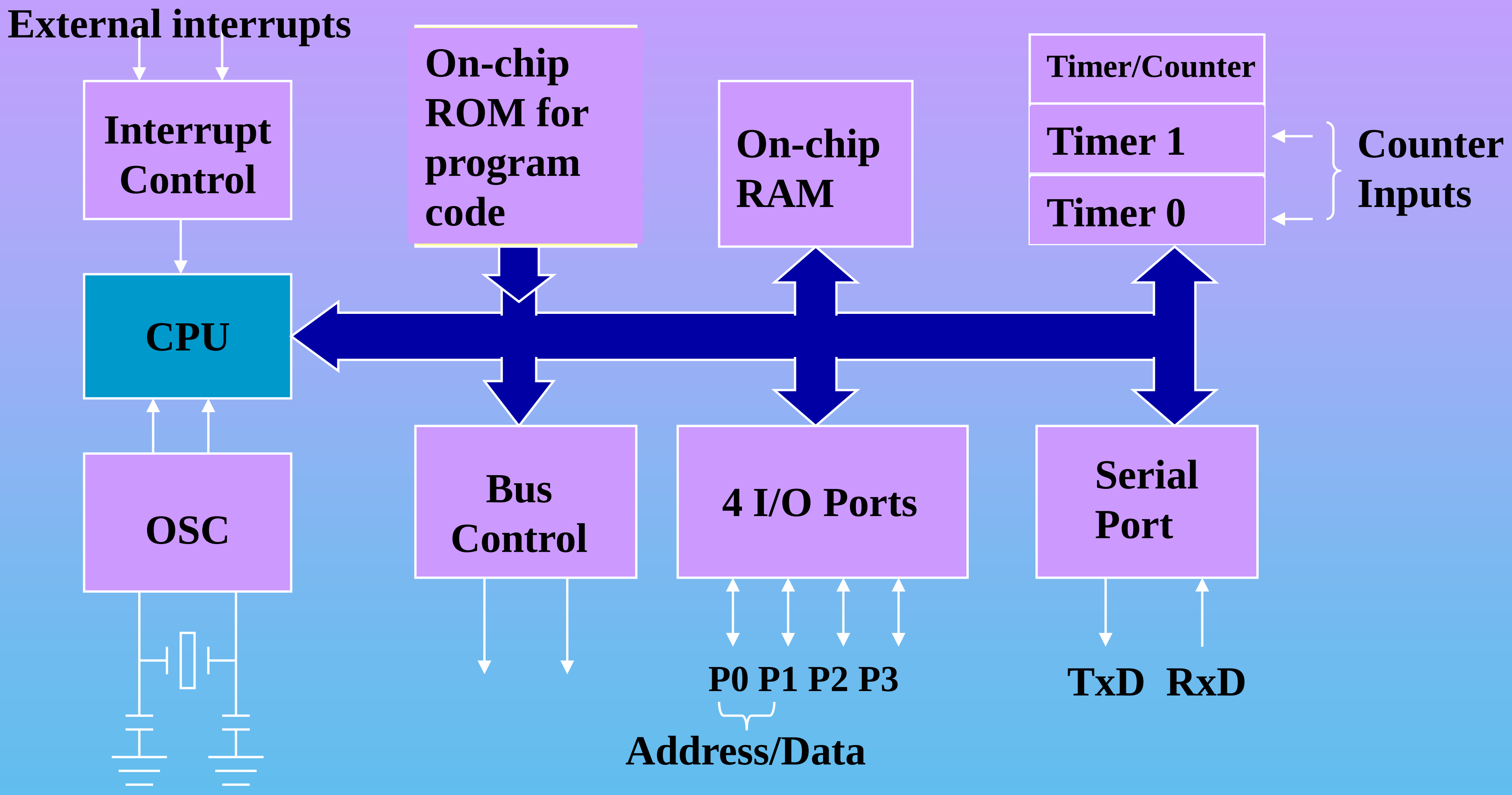
Microcontroller

- CPU, RAM, ROM, I/O and timer are all on a single chip
- Fixed amount of on-chip ROM, RAM, I/O ports
- For applications in which cost, power and space are critical
- Single-purpose

Embedded System

- Embedded system means the processor is embedded into that application.
- An embedded product uses a microprocessor or microcontroller to do one task only.
- In an embedded system, there is only one application software that is typically burned into ROM.
- Example：printer, keyboard, video game player

Block Diagram


External interrupts

Interrupt Control

On-chip ROM for program code

On-chip RAM

Timer/Counter

Timer 1

Timer 0

Counter Inputs

CPU

OSC

Bus Control

4 I/O Ports

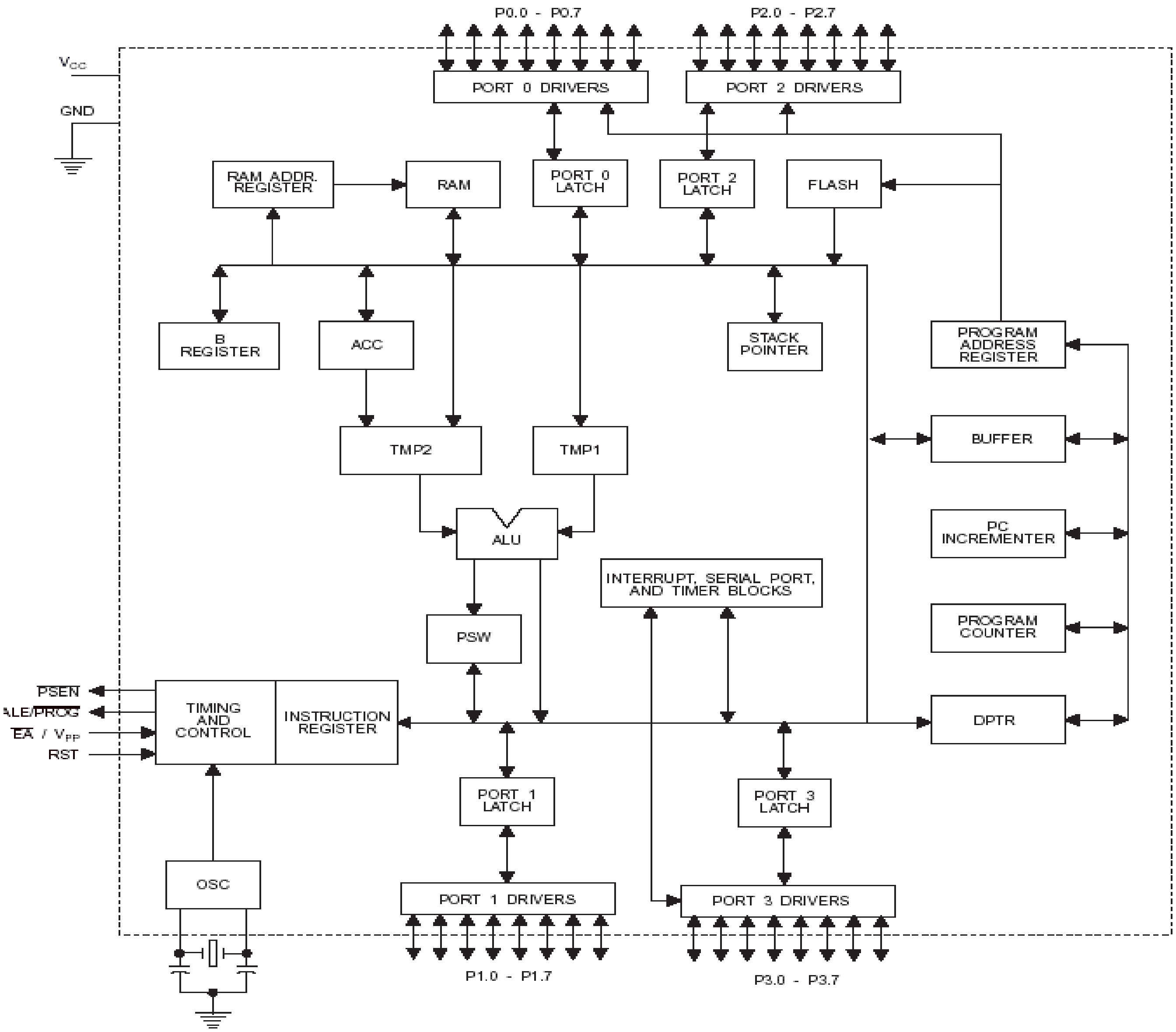
Serial Port

P0 P1 P2 P3

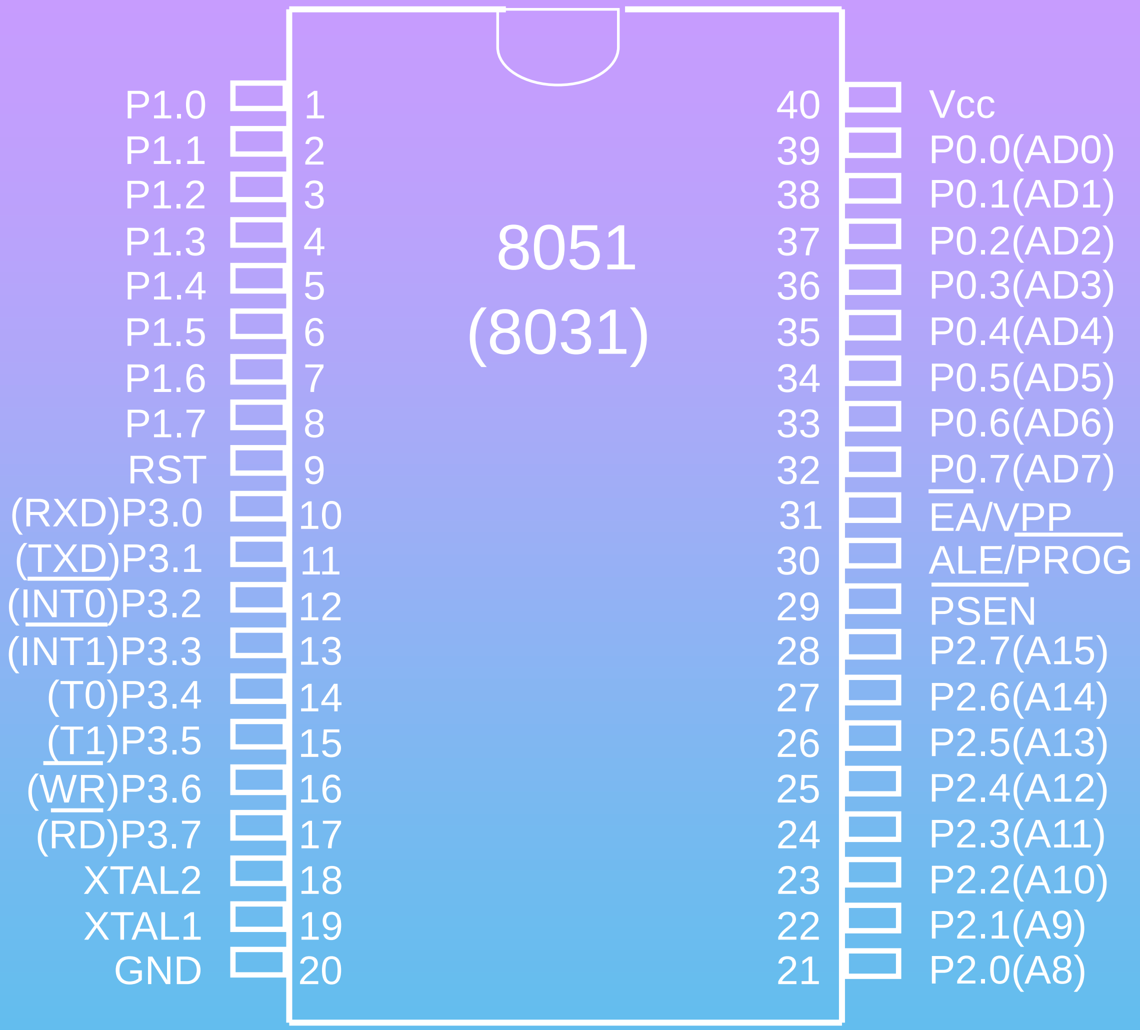
Address/Data

TxD RxD

Block Diagram



Pin Description of the 8051



8051

(8031)

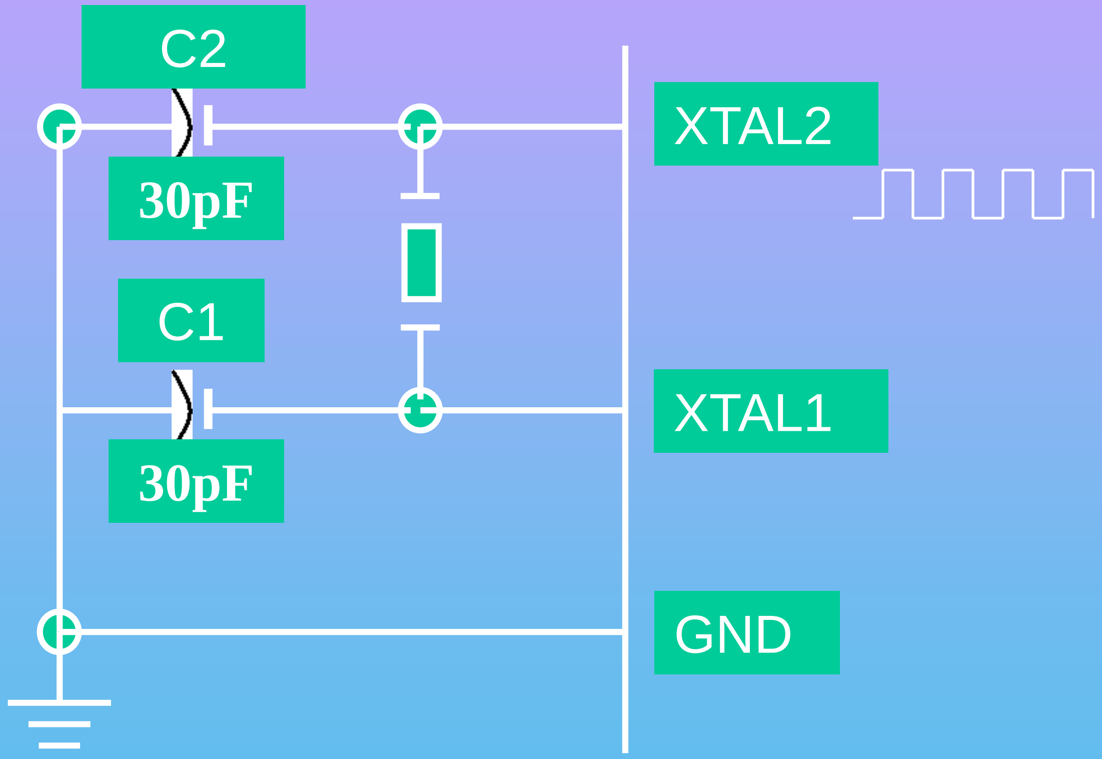
P1.0	1	40	Vcc
P1.1	2	39	P0.0(AD0)
P1.2	3	38	P0.1(AD1)
P1.3	4	37	P0.2(AD2)
P1.4	5	36	P0.3(AD3)
P1.5	6	35	P0.4(AD4)
P1.6	7	34	P0.5(AD5)
P1.7	8	33	P0.6(AD6)
RST	9	32	P0.7(AD7)
(RXD)P3.0	10	31	EA/VPP
(TXD)P3.1	11	30	ALE/PROG
(INT0)P3.2	12	29	PSEN
(INT1)P3.3	13	28	P2.7(A15)
(T0)P3.4	14	27	P2.6(A14)
(T1)P3.5	15	26	P2.5(A13)
(WR)P3.6	16	25	P2.4(A12)
(RD)P3.7	17	24	P2.3(A11)
XTAL2	18	23	P2.2(A10)
XTAL1	19	22	P2.1(A9)
GND	20	21	P2.0(A8)

Pins of 8051（1/4）

- Vcc（pin 40）：
  - Vcc provides supply voltage to the chip.
  - The voltage source is +5V.
- GND（pin 20）：ground
- XTAL1 and XTAL2（pins 19,18）

Figure (a). XTAL Connection to 8051

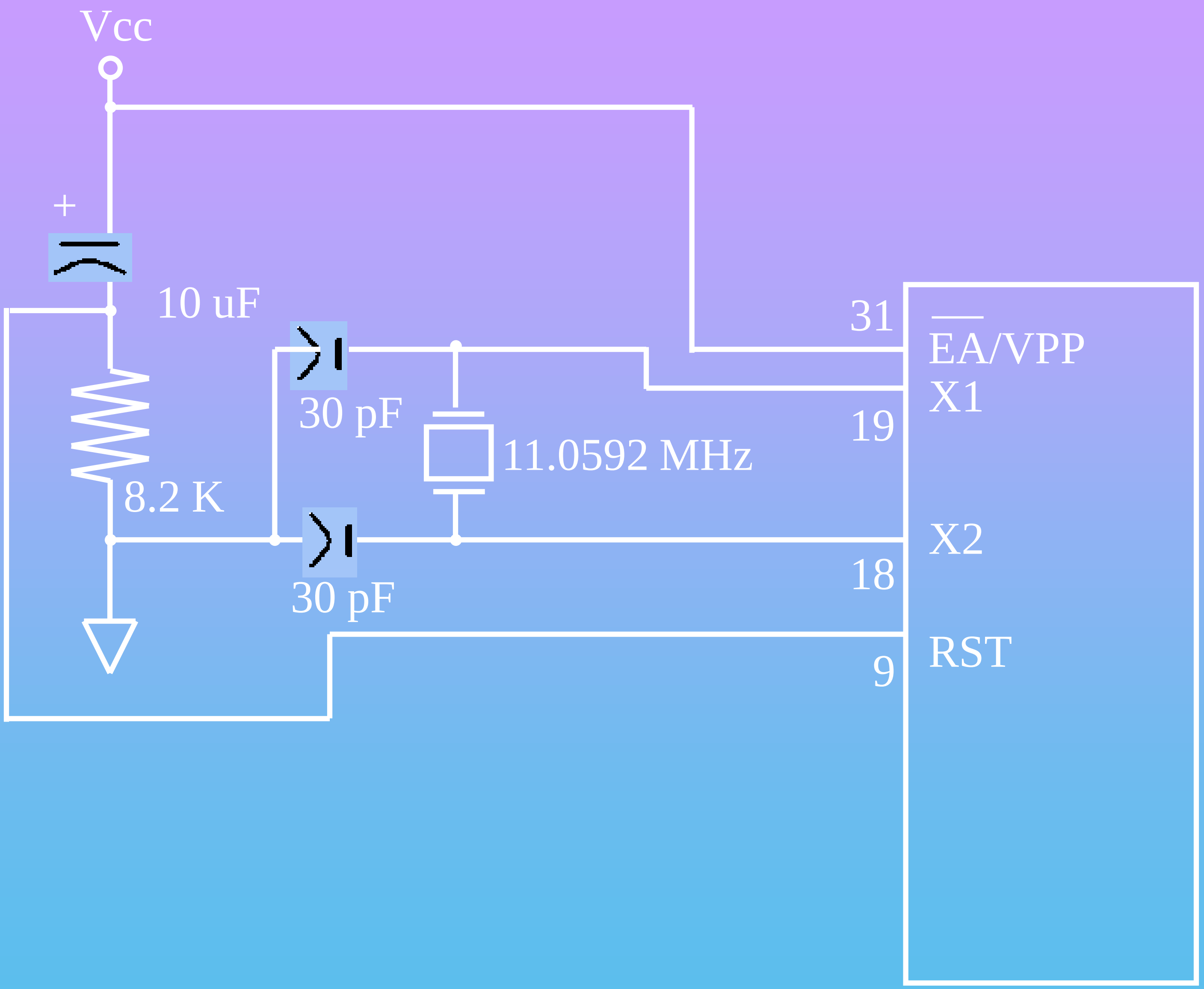
- Using a quartz crystal oscillator
- We can observe the frequency on the XTAL2 pin.



Pins of 8051（2/4）

- RST（pin 9）：reset
  - It is an input pin and is active high（normally low）．
    - The high pulse must be high at least 2 machine cycles.
  - It is a power-on reset.
    - Upon applying a high pulse to RST, the microcontroller will reset and all values in registers will be lost.
    - Reset values of some 8051 registers

Figure (b). Power-On RESET Circuit



Pins of 8051（3/4）

- /EA（pin 31）：external access
  - There is no on-chip ROM in 8031 and 8032 .
  - The /EA pin is connected to GND to indicate the code is stored externally.
  - /PSEN ＆ ALE are used for external ROM.
  - For 8051, /EA pin is connected to Vcc.
  - “/” means active low.
- /PSEN（pin 29）：program store enable
  - This is an output pin and is connected to the OE pin of the ROM.

Pins of 8051（4/4）

- ALE（pin 30） ：address latch enable
  - It is an output pin and is active high.
  - 8051 port 0 provides both address and data.
  - The ALE pin is used for de-multiplexing the address and data by connecting to the G pin of the 74LS373 latch.
- I/O port pins
  - The four ports P0, P1, P2, and P3.
  - Each port uses 8 pins.
  - All I/O pins are bi-directional.

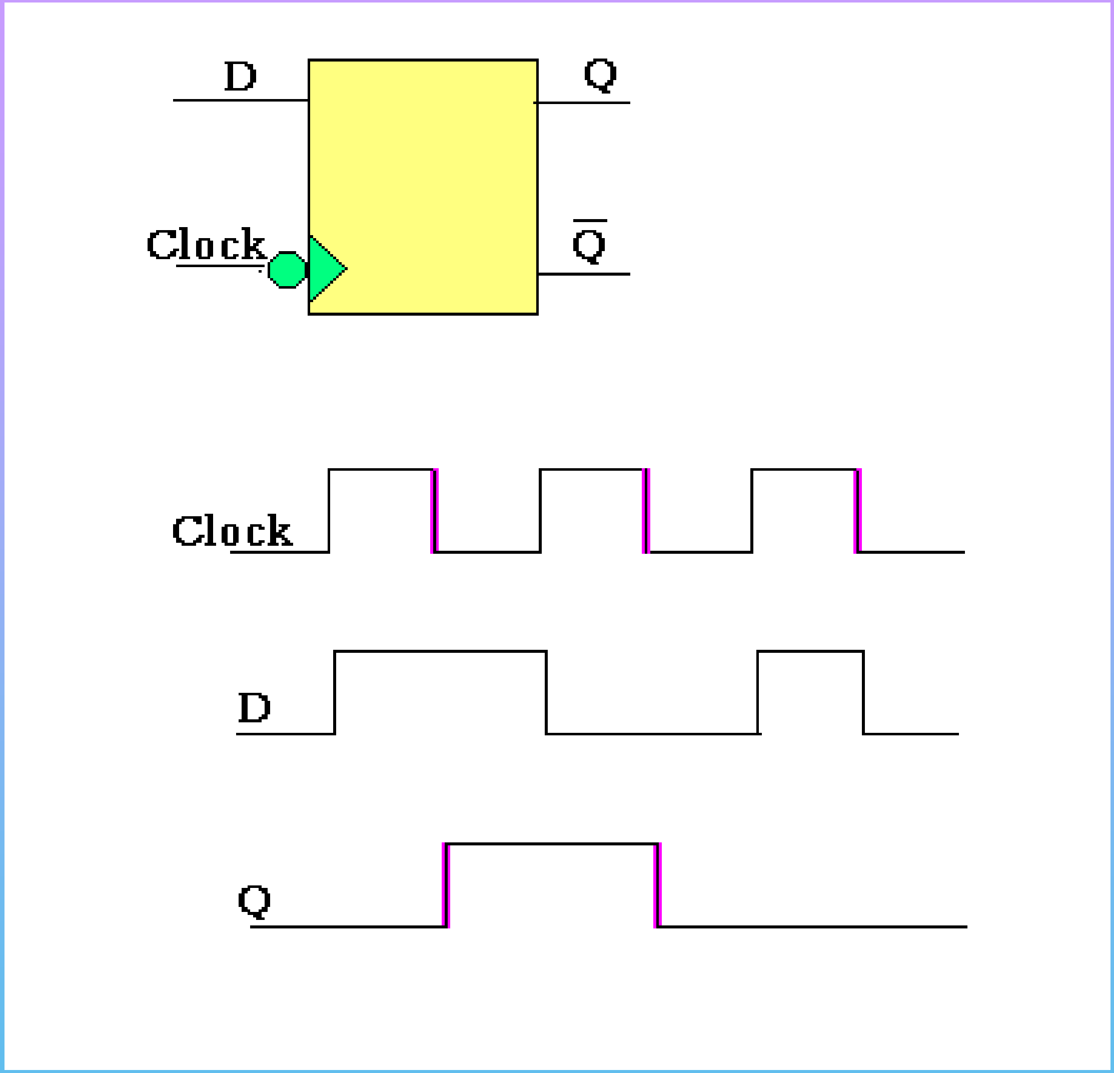
Pins of I/O Port

- The 8051 has four I/O ports
  - Port 0 （pins 32-39）：P0（P0.0～P0.7）
  - Port 1（pins 1-8）：P1（P1.0～P1.7）
  - Port 2（pins 21-28）：P2（P2.0～P2.7）
  - Port 3（pins 10-17）：P3（P3.0～P3.7）
  - Each port has 8 pins.
    - Named P0.X （X=0,1,...,7）, P1.X, P2.X, P3.X
    - Ex：P0.0 is the bit 0（LSB）of P0
    - Ex：P0.7 is the bit 7（MSB）of P0
    - These 8 bits form a byte.
- Each port can be used as input or output (bi-direction).

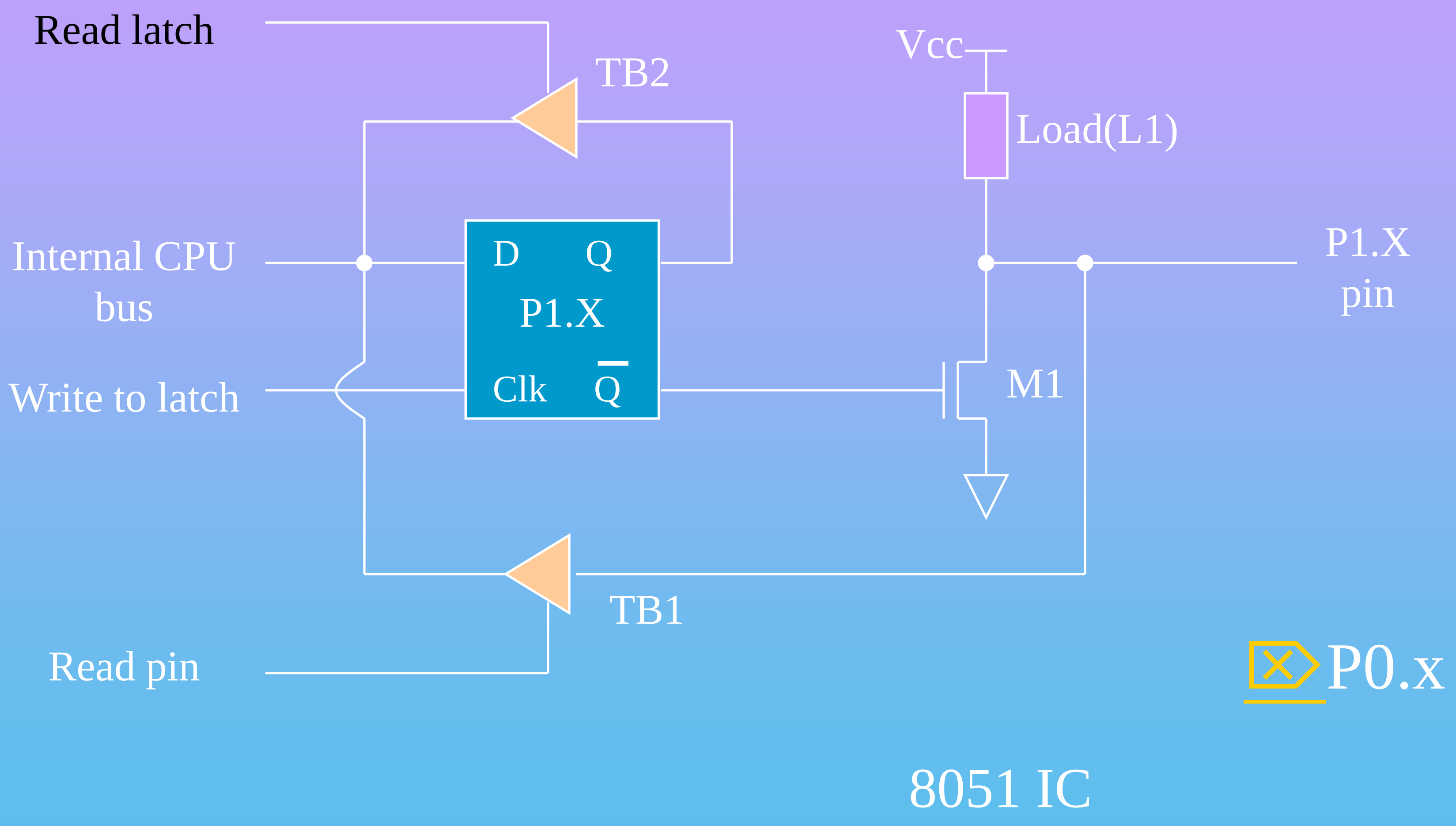
Hardware Structure of I/O Pin

- Each pin of I/O ports
  - Internal CPU bus：communicate with CPU
  - A D latch store the value of this pin
    - D latch is controlled by “Write to latch”
      - Write to latch＝1：write data into the D latch
  - 2 Tri-state buffer：
    - TB1: controlled by “Read pin”
      - Read pin＝1：really read the data present at the pin
    - TB2: controlled by “Read latch”
      - Read latch＝1：read value from internal latch
  - A transistor M1 gate
    - Gate=0: open
    - Gate=1: close

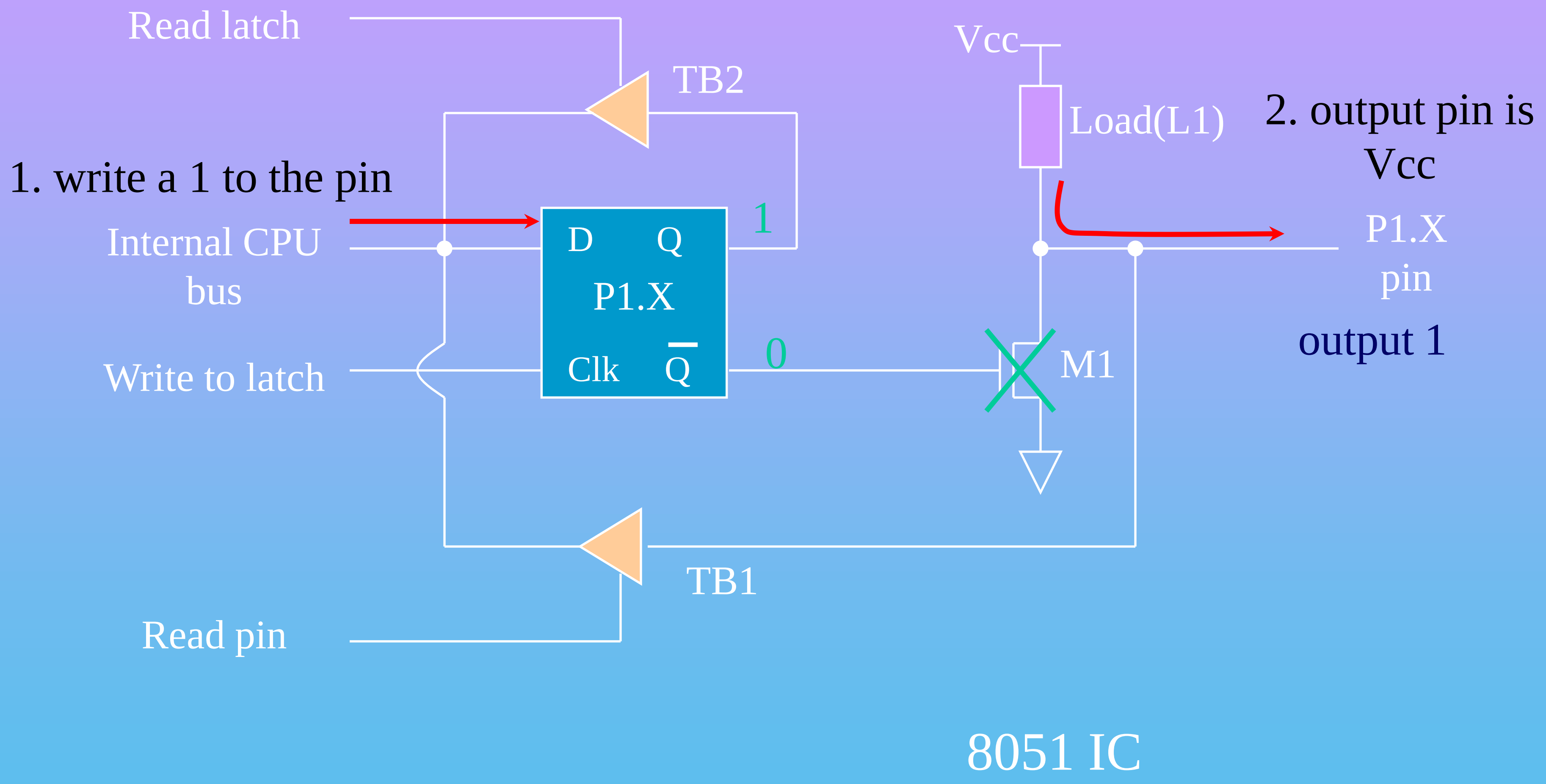
D Latch:



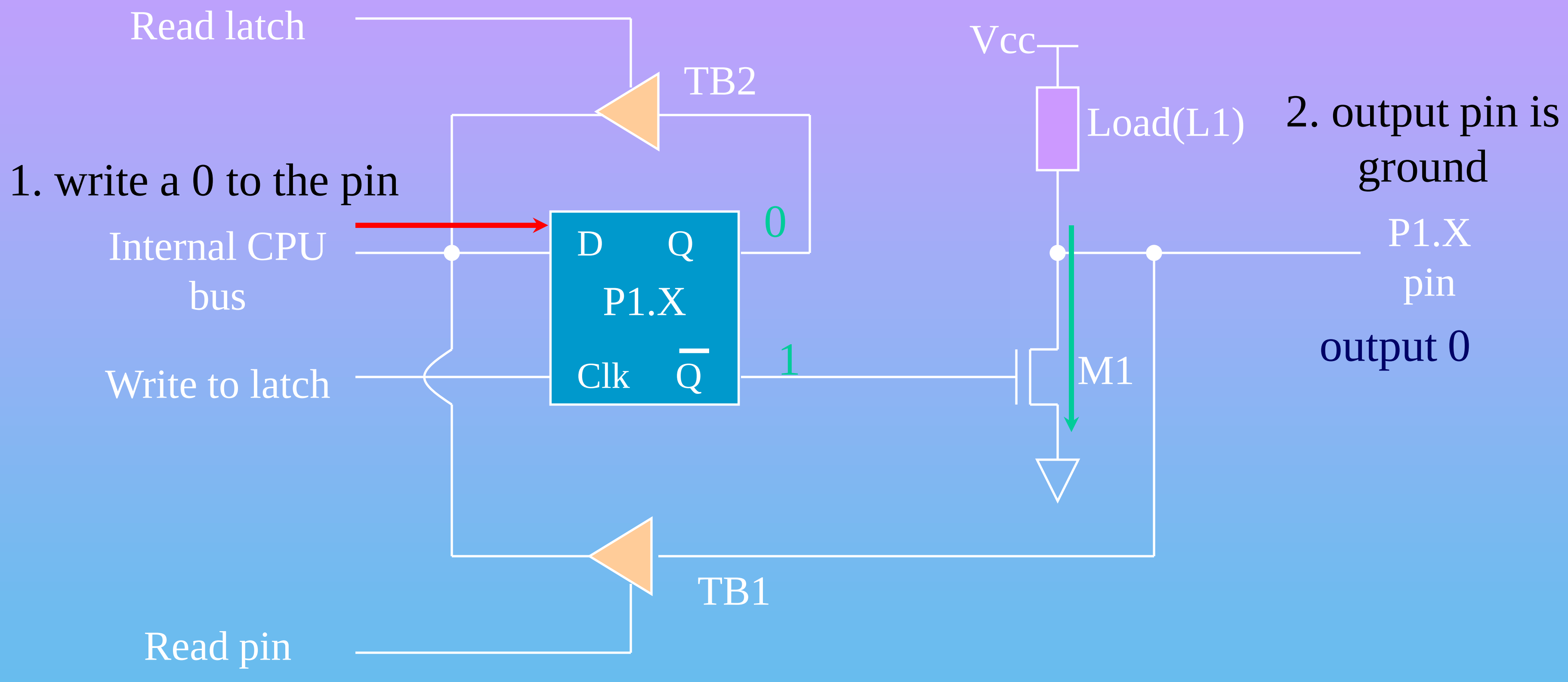
A Pin of Port 1



Writing “1” to Output Pin P1.X

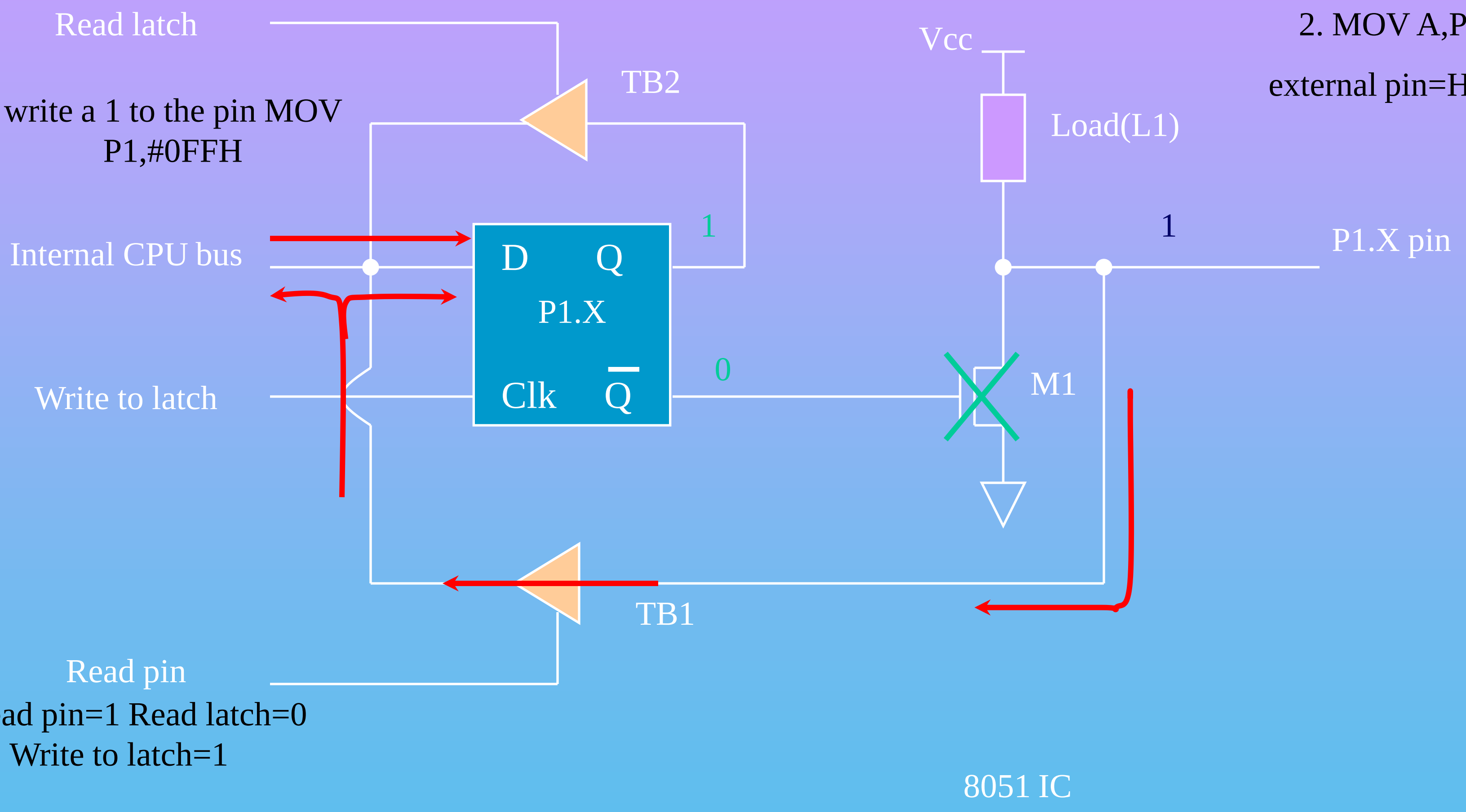


Writing "0" to Output Pin P1.X



8051 IC

Reading “High” at Input Pin



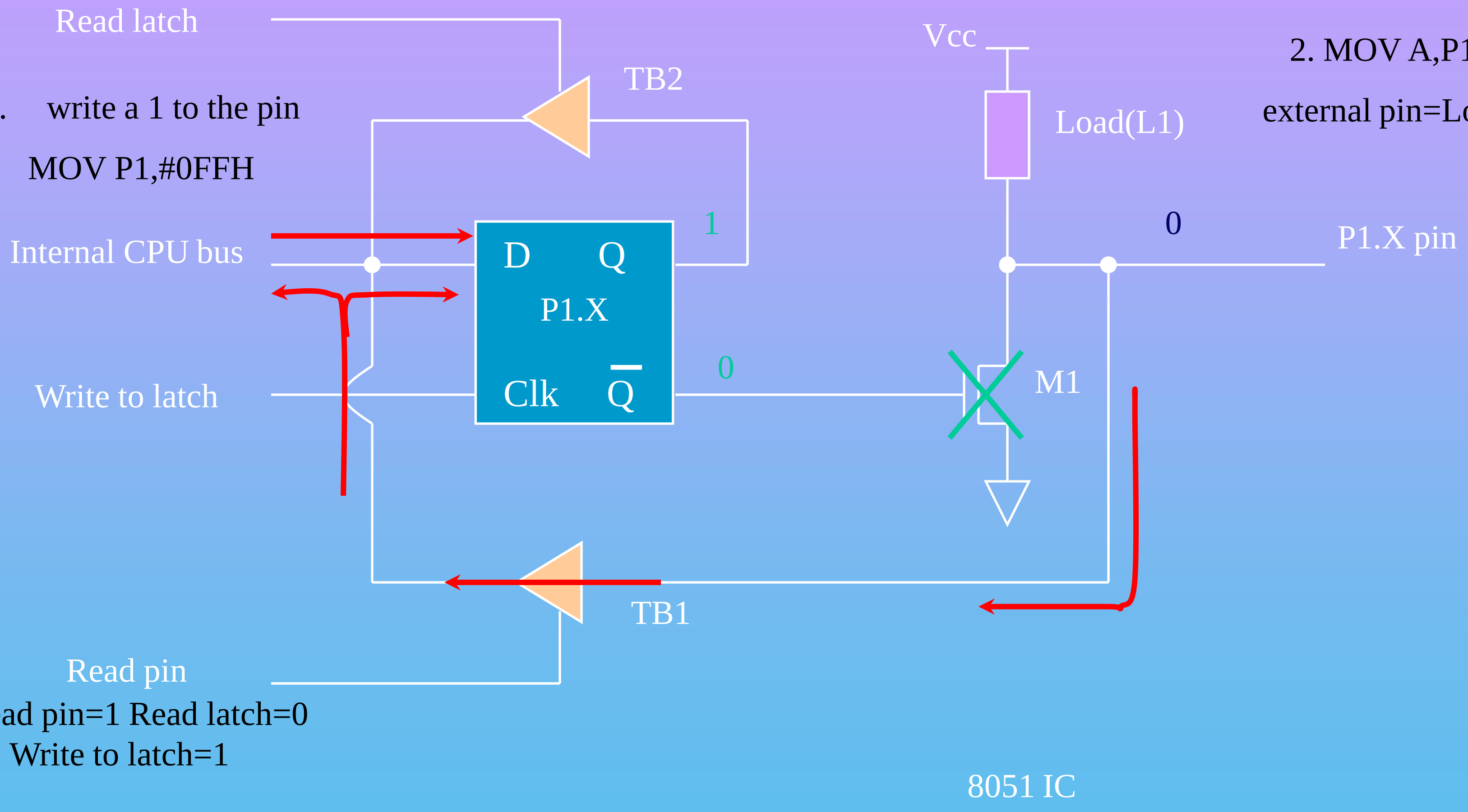
1. write a 1 to the pin MOV P1,#0FFH

2. MOV A,P1

external pin=High

3. Read pin=1 Read latch=0
Write to latch=1

Reading "Low" at Input Pin



1. write a 1 to the pin

MOV P1,#0FFH

2. MOV A,P1

external pin=Low

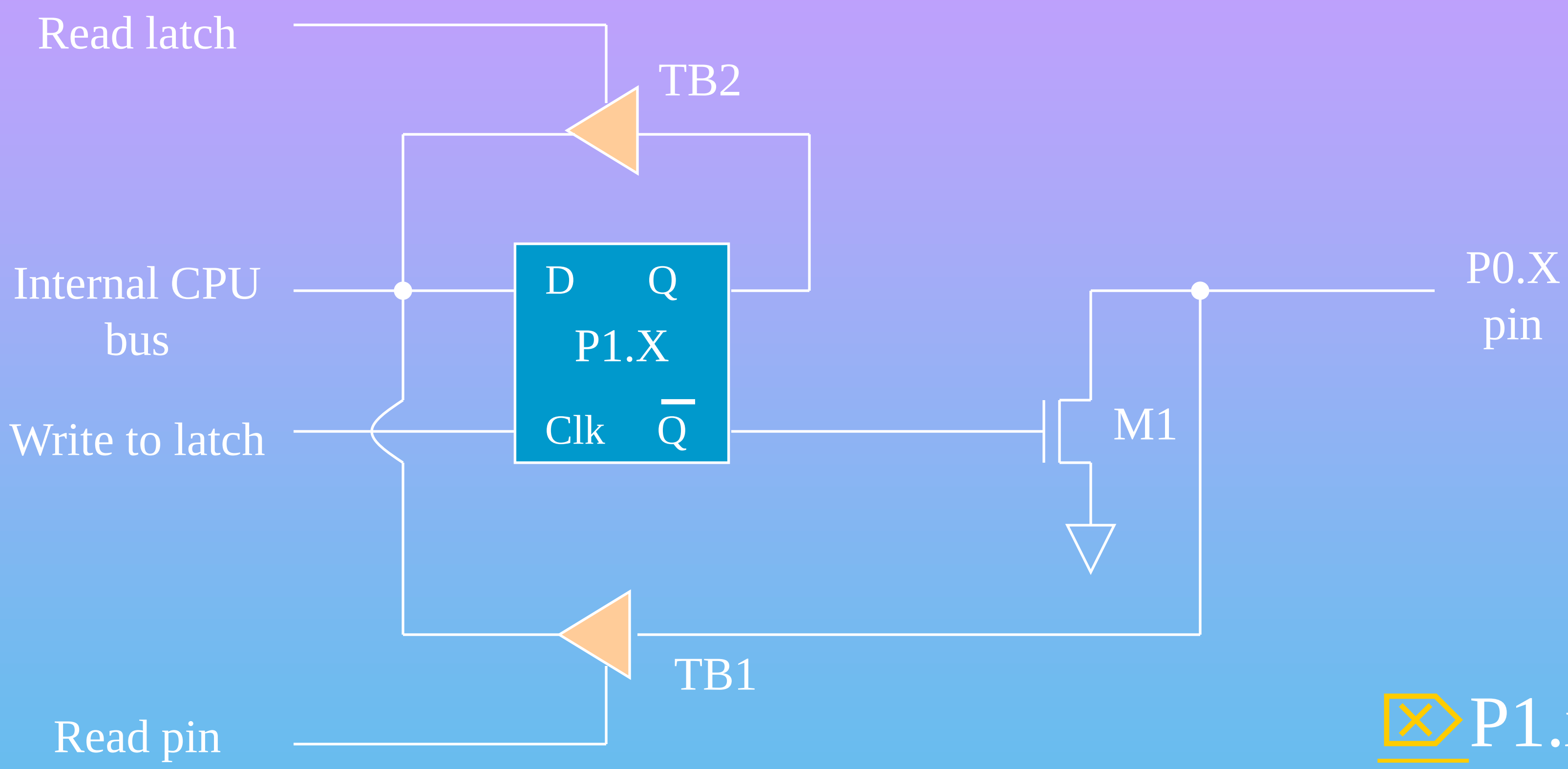
3. Read pin=1 Read latch=0

Write to latch=1

Other Pins

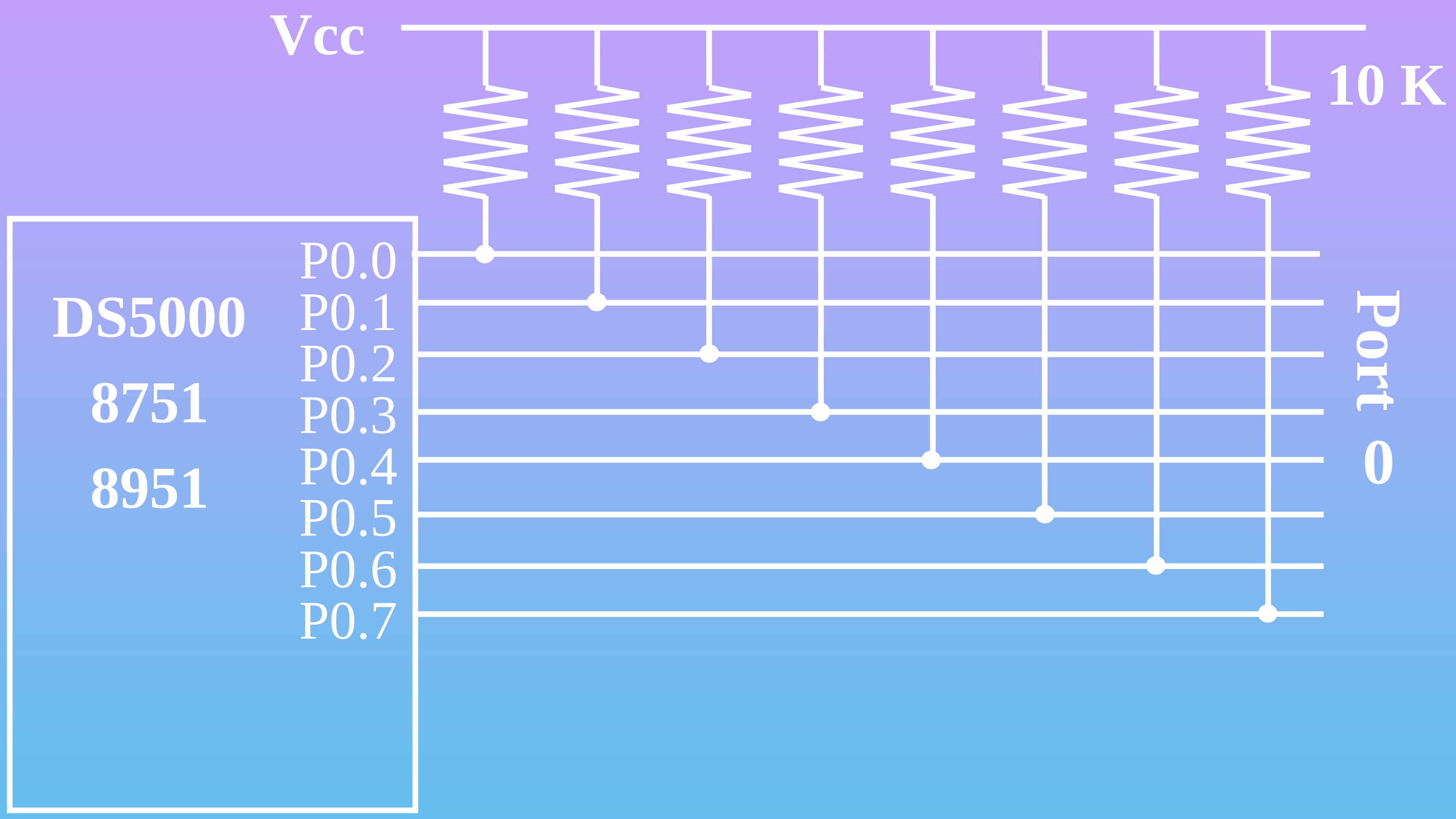
- P1, P2, and P3 have internal pull-up resisters.
  - P1, P2, and P3 are not open drain.
- P0 has no internal pull-up resistors and does not connects to Vcc inside the 8051.
  - P0 is open drain.
  - Compare the figures of P1.X and P0.X.
- However, for a programmer, it is the same to program P0, P1, P2 and P3.
- All the ports upon RESET are configured as output.

A Pin of Port 0



8051 IC

Port 0 with Pull-Up Resistors



Port 3 Alternate Functions

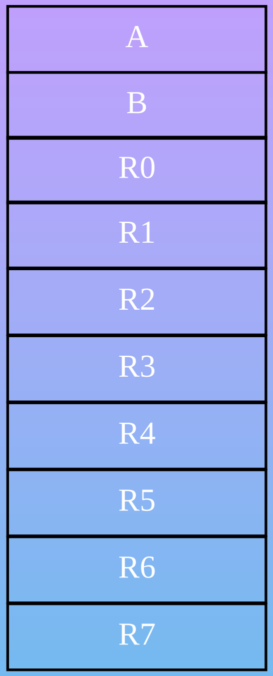
P3 Bit	Function	Pin
P3.0	RxD	10
P3.1	TxD	11
P3.2	$\overline{\text{INT0}}$	12
P3.3	$\overline{\text{INT1}}$	13
P3.4	T0	14
P3.5	T1	15
P3.6	$\overline{\text{WR}}$	16
P3.7	$\overline{\text{RD}}$	17

RESET Value of Some 8051 Registers:

Register	Reset Value
PC	0000
ACC	0000
B	0000
PSW	0000
SP	0007
DPTR	0000

RAM are all zero.

Registers



A

B

R0

R1

R2

R3

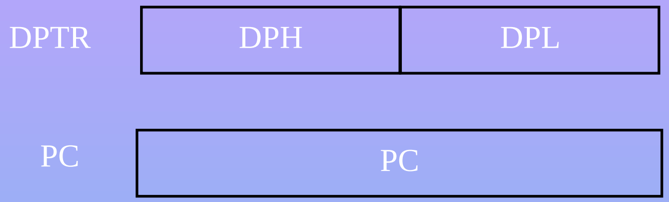
R4

R5

R6

R7

Some 8-bitt Registers of
the 8051



DPTR

DPH

DPL

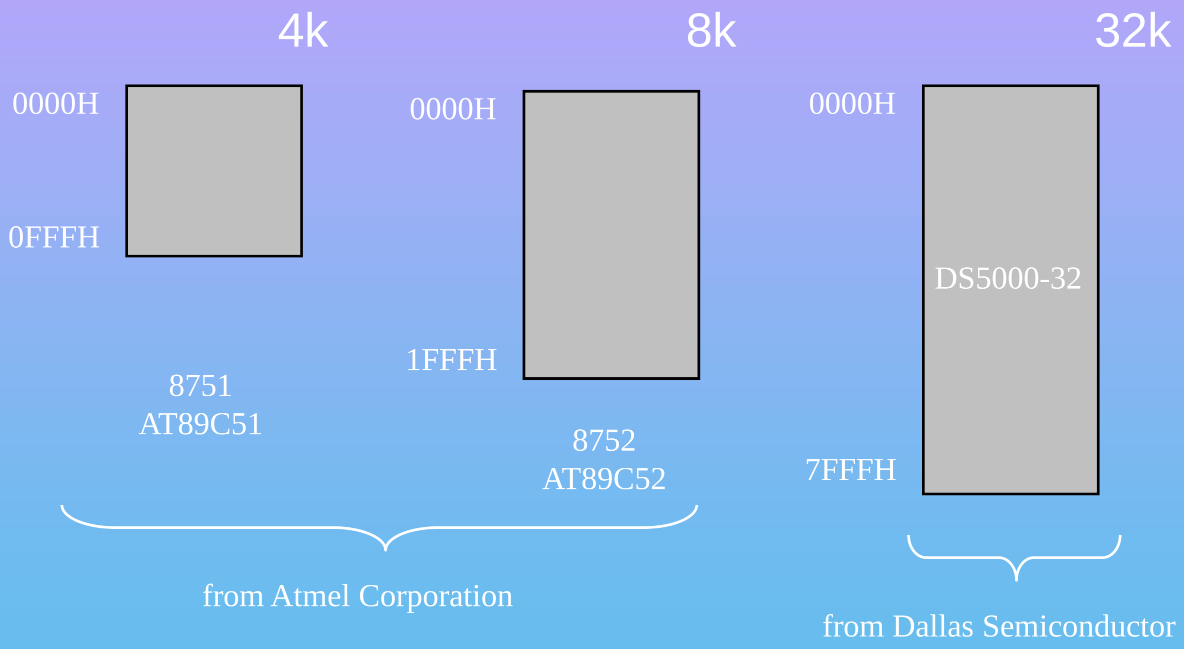
PC

PC

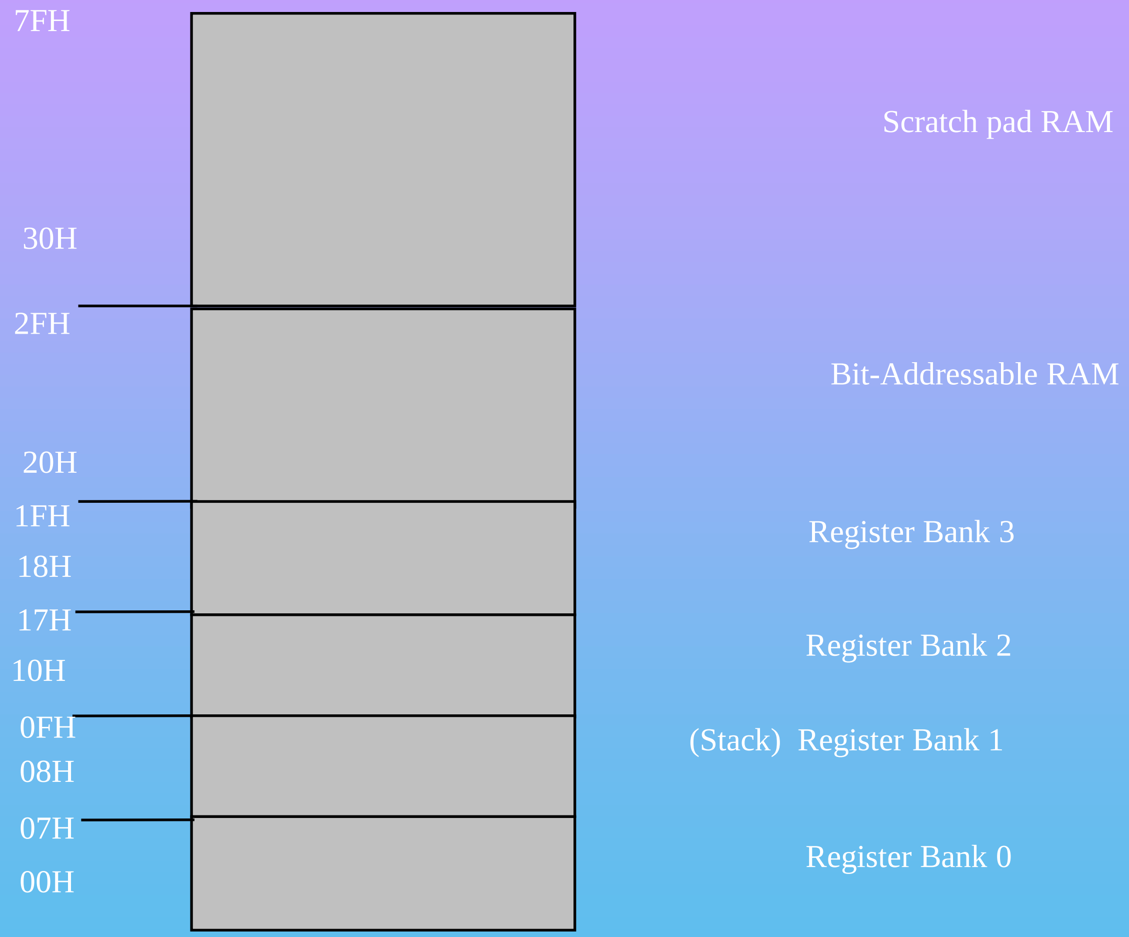
Some 8051 16-bit Register

Memory mapping in 8051

- **ROM memory map in 8051 family**



- **RAM memory space allocation in the 8051**



7FH

Scratch pad RAM

30H

2FH

Bit-Addressable RAM

20H

1FH

Register Bank 3

18H

17H

Register Bank 2

10H

0FH

(Stack) Register Bank 1

08H

07H

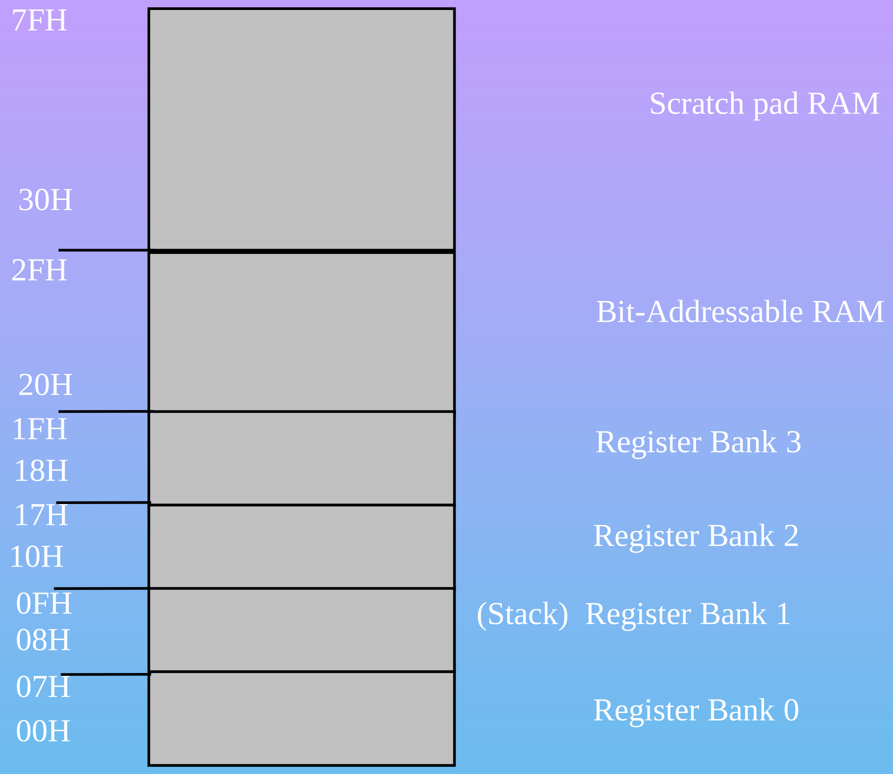
Register Bank 0

00H

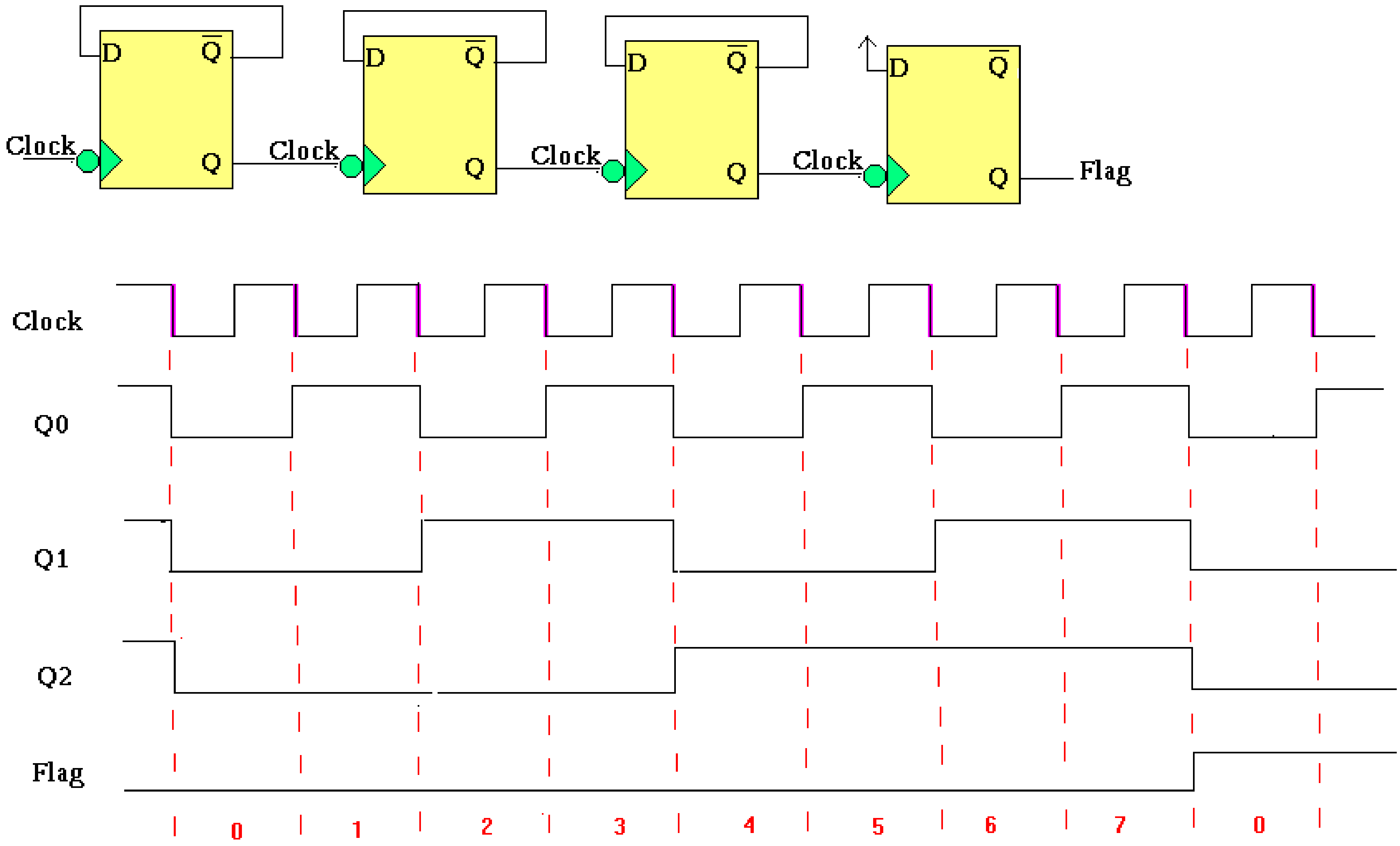
Stack in the 8051

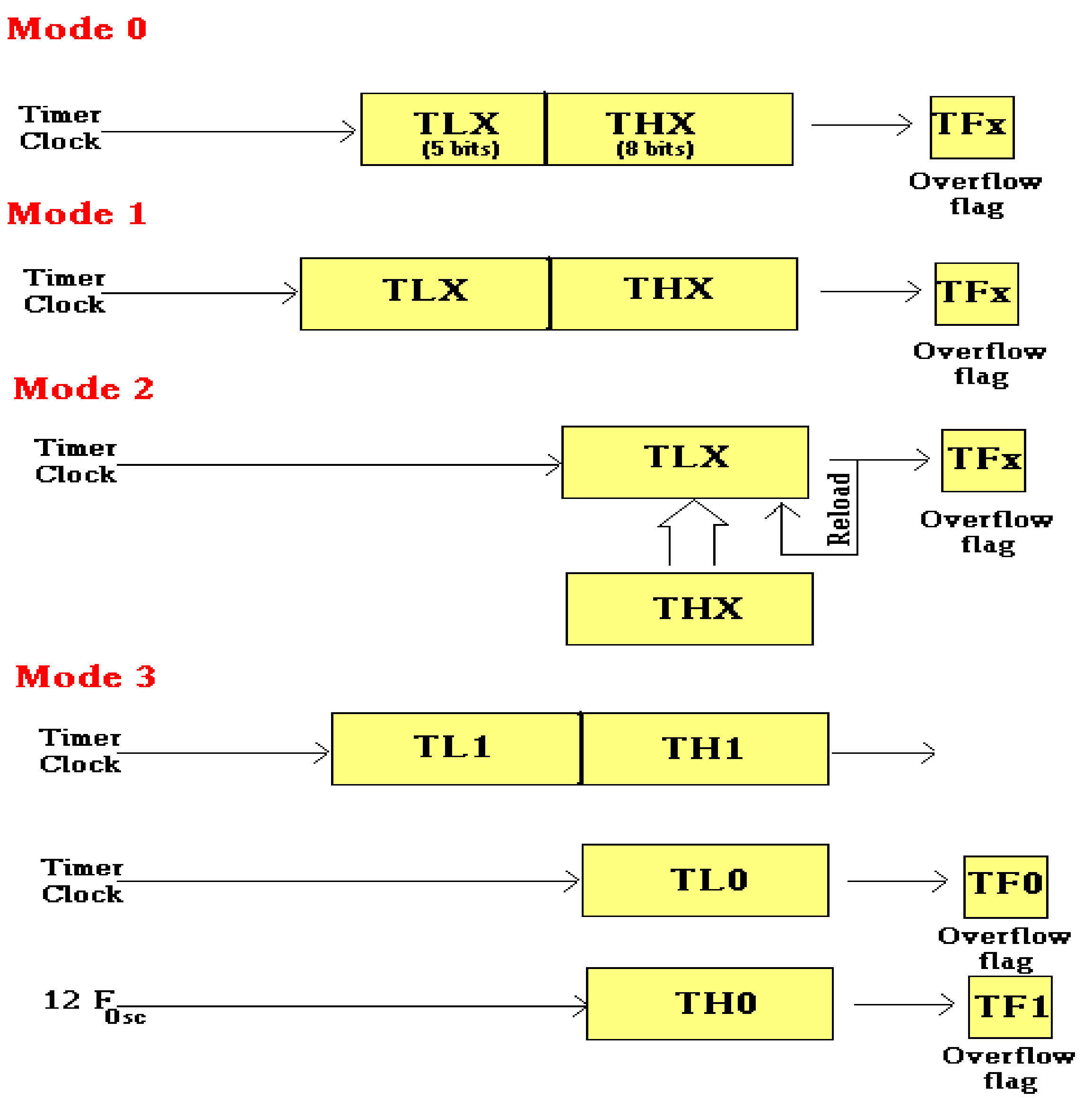
- The register used to access the stack is called **SP** (stack pointer) register.

- The stack pointer in the 8051 is only 8 bits wide, which means that it can take value 00 to FFH. When 8051 powered up, the SP register contains value 07.



Timer :





Mode 0

Timer Clock

TLX (5 bits)

THX (8 bits)

TFx

Overflow flag

Mode 1

Timer Clock

TLX

THX

TFx

Overflow flag

Mode 2

Timer Clock

TLX

Reload

THX

TFx

Overflow flag

Mode 3

Timer Clock

TL1

TH1

Timer Clock

TL0

TF0

Overflow flag

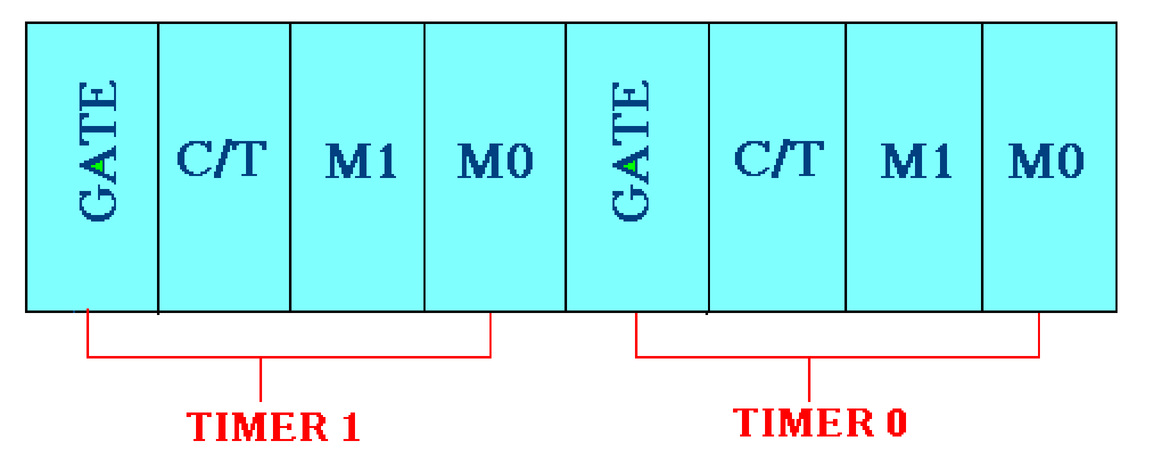
12 F_{0sc}

TH0

TF1

Overflow flag

TMOD Register:

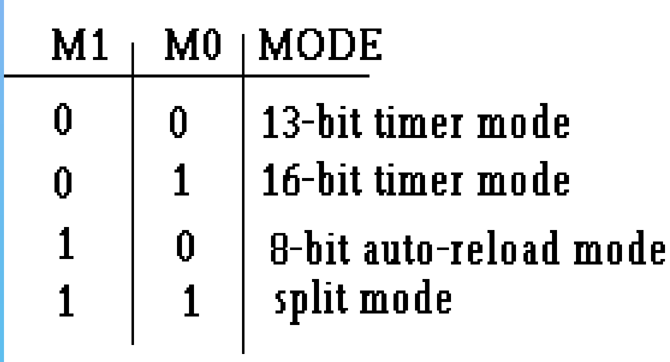


GATE	C/T	M1	M0	GATE	C/T	M1	M0

TIMER 1

TIMER 0

- **Gate :** When set, timer only runs while INT(0,1) is high.
- **C/T :** Counter/Timer select bit.
- **M1 :** Mode bit 1.
- **M0 :** Mode bit 0.



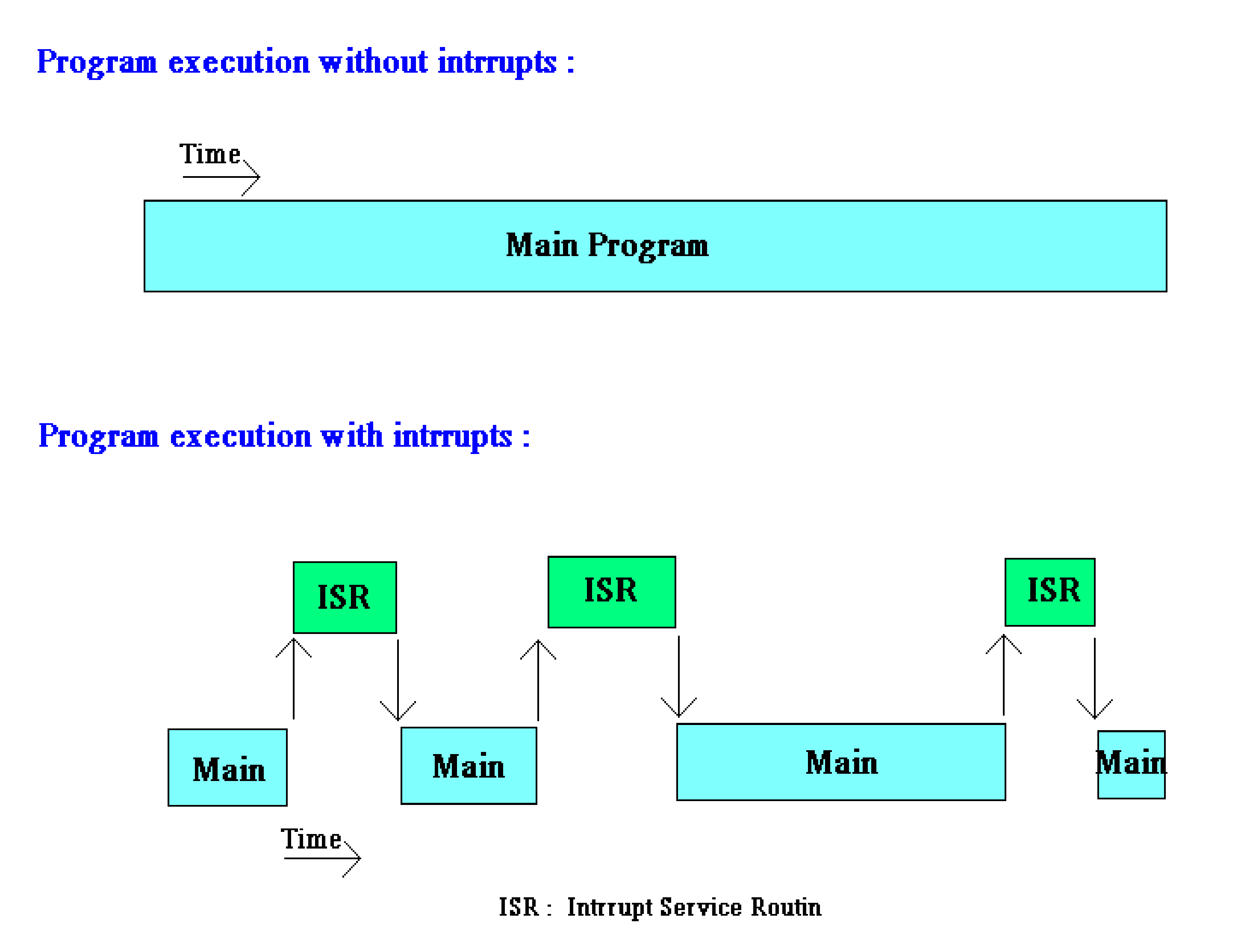
M1	M0	MODE
0	0	13-bit timer mode
0	1	16-bit timer mode
1	0	8-bit auto-reload mode
1	1	split mode

TCON Register:

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

- TF1: Timer 1 overflow flag.
- TR1: Timer 1 run control bit.
- TF0: Timer 0 overflag.
- TR0: Timer 0 run control bit.
- IE1: External interrupt 1 edge flag.
- IT1: External interrupt 1 type flag.
- IE0: External interrupt 0 edge flag.
- IT0: External interrupt 0 type flag.

Interrupt :

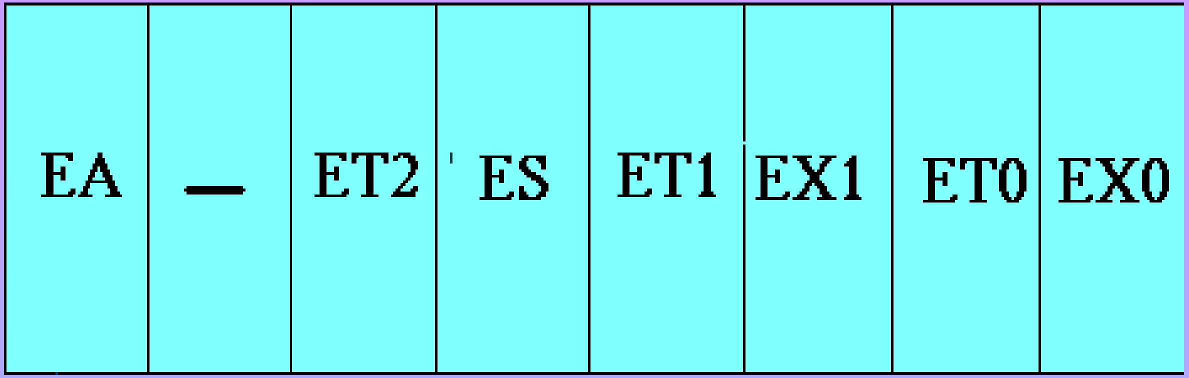


Program execution without intrrupts :

Program execution with intrrupts :

ISR : Intrrupt Service Routin

Interrupt Enable Register :



EA	—	ET2	ES	ET1	EX1	ET0	EX0

- EA : **Global enable/disable.**
- **---** **: Undefined.**
- ET2 :Enable Timer 2 interrupt.
- ES :Enable Serial port interrupt.
- ET1 :Enable Timer 1 interrupt.
- EX1 :Enable External 1 interrupt.
- ET0 : Enable Timer 0 interrupt.
- EX0 : Enable External 0 interrupt.

MICROCONTROLLER INSTRUCTION SET

- **Subject: MICROCONTROLLER**

PRESENTED BY

Prof.Mahesh Yanagimath
Dept of EEE

Introduction

- An instruction is an order or command given to a processor by a computer program. All commands are known as instruction set and set of instructions is known as program.
- 8051 have in total 111 instructions, i.e. 111 different words available for program writing.

Instruction Format

- irst part describes WHAT should be done, while other explains HOW to do it.
- The latter part can be a data (binary number) or the address at which the data is stored.
- Depending upon the number of bytes required to represent 1 instruction completely.

Types Of Instructions

- Instructions are divided into 3 types;

1. One/single byte instruction.
2. Two/double byte instruction.
3. Three/triple byte instruction.

Types Of Instructions

1. One/single byte instructions :

- If operand is not given in the instruction or there is no digits present with instruction, the instructions can be completely represented in one byte opcode.

- OPCODE 8 bit

Types Of Instructions

2. Two/double byte instruction:

- If 8 bit number is given as operand in the instruction, the such instructions can be completed represented in two bytes.

First byte	OPCODE
Second byte	8 bit data or I/O port

Types Of Instructions

3. Three/triple byte instruction:

- If 16 bit number is given as operand in the instructions than such instructions can be completely represented in three bytes 16 bit number specified may be data or address.

Types Of Instructions

1. First byte will be instruction code.
2. Second byte will be 8 LSB’s of 16 bit number.
3. Third byte will be 8 MSB’s of 16 bit number.

First byte	OPCODE.
Second byte	8 LSB’s of data/address.
Third byte	8 MSB’S of data/address.

Addressing Modes

- Addressing modes specifies where the data (operand) is. They specify the source or destination of data (operand) in several different ways, depending upon the situation.

- Addressing modes are used to know where the operand located is.

Addressing Modes

- There are 5 types of addressing modes:

1. Register addressing.
2. Direct addressing.
3. Register indirect addressing.
4. Immediate addressing.
5. Index addressing.

Register Addressing Mode

- In register addressing mode; the source and/or destination is a register.

- In this case; data is placed in any of the 8 registers(R0-R7); in instructions it is specified with letter Rn (where N indicates 0 to 7).

Register Addressing Mode

- For example;

1. ADD A, Rn (This is general instruction).

2. ADD A, R5 (This instruction will add the contents of register R5 with the accumulator contents).

Direct Addressing Mode

- In direct addressing mode; the address of memory location containing data to be read is specified in instruction.

- In this case; address of the data is given with the instruction itself.

Direct Addressing Mode

- For example;

1. MOV A, 25H (This instruction will read/move the data from internal RAM address 25H and store it in the accumulator.

Register Indirect Addressing Mode

- In register indirect addressing mode; the contents of the designated register are used as a pointer to memory.

- In this case; data is placed in memory, but address of memory location is not given directly with instruction.

Register Indirect Addressing Mode

- For example;

1. MOV A,@Ro This instruction moves the data from the register whose address is in the Ro register into the accumulator.

Immediate Addressing Mode

- In immediate addressing mode, the data is given with the instruction itself.
- In this case; the data to be stored in memory immediately follows the opcode.

Immediate Addressing Mode

- For example;

1. MOV A, #25H (This instruction will move the data 25H to accumulator.

Index Addressing Mode

- Offset (from accumulator) is added to the base index register(DPTR OR Program Counter) to form the effective address of the memory location.

- In this case; this mode is made for reading tables in the program memory.

Index Addressing Mode

- For example;

1. MOVC A, @ A + DPTR (This instruction moves the data from the memory to accumulator; whose address is computed by adding the contents of accumulator and DPTR)

Types Of Instructions

1. Data transfer instructions.
2. Arithmetic instructions.
3. Logical instructions.
4. Logical instructions with bits.
5. Branch instructions.

Data Transfer Instructions

- These instructions move the content of one register to another one.

- Data can be transferred to stack with the help of PUSH and POP instructions.

Data Transfer Instructions

- **MNEMONIC DESCRIPTION BYTES**
- MOV A,Rn (A) → (Rn) 1

←

- MOV A,Rx (A) (Rx) 2

←

- MOV A,@Ri (A) (Ri) 1

←

Data Transfer Instructions

- MOV A,#X (A) ← Data 2
- MOV Rn,A (Rn) ← (A) 1
- MOV Rn, Rx (Rn) ← (Rx) 2

Data Transfer Instructions

- MOV Rn, #X (Rn) ← Data 2
- MOV Rx, A (Rx) ← (A) 2
- MOV Rx, Rn (Rx) ← (Rn) 2

Data Transfer Instructions

MOV Rx, Ry	(RX) ← (Ry)	3
MOV Rx, @ Ri	(Rx) ← (Ri)	2
MOV Rx, # X	(Rx) ← Data	3

Data Transfer Instructions

MOV @ Ri, A	(Ri) ← (A)	1
MOV @ Ri, Rx	(Ri) ← (Rx)	2
MOV @ Ri, #X	(Ri) ← Data	2

Data Transfer Instructions

- MOV DPTR, #X (DPTR) ← Data 3
- MOVC A @ A+DPTR (A) ← (A+DPTR) 1
- MOVC A@ A+PC (A) ← (A+PC) 1

Data Transfer Instructions

- MOVX A,@ Ri A (Ri) ← 1
- MOVX A, @ (A) (DPTR) 1
  ←

  DPTR
- MOVX @Ri, A (Ri) (A) 1
  ←

Data Transfer Instructions

- MOVX @ (DPTR) (A) 1

←

DPTR, A

- PUSH Rx Push directly addressed Rx register on stack 2

- POP Rx (A) (Rx) 2

←

Data Transfer Instructions

- XCH A, Rn (A) ↔ (Rn) 1
- XCH A, Rx (A) ↔ (Rx) 2
- XCH A, @Ri (A) ↔ (Ri) 1

Data Transfer Instructions

- XCHD Exchange 4 lower 1 bits in accumulator with indirectly addressed register

Arithmetic Instructions

- These instructions perform several basic operations. After execution, the result is stored in the first operand.

- 8 bit addition, subtraction, multiplication, increment-decrement instructions can be performed.

Arithmetic Instructions

MNEMONICS	**DESCRIPTION**	**BYTE**
ADD A, Rn	A = A + Rn	1
ADD A, Rx	A = A + Rx	2
AAD A, @ Ri	A = A+ Ri	1

Arithmetic Instructions

ADD A, # X	A = A + Byte	2
ADDC A, Rn	A = A + Rn + C	1
ADDC A , Rx	A = A + Rx + C	2

Arithmetic Instructions

ADDC A, @ Ri	A = A + Ri + C	1
ADDC A, # X	A = A + Byte + C	2
SUBB A, Rn	A = A – Rn – 1	1

Arithmetic Instructions

SUBB A, Rx	A = A – Rx – 1	2
SUBB A, @ Ri	A = A – Ri – 1	1
SUBB A, # X	A = A – Byte – 1	2

Arithmetic Instructions

INC A	A = A + 1	1
INC Rn	Rn = Rn + 1	1
INC Rx	Rx = Rx + 1	2

Arithmetic Instructions

INC @ Ri	Ri = Ri + 1	1
DEC A	A = A – 1	1
DEC Rn	Rn = Rn – 1	1

Arithmetic Instructions

DEC Rx	Rx = Rx – 1	2
DEC @ Ri	Ri = Ri – 1	1
INC DPTR	DPTR = DPTR + 1	1

Arithmetic Instructions

- MUL AB B:A = A * B 1
- DIV AB A = [A/B] 1
- DA A Decimal adjustment of 1
  accumulator according to BCD code

Logical Instructions

- These instructions perform logical operations between two register contents on bit by bit basis.

- After execution, the result is stored in the first operand.

Logical Instructions

MNEMONIC	DESCRIPTION	BYTE
ANL A, Rn	(A) (A) ^ (Rn) ←	1
ANL A, Rx	(A) (A) ^ (Rx) ←	2
ANL A,@ Ri	(A) (A) ^ (Ri) ←	1

Logical Instructions

- ANL A, # X (A) (←8 bit data) ^ (A) 2
- ANL Rx, A (Rx) (A) ^← (Rx) 2
- ANL Rx,# X (Rx) (8 ← bit data) ^ (Rx) 3

Logical Instructions

- ORL A, Rn (A) ← (A) + (Rn) 1
- ORL A, Rx (A) ← (A) + (Rx) 2
- ORL A, @ Ri (A) ← (A) + (Ri) 2

Logical Instructions

- ORL Rx, A (Rx) (A) + (Rx) 2

←

- ORL Rx,# X (Rx) (8 bit data) + (Rx) 2

←

- XORL A, Rn Logical exclusive 1 OR operation between the contents of accumulator and R register.

Logical Instructions

- XORL A, Rx Logical exclusive OR 2 operation between the contents of the accumulator and directly addressed register Rx.
- XORL A,@ Ri Logical exclusive OR 1 operation between the contents of the accumulator and directly addressed register.

Logical Instructions

- XORL A, # X Logical exclusive OR 2 operation between the contents of accumulator and the given 8 bit data.
- XORL Rx, A Logical exclusive OR 2 operation between the contents of the accumulator and directly addressed register Rx.

Logical Instructions

- XORL Rx, # X Logical exclusive OR 3 operation between the contents of the directly addressed register Rx and the given 8 bit data.
- CLR A (A) 0 1
- CPL A (A) (/A)← 1

←

Logical Instructions

- SWAP A (A3-0) (A7-4) 1

↔

- RL A (An + 1) (An) 1

(A0) (A7)←

- RLC (An + 1) (An)← 1

(A0) (C)

←

(C) (A7)

←

←

Logical Instructions

- RR A (An) (An + 1) 1

  (A7) (A0)←

  ←

- RRC A (An) (An + 1) 1

  (A7) (C)←

  (C) (A0)←

  ←

Logical Instructions On Bits

- Similar to logical instructions, these instructions also perform logical operations.

- The difference is that these operations are performed on single bits.

Logical Instructions On Bits

MNEMONIC	**DESCRIPTION**	**BYTE**
CLR C	(C = 0)	1
CLR bit	clear directly addressed bit	2
SETB C	(C = 1)	1

Logical Instructions On Bits

- SETB bit addressed bit Set directly 2
- CPL C (1 = 0, 0 = 1) 1
- CPL bit addressed bit Complement directly 2

Logical Instructions On Bits

- ANL C, bit Logical AND operation 2 between Carry bit and directly addressed bit.

- ANL C,/bit Logical AND operation 2 between Carry bit and inverted directly addressed bit.

Logical Instructions On Bits

- ORL C, bit Logical OR operation 2 between Carry bit and directly addressed bit.

- ORL C,/bit Logical OR operation 2 between Carry bit and inverted directly addressed bit.

Logical Instructions On Bits

- MOV C, bit Move directly addressed 2 bit to carry bit.

- MOV bit, C Move Carry bit to directly 2 addressed bit.

Program Flow Control Instructions

- In this group, instructions are related to the flow of the program, these are used to control the operation like, JUMP and CALL instructions.

- Some instructions are used to introduce delay in the program, to the halt program.

Program Flow Control Instructions

MNEMONIC	DESCRIPTION	BYTE
ACALL adr11	(PC) ← (PC) + 2	2
	(SP) ← (SP) + 1	
	((SP)) ← (PC7 – 0)	
	(SP) ← (SP) + 1	
	((SP)) ← (PC15-8)	

Program Flow Control Instructions

- LCALL adr16 $(PC) \leftarrow (PC) + 3$ 3

$(SP) \leftarrow (SP) + 1$

$((SP)) \leftarrow (PC_{7-0})$

$(SP) \leftarrow (SP) + 1$

$((SP)) \leftarrow (PC_{15-8})$

$(PC) \leftarrow addr_{15-0}$

Program Flow Control Instructions

- RET (PC15-8) ((←SP)) 1

  (SP) (S←P) – 1

  (PC7-0) ((←SP))

  (SP) (←SP) - 1

Program Flow Control Instructions

- RET1 (PC15-8) ((SP)) 1

  (SP) (SP) ← 1

  (PC7-0) ((SP)) ←

  (SP) (SP) ← 1

  ←

- AJMP addr11 (PC) (PC) + 2 1

  (PC10-0) page address ←

  ←

Program Flow Control Instructions

- LJMP addr16 (PC) ← addr15-0 3
- SJMP rel short jump from 2
  (from -128 to +127 locations in
  relation to first next instruction)

Program Flow Control Instructions

- JC rel (PC) (PC) + 2 2

  IF (C) = 1 ←

  THEN (PC) (PC) + rel

  ←

- JNC rel (PC) (PC) + 2 2

  IF (C) = 0 ←

  THEN (PC) (PC) + rel

  ←

Program Flow Control Instructions

JB bit, rel	Jump if addressed bit is set. Short jump.	3
JBC bit, rel	Jump if addressed bit is set and clear it. Short jump.	3

Program Flow Control Instructions

- JMP @A + DPTR (PC) ← (A) + (DPTR) 1
- JZ rel (PC) ← (PC) + 2 2

  IF (A) = 0

  THEN (PC) ← (PC) + rel

Program Flow Control Instructions

- JNZ rel (PC) (P←C) + 2 2

  IF (A) = 0/

  THEN (PC) (←PC) + rel

- CJNE A, Rx, rel Compare the contents 3 of acc. And directly addressed register Rx. Jump if they are different. Short jump.

Program Flow Control Instructions

- CJNE A, #X, rel (PC) (PC) + 3 3

  IF (A) < > data ←

  THEN (PC) (PC) + relative

  ← offset

  IF (A) < data

  THEN (C) 1

  ELSE (C) 0 ←

  ←

Program Flow Control Instructions

- CJNE @ RI, # x, rel (PC) (PC) + 3 3

```
        IF (Rn) <> data←
   THEN (PC)      (PC) + relative
                     ←offset
        IF (Rn)  <  data
    THEN ( C )      1
     ELSE ( C )     0      ←

                           ←
```

Program Flow Control Instructions

- CJNE @ Ri, # X, rel (PC) (PC) + 3 3

IF ((Ri)) <> data ←

THEN (PC) (PC) + relative

← offset

IF ((Ri)) < data

THEN (C) 1

ELSE (C) 0 ←

←

Program Flow Control Instructions

- DJNZ Rn , rel $\quad$ (PC) ← (PC) + 2 $\quad$ 2

  (Rn) ← (Rn) - 1

  IF (Rn) > 0 or (Rn) < 0

  THEN (PC) ← (PC) + rel

Program Flow Control Instructions

- DJNZ Rx, rel (PC) ← (PC) + 2 3

  (Rx) ← (Rn) – 1

  IF (Rx) > 0 or (Rx) < 0

  THEN (PC) ← (PC) + rel

- NOP No operation 1

Summary

- Instruction set.
- Addressing modes.
- Data transfer instruction.
- Arithmetic instruction.
- Logical instruction.
- Logical operation on bits.

Chapter 3
8051 Microcontroller

Objectives

- Understand the 8051 Architecture
- Use SFR in C
- Use I/O ports in C

3.1 Overview

- The Intel 8051 is a very popular general purpose microcontroller widely used for small scale embedded systems. Many vendors such as Atmel, Philips, and Texas Instruments produce MCS-51 family microcontroller chips.
- The 8051 is an 8-bit microcontroller with 8 bit data bus and 16-bit address bus. The 16 bit address bus can address a 64K(2^{16}) byte code memory space and a separate 64K byte of data memory space. The 8051 has 4K on-chip read only code memory and 128 bytes of internal Random Access Memory (RAM)

Contd.

- Besides internal RAM, the 8051 has various *Special Function Registers* (SFR) such as the Accumulator, the B register, and many other control registers.
- 34 8-bit general purpose registers in total.

The ALU performs one 8-bit operation at a time.

- Two 16 bit /Counter timers
- 3 internal interrupts (one serial), 2 external interrupts.
- 4 8-bit I/O ports (3 of them are dual purposed). One of them used for serial port,

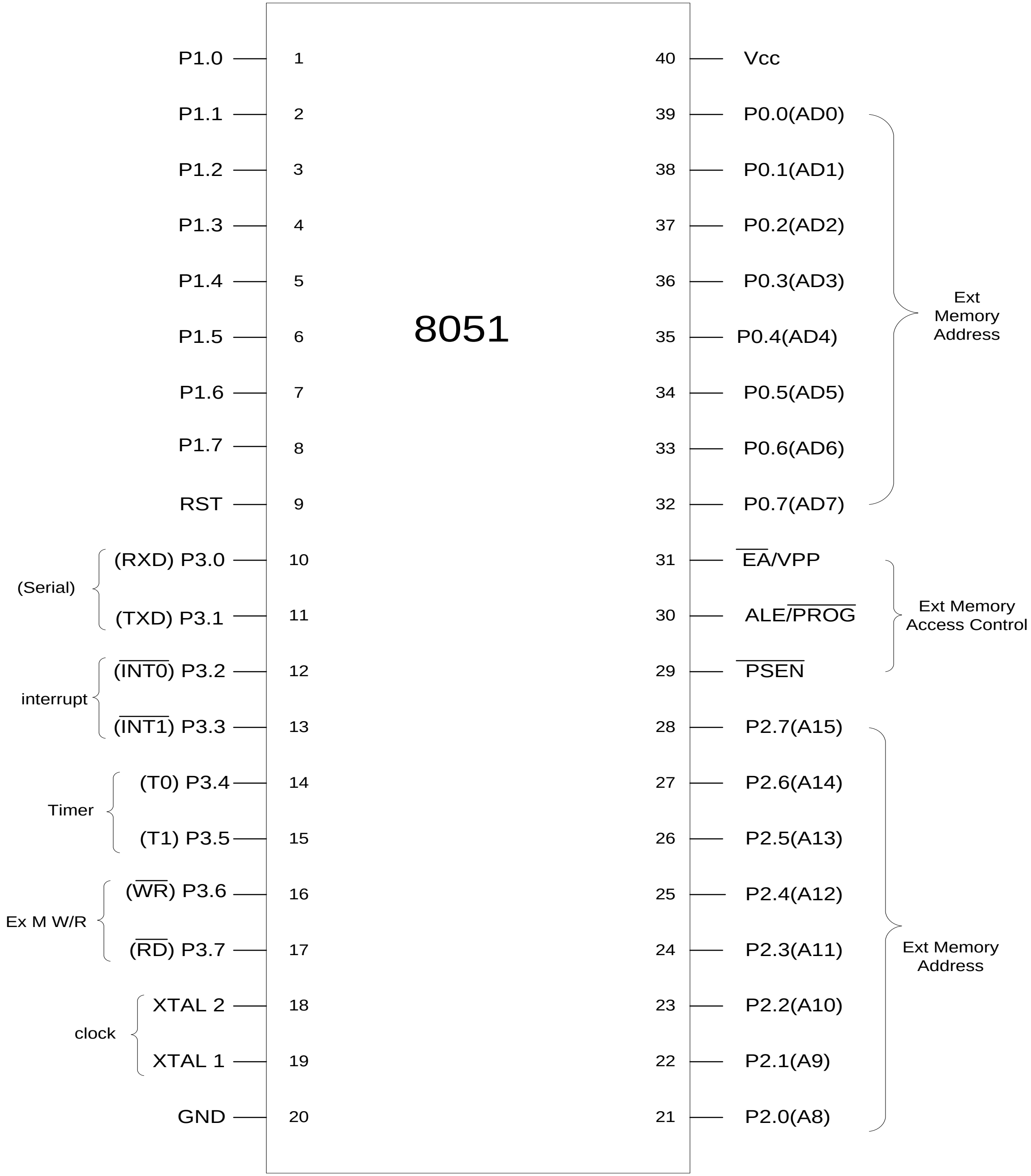
Some 8051 chips come with UART for serial communication and ADC for analog to digital conversion.

3.1.1 8051 Chip Pins

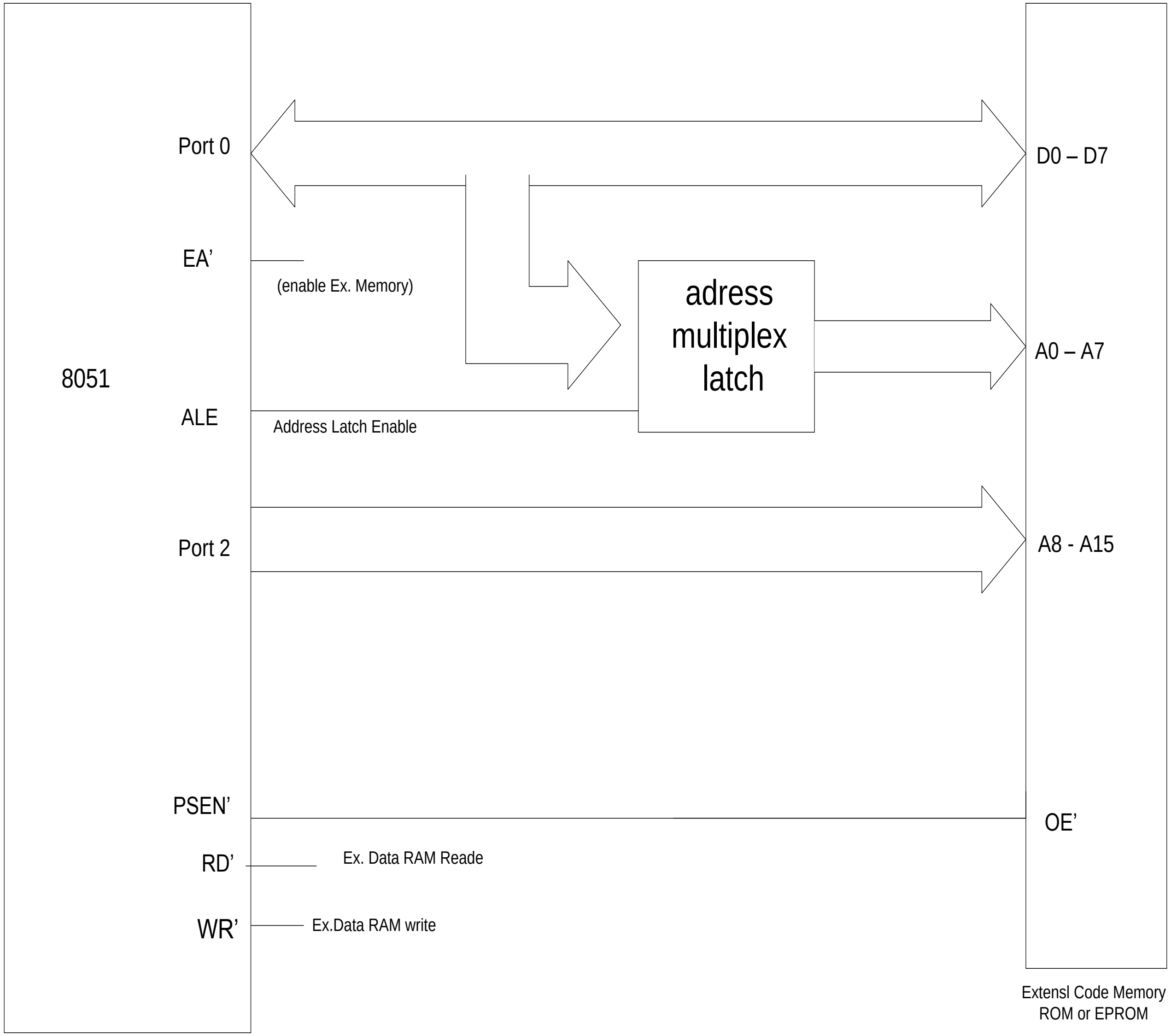
40 pins on the 8051 chip.

Most of these pins are used to connect to I/O devices or external data and code memory.

- 4 I/O port take 32 pins(4 x 8 bits) plus a pair of XTALS pins for crystal clock
- A pair of Vcc and GND pins for power supply (the 8051 chip needs +5V 500mA to function properly)
- A pair of timer pins for timing controls, a group of pins (EA, ALE, PSEN, WR, RD) for internal and external data and code memory access controls
- One Reset pin for reboot purpose



Pin out Diagram of the 8051 Microcontroller



The Pin Connection for External Code and Data Memory

- The EA' (External Access) pin is used to control the internal or external memory access.

  The signal 0 is for external memory access and signal 1 for internal memory access.
- The PSEN' (Program Store Enable) is for reading external code memory when it is low (0) and EA is also 0.
- The ALE (Address Latch Enable) activates the port 0 joined with port 2 to provide 16 bit external address bus to access the external memory. The ALE multiplexes the P0:

  1 for latching address on P0 as A0-A7 in the 16 bit address buss, 0 for latching P0 as data I/O.
- P0.x is named ADx because P0 is multiplexed for Address bus and Data bus at different clock time.

  WR' only provides the signal to write external data memory

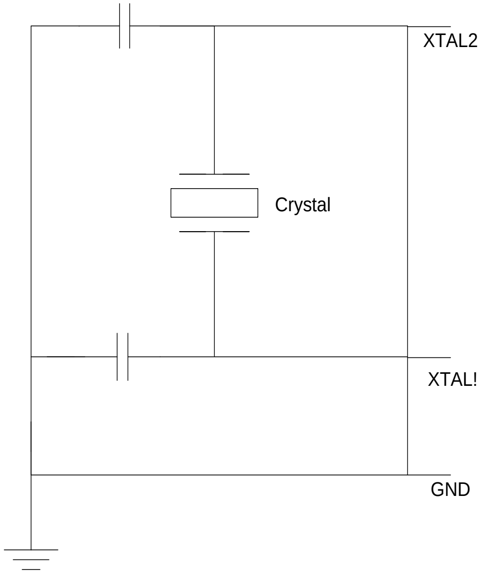
  RD' provides the signal to read external data and code memory.

3.1.2. System Clock and Oscillator Circuits

- The 8051 requires an external oscillator circuit. The oscillator circuit usually runs around 12MHz. the crystal generates 12M pulses in one second. The pulse is used to synchronize the system operation in a controlled pace..
- A machine cycle is minimum amount time a simplest machine instruction must take
- An 8051 machine cycle consists of 12 crystal pulses (clock cycle).
- instruction with a memory oprand so that it needs multiple memory accesses.

Contd.

The first 6 crystal pulses (clock cycle) is used to fetch the opcode and the second 6 pulses are used to perform the operation on the operands in the ALU. This gives an effective machine cycle rate at 1MIPS (Million Instructions Per Second).



Crystal to 8051 XTAL 1/2

3.1.3. 8051 Internal Architecture

- The CPU has many important registers. The Program Count (PC) always holds the code memory location of next instruction.
- The CPU is the heart of any computer which is in charge of computer operations.
- It fetches instructions from the code memory into the instruction Register (IR),

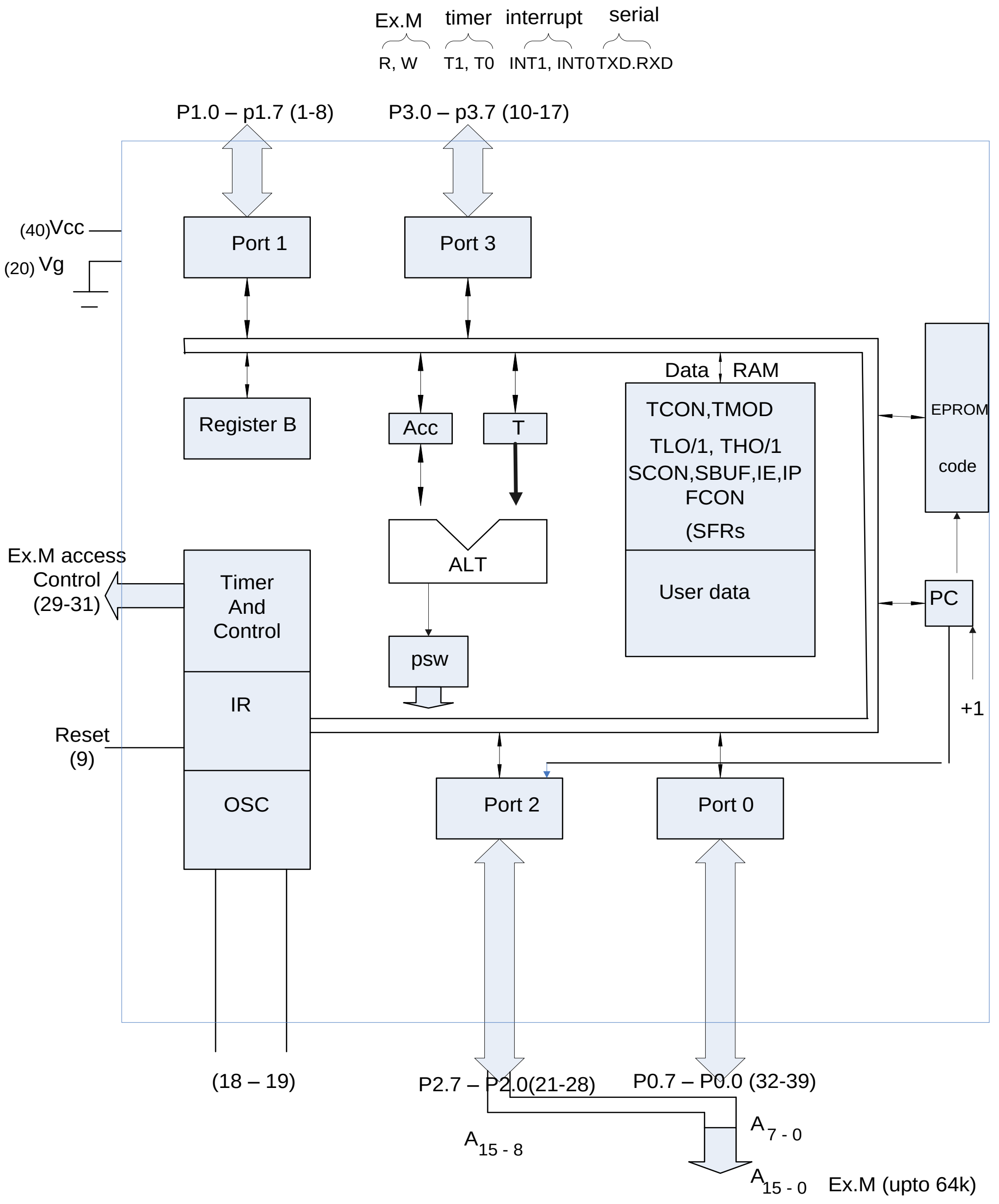
  analyzes the opcode of the instruction, updates the PC to the location of next instruction,

  fetches the oprand from the data memory if necessary, and finally performs the operation in the Arithmetic-Logic Unit (ALU) within the CPU.

Contd.

- The B register is a register just for multiplication and division operation which requires more register spaces for the product of multiplication and the quotient and the remainder for the division.
- The immediate result is stored in the accumulator register (Acc) for next operation
- and the Program Status Word (PSW) is updated depending on the status of the operation result

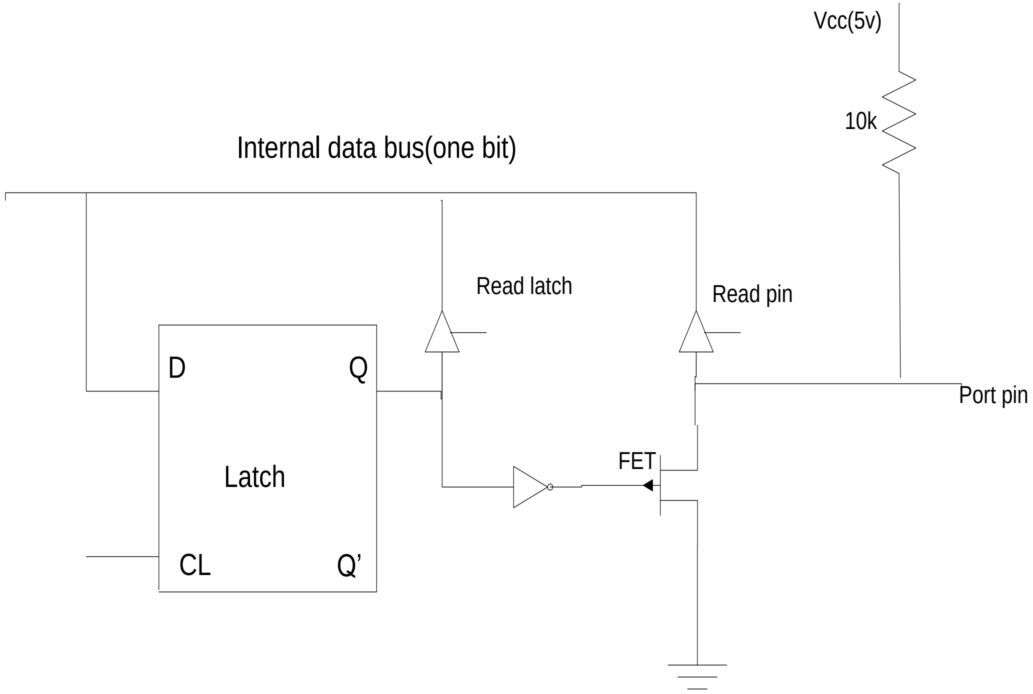
8051 Internal Architecture



Simplified 8051 block diagram

3.2 Ports
3.2.1. Port Reading and Writing

There are 4 8-bit ports: P0, P1, P2 and P3. All of them are dual purpose ports except P1 which is only used for I/O. The following diagram shows a single bit in an 8051 I/O port.



Single Bit In I/O Port

Contd.

- When a C program writes a one byte value to a port or a single bit value to a bit of a port, just simply assign the value to the port as follows:

P1 = 0x12; or P1^2=1;

- P1 represents the 8 bits of port 1 and P1^2 is the pin #2 of the port 1 of 8051 defined in the reg51.h of C51, a C dedicated for 8051 family.
- When data is written to the port pin, it first appears on the latch input (D) and is then passed through to the output (Q) and through an inverter to the Field Effect Transistor (FET).

Contd.

- If you write a logic 0 to the port pin, this Q is logic 0 which is inverted to logic 1 and turns on the FET gate. It makes the port pin connected to ground (logic 0).
- If you write a logic 1 is written to the port pin, then Q is 1 which is inverted to a logic 0 and turns off the FET gate. Therefore the pin is at logic 1 because it is connected to high.
- You can see the written data is stored in the D latch after the data is written to the port pin.

Contd.

- However, you must initialize the port for reading before reading.
- If the latch was logic 0, then you will always get 0 regardless the data in the port pin because it is grounded through the FET gate.
- Therefore, in order to read the correct data from a port or a port pin, the last written logic (stored in the latch D) must be 0XFF(8 bits) or 1(single bit). E.g., you read entire P1 port or single bit of P1 port in this way:

```
unsigned char x; bit y;
P1 = 0xFF;   //port reading initialization
x = P1;        //read port
y = P1^2;     //read bit
```

3.2.2. The Port Alternate Functions

- **PORT P1 (Pins 1 to 8)**: The port P1 is a port dedicated for general I/O purpose. The other ports P0, P2 and P3 have dual roles in addition to their basic I/O function.
- **PORT P0 (pins 32 to 39):** When the external memory access is required then Port P0 is multiplexed for address bus and data bus that can be used to access external memory in conjunction with port P2. P0 acts as A0-A7 in address bus and D0-D7 for port data. It can be used for general purpose I/O if no external memory presents.
- **PORT P2 (pins 21 to 28)**: Similar to P0, the port P2 can also play a role (A8-A15) in the address bus in conjunction with PORT P0 to access external memory.

Contd.

- **PORT P3 (Pins 10 to 17)**:

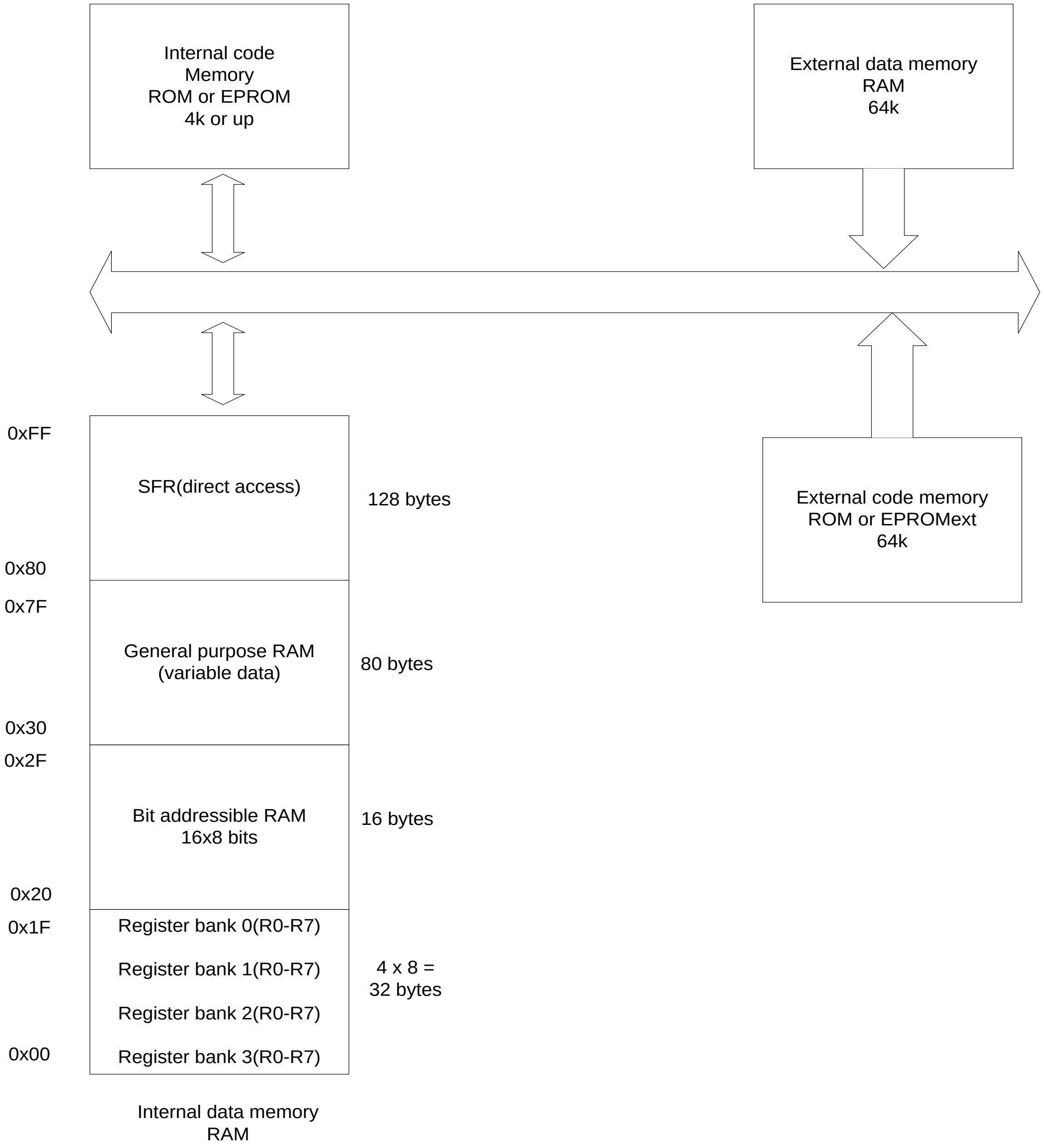
  In addition to acting as a normal I/O port,
- P3.0 can be used for serial receive input pin(RXD)
- P3.1 can be used for serial transmit output pin(TXD) in a serial port,
- P3.2 and P3.3 can be used as external interrupt pins(INT0’ and INT1’),
- P3.4 and P3.5 are used for external counter input pins(T0 and T1),
- P3.6 and P3.7 can be used as external data memory write and read control signal pins(WR’ and RD’)read and write pins for memory access.

3.3 Memory and SFR

3.3.1. Memory

- The 8051 code(program) memory is read-only, while the data memory is read/write accessible. The program memory(in EPROM) can be rewritten by the special programmer circuit.
- The 8051 memory is organized in a Harvard Architecture. Both the code memory space and data memory space begin at location 0x00 for internal or external memory which is different from the Princeton Architecture where code and data share same memory space.
- The advantage of the Harvard Architecture is not only doubling the memory capacity of the microcontroller with same number of address lines but also increases the reliability of the microcontroller, since there are no instructions to write to the code memory which is read only.

Separate read instructions for external data and code memory.



Contd.

- In this model, the data memory and code memory use separate maps by a special control line called Program Select Enable (PSEN’).
- This line (i.e. when PSEN’ = 0) is used to indicate that the 16 address lines are being used to address the code memory.
- When this line is ‘1’, the 16 address lines are being used to address the data memory.

Contd.

- The 8051 has 256 bytes of internal addressable RAM, although only first 128 bytes are available for general use by the programmer.
- The first 128 bytes of RAM (from 0x00 to 0x7F) are called the direct memory, and can be used to store data.
- The lowest 32 bytes of RAM are reserved for 4 general register banks. The 8051 has 4 selectable banks of 8 addressable 8-bit registers, R0 to R7.

Contd.

- This means that there are essentially 32 available general purpose registers, although only 8 (one bank) can be directly accessed at a time.
- The advantage of using these register banks is time saving on the context switch for interrupted program to store and recover the status.
- Otherwise the push and pop stack operations are needed to save the current state and to recover it after the interrupt is over.
- The default bank is bank 0.
- The second 128 bytes are used to store Special Function Registers (SFR) that C51 program can configure and control the ports, timer, interrupts, serial communication, and other tasks.

3.3.2. Special Function Registers (SFRs)

- The SFR is the upper area of addressable memory, from address 0x80 to 0xFF. This area consists of a series of memory-mapped ports and registers.
- All port input and output can therefore be performed by get and set operations on SFR port name such as P3.
- Also, different status registers are mapped into the SFR for checking the status of the 8051, and changing some operational parameters of the 8051.
- All 8051 CPU registers, I/O ports, timers and other architecture components are accessible in 8051 C through SFRs
- They are accessed in normal internal RAM (080H – 0FFH) by 8051 C, and they all are defined in the header file *reg51.h* listed below.

Contd.

- There are 21 SFRs.
- In addition to I/O ports, the most frequently used SFRs to control and configure 8051 operations are:
    - TCON (Timer CONtrol)
    - TMOD (Timer MODe)
    - TH0/TH1 and TL0/TL1 (Timer's high and low bytes)
    - SCON (Serial port CONtrol)
    - IP (Interrupt Priority)
    - IE (Interrupt Enable)
- Almost all 8051 C embedded programs include the reg51.h.

Contd.

- /*---
- REG51.H
- Header file for generic 80C51 and 80C31 microcontroller.
- Copyright (c) 1988-2001 Keil Elektronik GmbH and Keil Software, Inc.
- All rights reserved.
- ---*/
- /* BYTE Register */
- sfr P0   = 0x80;
- sfr P1   = 0x90;
- sfr P2   = 0xA0;
- sfr P3   = 0xB0;
- sfr PSW = 0xD0;
- sfr ACC = 0xE0;
- sfr B   = 0xF0;
- sfr SP   = 0x81;
- sfr DPL = 0x82;
- sfr DPH = 0x83;
- sfr PCON = 0x87;
- sfr TCON = 0x88;
- sfr TMOD = 0x89;
- sfr TL0 = 0x8A;

Contd.

- sfr TL1 = 0x8B;
- sfr TH0 = 0x8C;
- sfr TH1 = 0x8D;
- sfr IE = 0xA8;
- sfr IP = 0xB8;
- sfr SCON = 0x98;
- sfr SBUF = 0x99;
- /* BIT Register */
- /* PSW */
- sbit CY = 0xD7;
- sbit AC = 0xD6;
- sbit F0 = 0xD5;
- sbit RS1 = 0xD4;
- sbit RS0 = 0xD3;
- sbit OV = 0xD2;
- sbit P = 0xD0;
- /* TCON */
- sbit TF1 = 0x8F;
- sbit TR1 = 0x8E;
- sbit TF0 = 0x8D;
- sbit TR0 = 0x8C;

Contd.

- sbit IE1 = 0x8B;
- sbit IT1 = 0x8A;
- sbit IE0 = 0x89;
- sbit IT0 = 0x88; /* IE   */
- sbit EA   = 0xAF;
- sbit ES   = 0xAC;
- sbit ET1 = 0xAB;
- sbit EX1 = 0xAA;
- sbit ET0 = 0xA9;
- sbit EX0 = 0xA8;
- /* IP   */
- sbit PS   = 0xBC;
- sbit PT1 = 0xBB;
- sbit PX1 = 0xBA;
- sbit PT0 = 0xB9;
- sbit PX0 = 0xB8;
- /* P3 */
- sbit RD   = 0xB7;
- sbit WR   = 0xB6;
- sbit T1   = 0xB5;
- sbit T0   = 0xB4;
- sbit INT1 = 0xB3;

Contd.

- sbit INT0 = 0xB2;
- sbit TXD = 0xB1;
- sbit RXD = 0xB0;
- /* SCON */
- sbit SM0 = 0x9F;
- sbit SM1 = 0x9E;
- sbit SM2 = 0x9D;
- sbit REN = 0x9C;
- sbit TB8 = 0x9B;
- sbit RB8 = 0x9A;
- sbit TI   = 0x99;
- sbit RI   = 0x98;

- The sbit register variables of these SFRs defined in reg51.h often used in embedded C program.
- 1. TCON (Timer/Counter Control Register) SFR for timer control

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0 (88H)
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

- TF0/TF1: Timer0/1 overflow flag is set when the timer counter overflows, reset by program
- TR0/TR1: Timer0/1 run control bit is set to start, reset to stop the timer0/1
- IE0/IE1: External interrupt 0/1 edge detected flag1 is set when a falling edge interrupt on the external port 0/1, reset(cleared) by hardware itself for falling edge transition-activated INT; Reset by code for low level INT.

  IT0/IT1 External interrupt type (1: falling edge triggered, 0 low level triggered)

2. IE (Interrupt Enable Register) SFR used for interrupt control

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0 (A8H)
EA		ET2	ES	ET1	EX1	ET0	EX0

- EX0/EX1 : (1/0) Enables/disables the external interrupt 0 and the external interrupt 1 on port P3.2 and P3.3
- ET0/ET1 : (1/0) Enables/disables the Timer0 and Timer1 interrupt via TF0/1
- ES : (1/0) Enables/disables the serial port interrupt for sending and receiving data
- EA : (1/0) Enables/disables all interrupts

3. IP (Interrupt Priority Register) SFR used for IP setting

- PX0/1: External interrupt 0/1 priority level
- PT0/1/2: Timer0, Timer1, Timer2(8052) interrupt priority level
- PS: Serial port interrupt priority level

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
		PT2	PS	PT1	PX1	PT0	PX0

4. <u>PSW (Program Status Word) SFR for CPU status</u>

- P: parity check flag
- OV: ALU overflow flag
- RS0/RS1: Register bank specification mode
- 00: bank 0 (00H-07H); 01: bank1; 10: bank 2; 11: bank 3(18H-1FH)
- F0:   User defined lag
- CY: ALU carry out
- AC: ALU auxiliary carry out

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
CY	AC	F0	RS1	RS0	OV		P

5. P3(Port 3) SFR used for I/O and other special purposes

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
RD	WR	T1	T0	INT1	INT0	TxD	RxD

- Addition to I/O usage, P3 can also be used for:
- RXD/TXD: Receive/Transmit serial data for RS232
- INT0, INT1: External interrupt port inputs
- T0,T1: Alternative Timer 0/1 bit
- WR/RD : Write/Read control bits used for external memory
- If external RAM or EPROM is used, ports P0 and P2 are used to address the external memory.
- Other port SFRs such as P0, P1, P2 are mainly used for data I/O.

6. TL0/TL1 SFRs: Lower byte of Timer 0/1, used to set timer interrupt period

TH0/TH1 SFRs: Higher byte of Timer 0,used to set timer interrupt period

7. TMOD (Timer Mode Register) SFR(not bit addressable)

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Gate	C/T	M1	M0	Gate	C/T	M1	M0

Note: bit 0-3 for Timer0 and bit 4-7 for Timer1

Gate Control:

- 0= Timer enabled(normal mode)
- 1 = if INT0/INT1 is high, the timer is enabled to count the number of pulses in the external interrupt ports (P3.2 and P3.3)

C/T Counter/Timer Selector

- 0 = count internal clock pulse (count once per machine cycle = oscillator clock/12)
- 1 = count external pulses on P3.4 (Timer 0) and P3.5(Timer 1)

- Working as a “Timer”, the timer is incremented by one every machine cycle. A machine cycle consists of 12 oscillator periods, so the count rate is 1/12 of the oscillator frequency.
- Working as a “Counter”, the counter is incremented in response to a falling edge transition in the external input pins.
- The external input is sampled once every machine cycle. A “high” sample followed by a low sample is counted once.
- Timer 0 and Timer 1 have four operating modes.

M1, M0		Mode Control	
0	0	(Mode 0)	13 bit count mode
0	1	(Mode 1)	16 bit count mode
1	0	(Mode 2)	Auto reload mode
1	1	(Mode 3)	Multiple mode

Contd.

- Note: Mode 0-2 are same for both Timer0 and timer1 but mode 3 is not
- The Timer0 has two SFRs called TL0 and TH0 and the Timer1 has TL1 and TH1 respectively.
- TL0/1 are used to store the low byte and TH0/1 are used to store the high byte of the number being counted by the timer/counter.
- In mode 0, only TH0/1 is used as an 8-bit Counter. The timer will count from the init value in the TH0/1 to 255, and then overflows back to 0.
- If interrupt is enable (ET0/1 = 1) then an overflow interrupt is triggered at this time which will set TF0/1 to 1.
- If used as a timer its rate equal to the oscillator rate divided by (12x32)
- If used as a counter, the counting rate equals to the oscillator rate divided by 32.

Contd.

- Mode 1 is the same as Mode 0, except that the Timer runs with both 16 bits of TH and TL registers together and it will count to 65535 and then overflow back to 0..
- If used as a timer its rate equals to the oscillator rate divided by 12.
- If used as a counter, the max counting rate equals to the oscillator rate divided by 24.
- Mode 2 configures the Timer register as an 8-bit Counter (TL0/1) with automatic reload from TH0/1 after overflow. Overflow from TL0/1 not only sets TF1, but also reloads TL0/1 with the preset value of TH0/1 automatically.
- Mode 3 is not very popular one so we skip it.

- C51 timer/counter configuration example

  //0X52 = 01010010_2 enable timer 0 in mode 2,

  //counter 1 in mode 1

  TMOD = 0X52;

- Here we set the Timer/couter1 as a counter in mode 1 with 0101_2 and set the Timer/counter0 as a timer in mode 2 with 0010_2.
- The counter in mode 1 counts the input pulses up to 65,535 and then overflows back to 0.
- If the T1(P3.5) pin is connected to an encoder which produces one pulse each revolution of a motor, then we can use TH1 and TL1 to calculate total input pulses in the port pin P3.5 by TH1*256 + TL1 in a specified period of time which is controlled by the timer0. In this way, we can conclude how fast the motor is running.

Contd.

- The timer 0 is set in mode 2 which is an auto reload mode. You can set TH0 and TH1 to control the time out period for calculation the rotation rate of the motor.
- After time out from timer 0, the TH1 and TL1 must be cleared to 0 to start over the pulse counting again.

- Example produces a 25 ms timeout delay by timer1. 25,000 machine clocks take 25ms, because one machine cycle = 1 µs in 12 MHZ crystal oscillator 8051.

```
//Clear all T1 control bits in TMOD.
TMOD &=  0x0F;
//set T1 in mode 1 and leave T0 unchanged
TMOD  |=   0X10;
ET1 = 0;               //don't need interrupt
TH =  0X9E;              //0X9E = 158
TL =  0X62;
          //0X62 = 98, 158 x 256 + 98 = 40536
          // 65536 – 25000 = 40536
TF1 = 0;        //reset timer 1 overflow flag
TR1 =1;                // start timer1
// The loop will go 25ms until the timer 1 //overflow flag is set to 1
while ( TF1 != 1);
TF1 =0;                //reset TF1
```

- You can not use any bit symbol in TMOD because it is not bit addressable. You must use bit-wise operation to set TMOD.

8. PCON (Power Control Register) SFR (Not bit addreesible)

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
SMOD	--	--	--	GF1	GF2	PD	IDL

- SMOD(serial mode) 1= high baud rate, 0 = low baud rate
- GF1, GF2 flags for free use
- PD: 1= power down mode for CMOS
- IDL: 1= idle mode.
- Ex. PCON |= 0x01;

  // to set the IDL bit 1 to force the CPU in a power save mode

  // the |operator is a shorthand bit wise logical OR operator
- The Acc, B, DPH, DPL, SP SFRs are only accessible by assembly languages

9. SCON (Serial Port Control Register) SFR

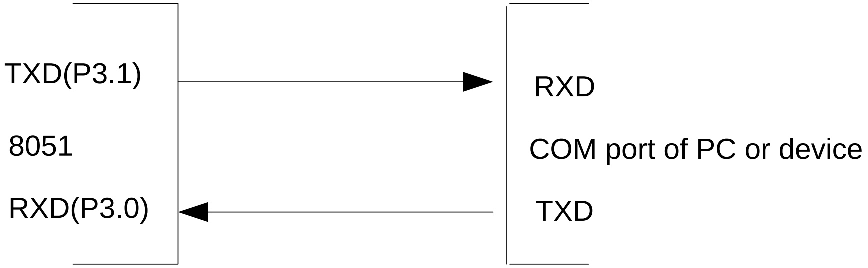
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
SM0	SM1	SM2	REN	TB8	RB8	TI	RI

- REN: Receiver enable is set/reset by program
- TB8: stores transmitted bit 8(9th bit, the stop bit)
- RB8: Stores received bit 8(9th bit, the stop bit)
- TI: Transmit Interrupt is set at the end of 8th bit (mode 0)/ at the stop bit (other modes) indicating the completion of one byte transmission, reset by program
- RI: Receive Interrupt is set at the end of 8th bit (mode 0)/at the stop bit (other modes) indicating the completion of one byte receiving, reset by program

- RI and TI flag in SCON SFR are used to detect the interrupt events.
- If RI = 1 then a byte is received at the RxD pin. If TI = 1 then a byte is transmitted from the TxD pin.

SM0	SM1	Serial Mode	Baud Rate	Device
0	0	0 (Sync.) half duplex,	Oscillator/12 (fixed)	8-bit shift register
0	1	1(Async) full duplex	Set by Timer 1	8-bit UART
1	0	2(Sync) half duplex	Oscillator/64 (fixed)	9-bit UART
1	1	3(Async) full duplex	Set by Timer 1	9-bit UART

We focus on mode 0 and mode1 because mode 2 and mode 3 are not often used.



- The built-in Universal Asynchronous Receiver/Transmitter (UART) integrated circuit can support serial full duplex asynchronous communications over a computer or peripheral device.
- In mode 0, 8051 TxD plays a role of synchronous clock and RxD is used for both receiving and transmitting data so that the mode 0 is a half duplex synchronous serial working mode.

  The frequency of TxD clock is 1MHz (1/12 of 12MHz) and the cycle period is 1 µs.
- The mode 1 works with UART without synchronous clock. it is an asynchronous full duplex serial communication mode. There are start and stop bits surrounding a byte data during the transmission.
- The baud rate of serial communication is measured by bps(bits/sec). The typical standard serial baud rates are 1200, 2400, 9600, 19200 bps.

- The baud rate is determined by timer 1 overflow rate of timer 1. By default, the baud rate = 1/32 of overflow rate of timer 1(if SMOD of PCON is 0).
- How to configure Timer 1 to get the desired overflow rate? Just set the Timer 1 in its auto reload mode 2.
- The loaded value of TH1 is determined by this formula for the 12MHz 8051 :

  TH1 = 256 – 1000000 / (32* (desired baud))

  For example, in order to get the 19200 bps

  TH1 = 256 – 1000000/32/19200 = 256 -2= 254 = 0xFE

  TH1 = 253 -> baud rate 9600 bps

  TH1 = 243 -> baud rate 2400 bps

  TH1 = 230 -> baud rate 1200 bps

- If you set SMOD bit of the PCON SPR you can double the baud rate. For example, if SMOD=1 then baud rate =19200 bps if TH1=253.
- The SBUF SFR is a serial buffer data register to store the received or transmitting data in byte. The SBUF usage is shown below.

```
char c;
c= 0X41;
SBUF = c;    // send ‘A’ to serial output line
c = SBUF;   //get a char from serial line
```

3.4 SFRs and Interrupts

- Interrupt is an internal or external event that suspends a program and transfers the control to an event handler or ISR to handle the event.
- After the service is over the control is back to the suspended program to resume the execution, The microcontroller in a embedded system connects many devices and need to handle service requests from devices all the time.

You can classify Registers for function

1. Enable interrupts

    IE SFR enables interrupts individually and globally

    EX0/EX1: Enable external interrupt INT0/INT1

    ET0/ET1: Enable Timer 0/Timer1 interrupt

    ES: Enable serial interrupt

    EA: Enable global interrupt

    Set 1 in a bit to enable its interrupt, e.g. EA =1;

    reset 0 to masks that interrupt, e.g., EA = 0;

2. Interrupt Flags

The interrupt flags are set to 1 when the interrupts occur.

IE0/IE1 in TCON - For External Interrupts

TF0/TF1 in TCON - For Timer Interrupts

TI/RI in SCON - For Serial Interrupts

The flag 1 indicates the interrupt occurrence and the flag 0 indicates no interrupt.

3. Interrupt Priority

There are two types of interrupt priority:

User Defined Priority and Automatic Priority

User Defined Priority

The IP register is used to define priority levels by users. The high priority interrupt can preempt the low priority interrupt. There are only two levels of interrupt priority.

```
//The external interrupt INT0 at port P3.2 is assigned a high priority.
EX0 = 1;
//the external interrupt INT1 at port P3.3 is assigned  a low priority.
EX1  = 0;
```

Automatic Priority

In each priority level, a priority is given in order of:

INT0, TF0, INT1, TF1, SI.

For example, if two external interrupts are set at same priority level, then INT0 has precedence over INT1.

1) External Interrupts

- An external interrupt is triggered by a low level or negative edge on INT0 and INT1 which depends on the external interrupt type setting.

  Set up an external interrupt type by IT0 and IT1 of TCON SFR.

- E.g.,

  IT0 = 1; //set INT0 as Negative edge triggered

  IT1 = 0; // set INT1 as Level Triggered

- The external interrupt source may be a sensor, ADC, or a switch connected to port P3.2(INT0) or P3.3(INT1). You use IE0/IE1 to test the external interrupt events: Ex. If IE0 = 1 then the INT0 interrupt takes place.
- Note that if an external interrupt is set to low level trigger, the interrupt will reoccur as long as P3.2 or P3.3 is low that makes the code difficult to manage.

- You enable external interrupt by EX0/EX1 of IE SFR. E.g.,

```
EA = 1;
EX0 = 1;  //enable external interrupt INT0
```

- If the interrupt is level activated, then IE0/1 flag has to be cleared by user software as

```
EX1 = 0;
```

- You don’t need to reset the edge triggered external interrupt.
- This fragment makes the INT1 external interrupt ready on port P3.3 pin:

```
EA =1;
EX1 =1;
IT1 =1;
```

2) Timer/Counter Interrupts

- This unit can be used as a counter to count external pulses on P3.4 and P3.5 pins or it can be used to count the pulses produced by the crystal oscillator of the microcontroller.
- Timer Interrupt is caused by Timer 0/ Timer1 overflow.

  TF0/1: 1 => Condition occurred

  Enabled using IE

  ET0/1 = 1, EA = 1
- E.g.

```
TMOD = 0X12;   //set timer 1 in mode 1, timer 0 in mode 2.
EA=1;          // enable global interrupt
TH1=16;         // Make timer 1 overflow every 240 clocks //240=256-16
TL1=16;        // Make timer 1 overflow after 240 clocks(240 µs)
ET0=1;         // enable timer 0
TR0=1;          //start timer0
               // Timer 0 overflows after 65535 clocks.
ET1=1;         // enable timer 1
TR1=1;         //start timer 1
```

3)Serial Interrupts

- Serial communication with Universal Asynchronous Receive Transmit (UART) protocol transmits or receives the bits of a byte one after the other in a timed sequence on a single wire.
- It is used to communicate any serial port of devices and computers.
- The serial interrupt is caused by completion of a serial byte transmitting or receiving.
- The transmit data pin (TxD) is at P3.1 and the receive data pin (RxD) is at P3.0.
- All communication modes are controlled through SCON, a non bit addressable SFR. The SCON bits are defined as SM0, SM1, SM2, REN, TB8, RB8, TI, RI.

- You use timers to control the baud of asynchronous serial communication which is set by TMOD and TCON as we discussed before
- full duplex asynchronous serial communication in mode 1 to transmit and receive data simultaneously

```
#include <reg51.h>
main()
    {
    char c;
// set timer1 in auto reload 8 bit timer mode 2
TMOD=0x20;

    // load timer 1 to generate baud rate of 19200 bps

 TH1 = 0xFD;
 TL1 = 0XFD;
```

```
// set serial communication in mode 1
   SCON=0x40;
// start timer 1

 TR1 = 1;
 While(1){
// enable reception


REN = 1;
//wait until data is received
  while((SCON & 0X02) == 0);
          // same as while(RI==0);
// reset receive flag
RI=0;
// read a byte from RXD
c = SBUF;
// disable reception

   REN = 0;
// write a byte to TXD

   SBUF = c;
// wait until data transmitted
while(SCON & 0X01TI==0);
             //same as  while(TI==0);

// reset transmission flag
TI=0;
}
```

Summary

- This chapter explores the internal architecture of 8051 microcontroller and its interface. The Harvard architecture for separation of code and data memory is discussed. The detail and usage of 8051 SFRs, especially, the timer control SFRs such as TMOD, TCON and serial control SFR such as SCON, the interrupt control registers such as IE. The configurations of external interrupt, timer/counter interrupt and serial communication interrupt with C51
- Understand how to use 8051 to connect and control external devices such as sensor, switch, LED, LCD, keypad.
- Learn how to design and handle the timing clock issue in the embedded systems.
- This chapter is a foundation for the next chapter on the embedded system software development and programming in C.

Serial Communication

Prepared by

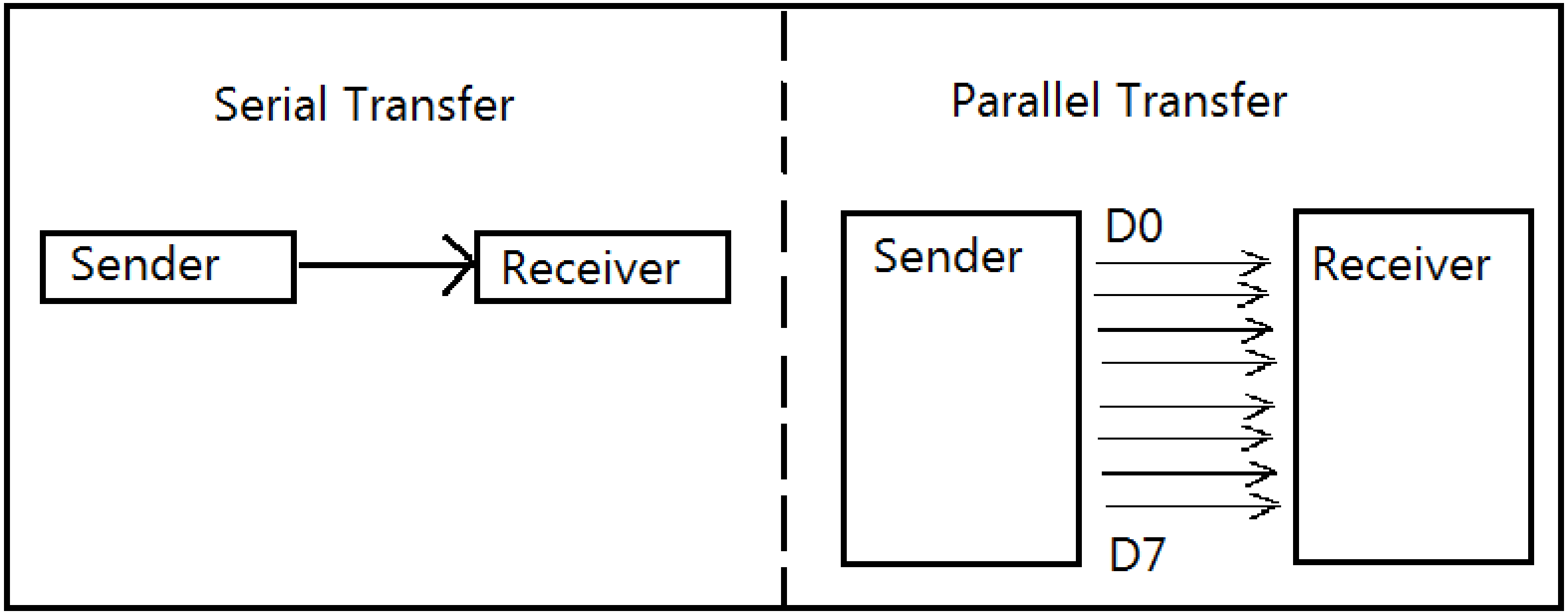
Prof.Mahesh Yanagimath

Serial Communications

Objectives

- Introduce the RS232 standard and position it within the crowded field of serial communications standards.
- Configure the 8051 serial port.
- Read and write to the serial port.
- Introduce software and hardware handshaking.

Basics of serial communication



Serial versus Parallel Data Transfer

6.1 Introduction

There are several popular types of serial communications. Here are a few worth noting:

- RS232. Peer-to-peer (i.e. communications between two devices)
- RS485. Multi-point (i.e. communications between two or more devices)
- USB (Universal Serial Bus). Replaced RS232 on desktop computers.
- CAN (Controller Area Network). Multi-point. Popular in the automotive industry.
- SPI (Serial Peripheral Interface). Developed by Motorola. Synchronous master/slave communications.
- I2C (Inter-Integrated Circuit).Developed by Philips. Multi-master communications.

- The Silicon Laboratories 8051 development kit used in this book supports RS232, SPI and I2C communications. An RS232 serial port is included on most 8051 microcontrollers. It is usually listed on the datasheet as UART.

- When we talk about serial communications, what do we really mean? How is the data transmitted? Serial data is transmitted between devices one bit at a time using agreed upon electrical signals. In our C programs though, we read and write bytes to the serial port – not bits. To accomplish the necessary translation between bytes and bits, another piece of hardware is required – the UART.

6.2 UARTs and Transceivers

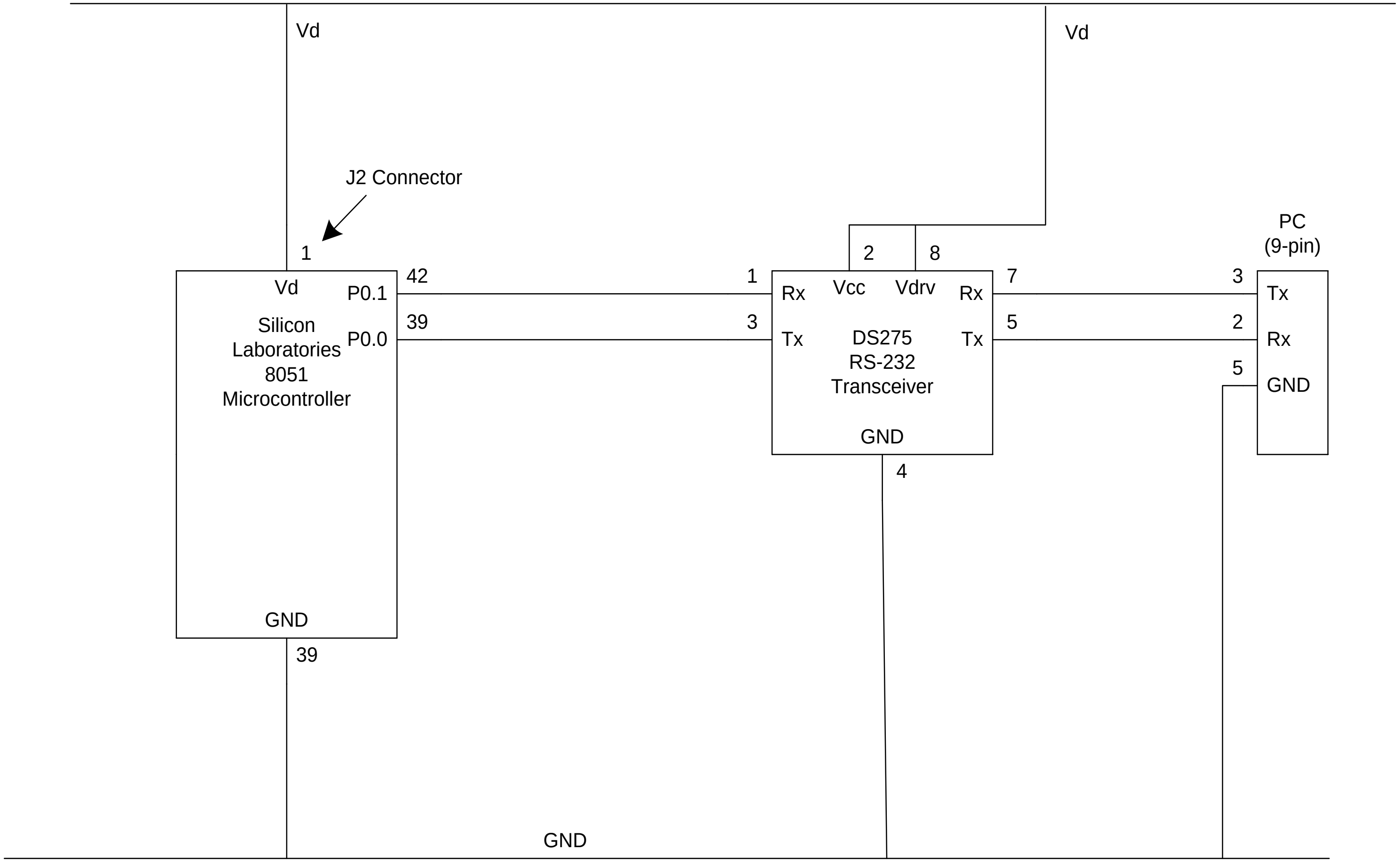
- UART (pronounced “You Art”) is an industry acronym that stands for Universal Asynchronous Receiver Transmitter. It is the interface circuitry between the microprocessor and the serial port. This circuitry is built in to the 8051 microcontroller.

- The UART is responsible for breaking apart bytes of data and transmitting it one bit at a time (i.e. serially). Likewise, the UART receives serialized bits and converts them back into bytes. In practice, it’s a little more complicated, but that’s the basic idea.

- The UART, however, doesn’t operate at the line voltages required by the RS232 standard. The UART operates at TTL voltage levels (i.e. 0 to 5V). For noise immunity and transmission length, the RS232 standard dictates the transmission of bits at a higher voltage range and different polarities (i.e. typically -9V to +9V). An external transceiver chip is needed.

- Binary 0: UART: 0V RS232: 3-25V
- Binary 1: UART: 5V RS232 -3V to -25V

8051 and DS275 RS-232 Transceiver



- UART communications is asynchronous (i.e. not synchronous). This means that there is no master clock used for timing data transfer between devices.
- The UART is also responsible for baud rate generation. This determines the speed at which data is transmitted and received. One baud is one bit per second (bps). As of this writing, data rates can reach up to 230,400 baud. The cable length between devices is limited by the baud rate -- the higher the speed, the shorter the cable. The RS-232C standard only permits transmission speeds up to 19200 baud with a cable length of 45 feet. With modern UARTs, 230,400 baud can be achieved with a short cable length of a few feet.

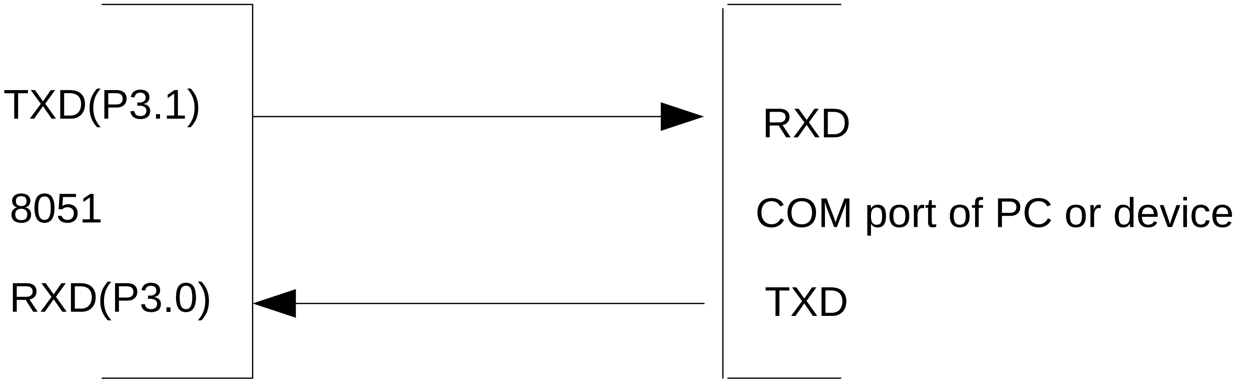
6.3 Configuring the Serial Port

- The 8051 serial port is configured and accessed using a group of SFRs (Special Function Registers).

4 UART operational modes

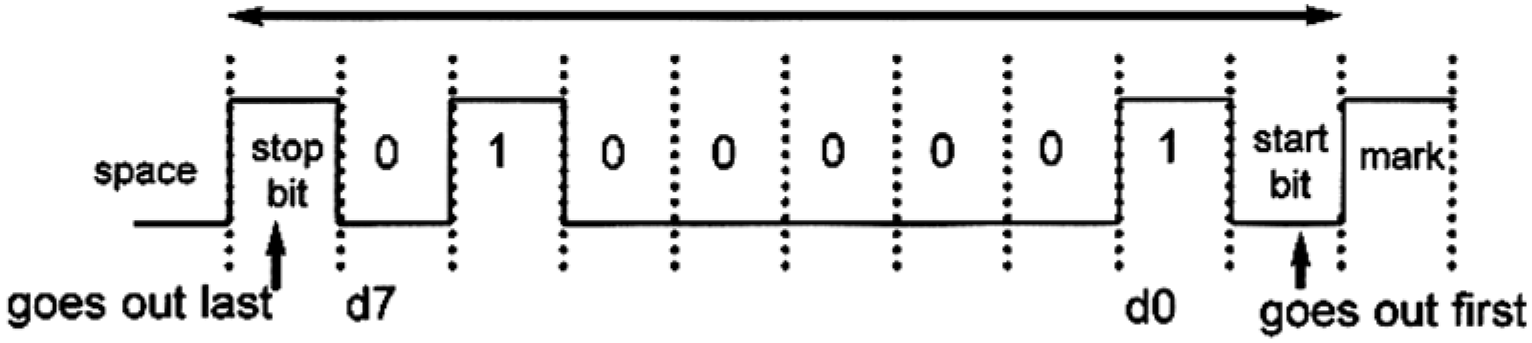
	SM0	SM1	Serial Mode	Baud Rate	Device
0	0	0	0 (Sync.) half duplex,	Oscillator/12 (fixed)	8-bit shift register
1	0	1	1(Async) full duplex	Set by Timer 1	8-bit UART
2	1	0	2(Sync) half duplex	Oscillator/64 (fixed)	9-bit UART
3	1	1	3(Async) full duplex	Set by Timer 1	9-bit UART

We focus on mode 0 and mode1 because mode 2 and mode 3 are not often used.

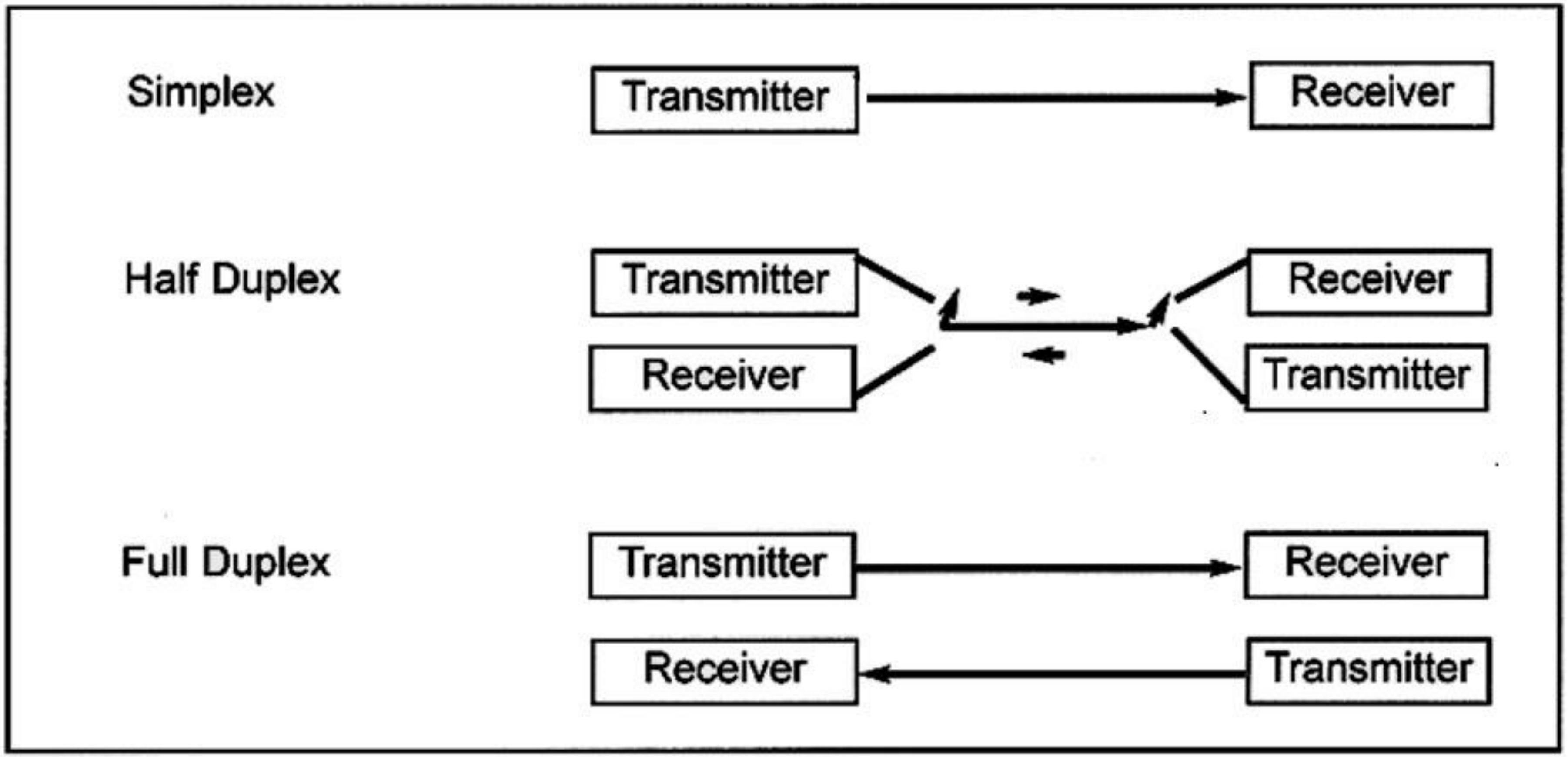


- Another job of the UART is to frame the byte of data that is serialized and transmitted. There is always one start bit (set to 0) and one stop bit (set to 1). Looking at it another way, for every byte of data, 10 bits are transmitted.

Start and stop bits



Framing ASCII "A" (41H)


Simplex
Transmitter
Receiver
Half Duplex
Transmitter
Receiver
Receiver
Transmitter
Full Duplex
Transmitter
Receiver
Receiver
Transmitter

SFRs	**Description**
SCON (Serial Port Control)	RI (Receive Interrupt). SCON.0 TI (Transmit Interrupt). SCON.1 REN (UART Receive Enable). SCON.4 SM0 and SM1 (UART Operation Mode). SCON.6, SCON.7
SBUF (Serial Data Buffer)	This is a one-byte buffer for both receive and transmit.
IE (Interrupt Enable)	ES (Enable Serial). IE.4 Set the bit to 1 to enable receive and transmit interrupts.
IP (Interrupt Priority)	PS (Priority Serial). IP.4 Set the bit to 0 for a low priority or 1 for a high priority.
UARTEN (UART Enable)	XBR0.2 (Port I/O Crossbar Register 0, Bit 2)
SMOD (Serial Port Baud Rate Doubler Enable)	PCON (Power Control Register). PCON.7 Set the bit to 1 to double the baud rate defined by serial port mode in SCON.

6.4 Setting the Baud Rate

The baud rate is a combination of factors:

- UART mode.
- The crystal frequency.
- The number of ticks required by the 8051 to complete a simple instruction. This varies from 1 to 12. For the 8051 microcontroller used in this book, the value is 1.
- The setting of the SMOD bit (i.e. normal or double baud rate).
- The reload value for the Timer.

- RS232 works in a restricted range of baud rates: 75, 110, 300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 33600, 56000, 115200 and 230400. With the UART operating in mode 1, the baud rate will be generated based on a formula using the factors listed above

$$\text{Baud rate}_{(\text{Mode1})} = (2^{\text{SMOD}^*}\text{Frequency}_{\text{osc}})/(32^*\text{Instructions}^*_{\text{cycle}}(256 - \text{TRV}))$$

- Where:
- SMOD is the normal/double baud rate bit.
- $\text{Frequency}_{\text{osc}}$ is the clock rate in hertz.
- $\text{Instructions}_{\text{cycle}}$ is the machine instruction executed each clock cycle. It is one for the 8051 microcontroller used in this book. For comparison, the original 8051 by Intel used 12 clock cycles for each instruction.
- TRV is the reload value for the timer.

Baud Summary

- Set the UART operational mode to 1. (SCON.6 = 1, SCON.7 = 0)
- Set the REN bit to enable UART receive. (SCON.4 = 1)
- Set the UART enable bit (UARTEN) in the XBR0 register. (XBR0.2 = 1)
- Set the bit for normal or double baud rate (SMOD) in the PCON register. (PCON.7 = 1 for double)
- Determine the TRV (Timer Reload Value) based on crystal frequency and desired baud rate.

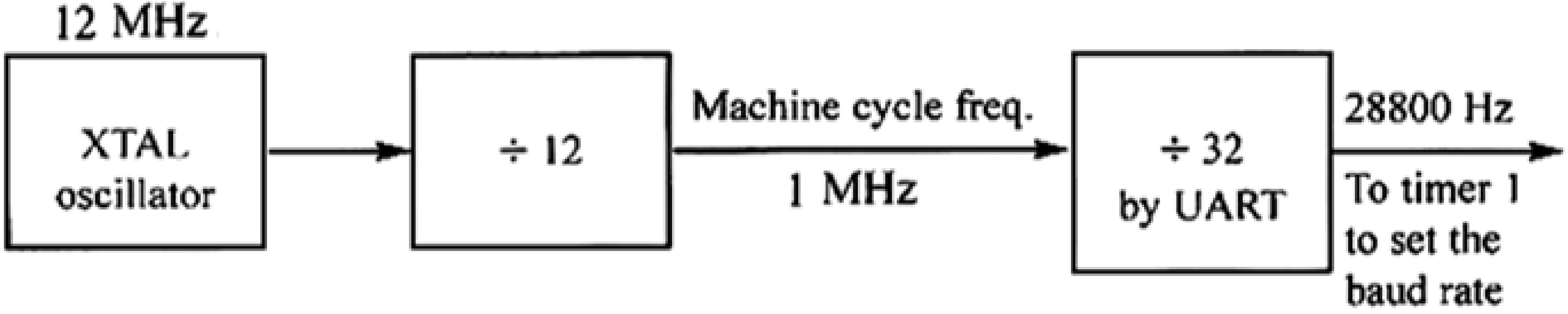
With XTAL = 12 MHz, find the TH1 value needed to have the following baud rates. (a) 9600 (b) 2400 (c) 1200

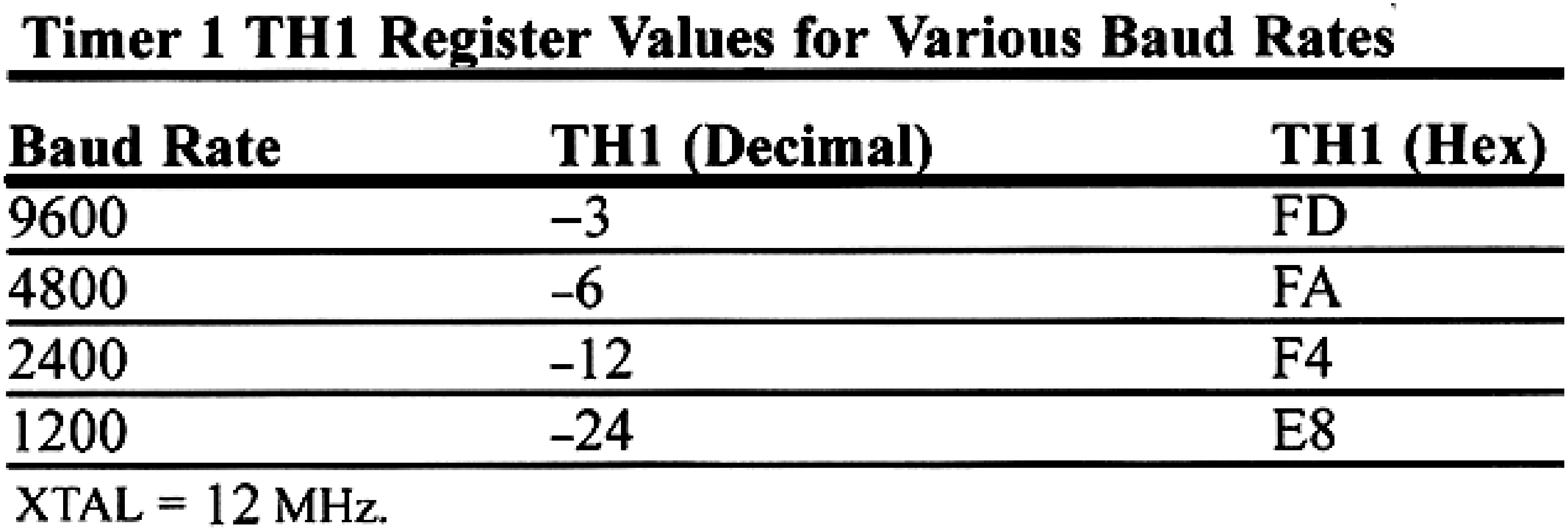
Solution:

With XTAL = 12 MHz, we have:

The machine cycle frequency of the 8051 = 12 MHz / 12 = 1 MHz, and
921.6 kHz/ 32 = 28,800 Hz is the frequency provided by UART to timer 1 to set baud rate.

(a) 28,800 / 3 = 9600 where –3 = FD (hex) is loaded into TH1
(b) 28,800 / 12 = 2400 where –12 = F4 (hex) is loaded into TH1
(c) 28,800 / 24 = 1200 where –24 = E8 (hex) is loaded into TH1





Timer 1 TH1 Register Values for Various Baud Rates

Baud Rate	TH1 (Decimal)	TH1 (Hex)
9600	–3	FD
4800	–6	FA
2400	–12	F4
1200	–24	E8

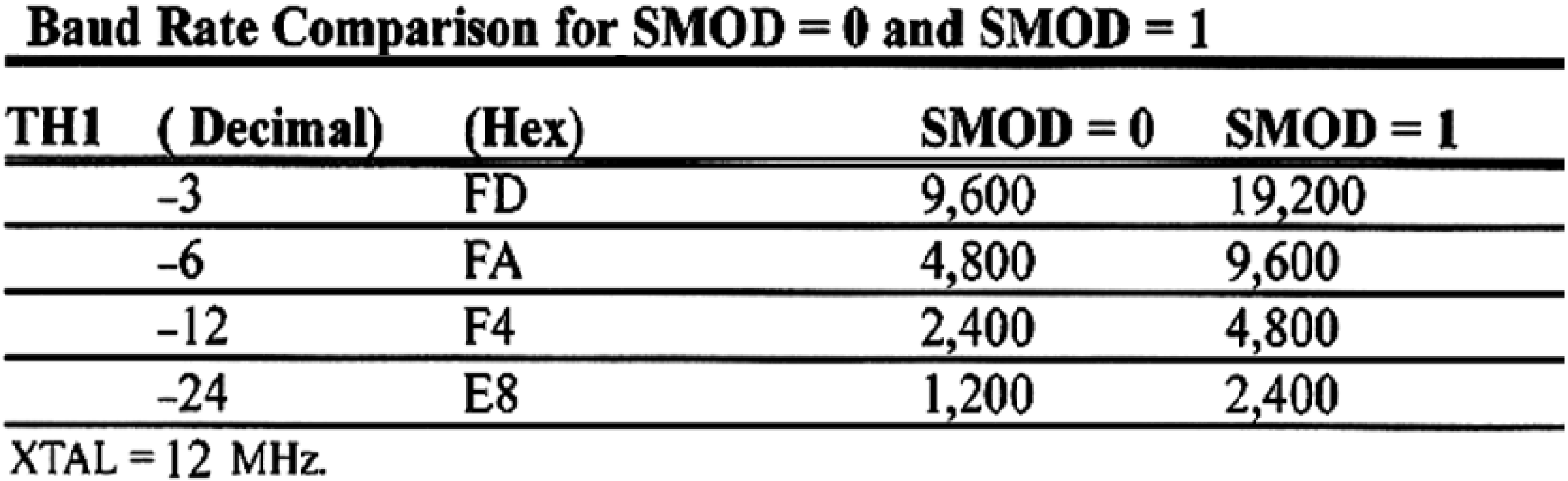
XTAL = 12 MHz.

Baud rates for SMOD=0

Machine cycle freq. = 12 MHz / 12 = 1 MHz
and
1MHz / 32 = 28,800 Hz since SMOD = 0

Baud rates for SMOD=1

Machine cycle freq. = 12 MHz / 12 = 1 MHz
and
1 MHz / 16 ≈ 57,600 Hz since SMOD = 1



Baud Rate Comparison for SMOD = 0 and SMOD = 1

TH1	(Decimal)	(Hex)	SMOD = 0	SMOD = 1
	–3	FD	9,600	19,200
	–6	FA	4,800	9,600
	–12	F4	2,400	4,800
	–24	E8	1,200	2,400

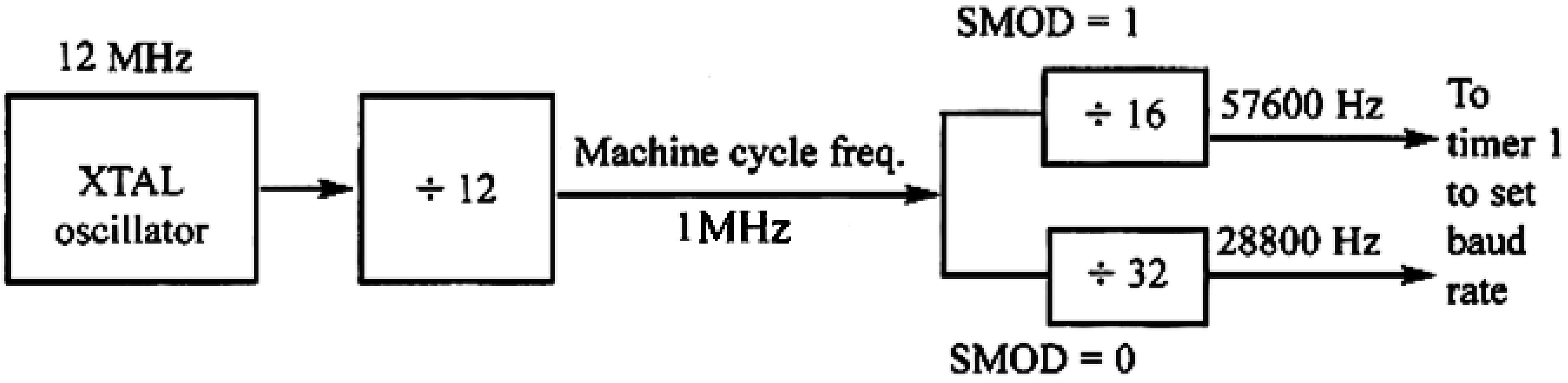
XTAL = 12 MHz.

Practice

Find the TH1 value (in both decimal and hex) to set the baud rate to each of the following.
(a) 9600 (b) 4800 if SMOD = 1 Assume that XTAL = 12 MHz

Solution:

With XTAL = 12 MHz and SMOD = 1, we have timer 1 frequency = 57,600 Hz.
(a) 57,600 / 9600 = 6; therefore, TH1 = –6 or TH1 = FAH.
(b) 57,600 / 4800 = 12; therefore, TH1 = –12 or TH1 = F4H.



Find the baud rate if TH1 = –2, SMOD = 1, and XTAL = 12 MHz.

Solution:

With XTAL = 12 MHz and SMOD = 1, we have timer 1 frequency = 57,600 Hz. The baud rate is 57,600 / 2 = 28,800.

6.5 Reading and Writing

- After all that we went through to configure the port, reading and writing bytes is easy. We simply read from and write to the SBUF register. For example:
- inByte = SBUF;   // Read a character from the UART
- SBUF = outByte; // Write a character to the UART
- The register SBUF is used for both reading and writing bytes. Internally, there are two separate registers. They are both represented as SBUF for the convenience of the programmer.

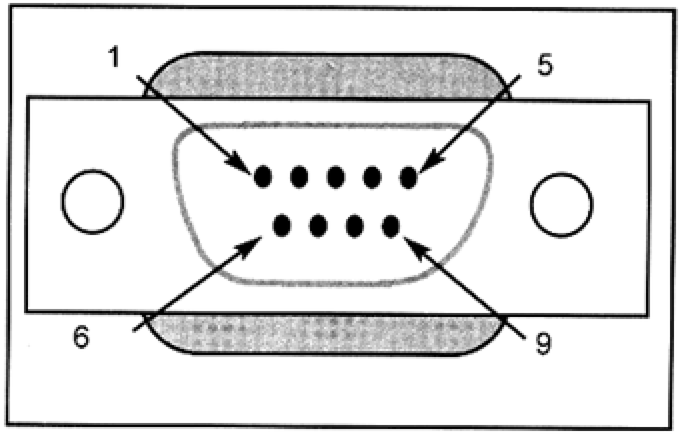
- The SBUF register (both transmit and receive) can only hold one byte. How do you know when the byte that you wrote to the port has been transmitted? Conversely, how do you know when a byte is available?
- There are ways to handle this using time delays and polling. If your application is simple enough, you may be able to get away with it.
- The best solution to the problem, however, is to use interrupts. The two interrupts we are interested in are TI (Transmit Interrupt) and RI (Receive Interrupt).

6.6 Handshaking

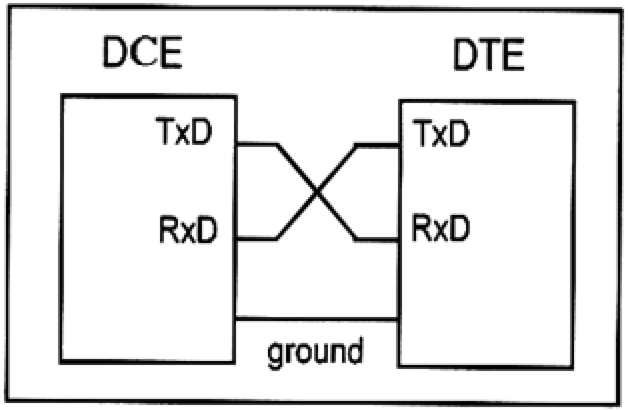
- The 8051 only has a one-byte buffer – SBUF. In contrast, a typical PC serial port with a UART with 16-byte buffer.
- If SBUF is not serviced “quickly” enough, an incoming byte may overwrite a byte that has not yet been read and processed. Using a control technique called handshaking, it is possible to get the transmitting device to stop sending bytes until the 8051 is ready.
- Likewise, the 8051 can be signaled by the receiving device to stop transmitting. There are two forms of handshaking – software and hardware.

- Software handshaking (also called XON/XOFF) uses control characters in the byte stream to signal the halting and resuming of data transmission. Control-S (ASCII 19) signals the other device to stop sending data. Control-Q (ASCII 17) signals the other device to resume sending data. The disadvantage with this approach is that the response time is slower and two characters in the ASCII character set must be reserved for handshaking use.
- Hardware handshaking uses additional I/O lines. The most common form of hardware handshaking is to use two additional control wires called RTS (Ready to Send) and CTS (Clear to Send). One line is controlled by each device. The line (either RTS or CTS) is asserted when bytes can be received and unasserted otherwise. These two handshaking lines are used to prevent buffer overruns.

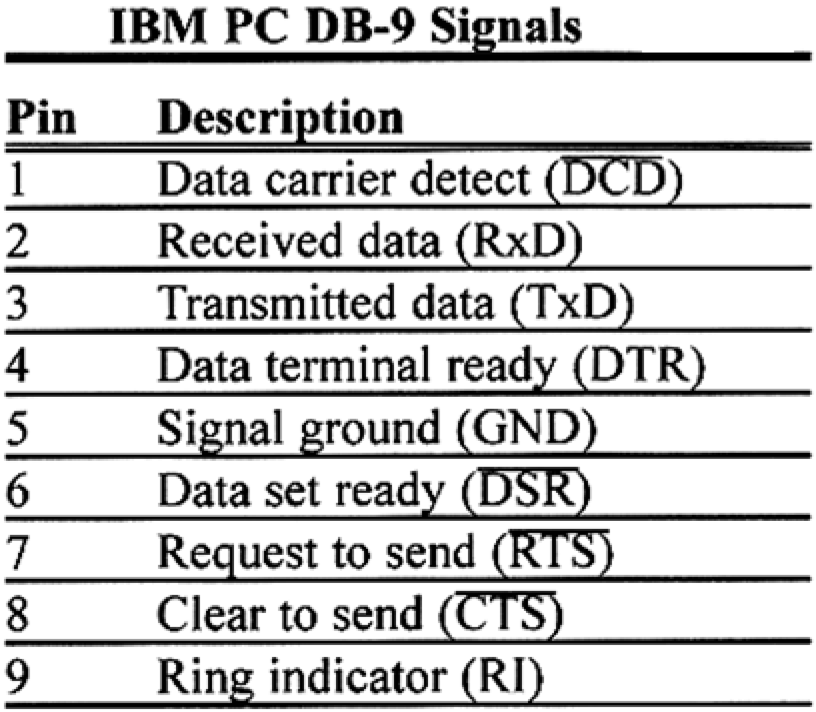
Data communication classification



DB-9 9-Pin Connector



Null Modem Connection



IBM PC DB-9 Signals

Pin	Description
1	Data carrier detect ($\overline{\text{DCD}}$)
2	Received data (RxD)
3	Transmitted data (TxD)
4	Data terminal ready (DTR)
5	Signal ground (GND)
6	Data set ready ($\overline{\text{DSR}}$)
7	Request to send ($\overline{\text{RTS}}$)
8	Clear to send ($\overline{\text{CTS}}$)
9	Ring indicator (RI)

Typically, the connector is “male” for DTE equipment and “female” for DCE equipment.

RS232 DB9 pin D-SUB male connector

- There are two other less commonly used lines – DTR (Data Terminal Ready) and DSR (Data Set Ready). These lines are typically used by devices signaling to each other that they are powered up and ready to communicate.
- To summarize, RTS/CTS are used for buffer control and DTS/DSR are used for device present and working indicators. In practice, serial communication with no handshaking uses 3 wires (TX, RX and GND). Serial communications with basic hardware handshaking uses 5 wires (TX, RX, RTS, CTS and GND).

DTE (Data Terminal Equipment) and DCE (Data Communications Equipment)

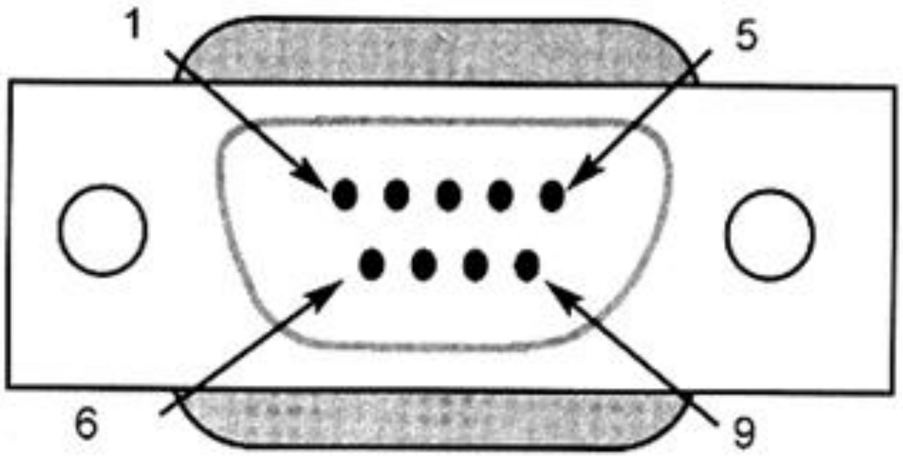
- RS232 is a point-to-point protocol meant to connect two devices together – terminals and modems. E.g., the PC is the DTE while the modem is the DCE.
- But what about other types of devices like barcode scanners and weigh scales that connect to a PC. With respect to the PC, they are all DCE devices.
- If you take the PC out of the picture, however, that may change. If you are developing an 8051 application that logs data from a weigh scale, your 8051 device will become the DTE. Knowing whether your device is DTE or DCE is important because it will determine which handshaking line to control. The DTE controls the RTS and DTR lines. In this case, point of reference is very important.

Pin	Signal Name	Direction(DTE ← DCE)
1	CD (Carrier Detect)	←
2	RXD (Receive Data)	←
3	TXD (Transmit Data)	→
4	DTR (Data Terminal Ready)	→
5	GND (System Ground)	
6	DSR (Data Set Ready)	←
7	RTS (Request To Send)	→
8	CTS (Clear To Send)	←
9	RI (Ring Indicator)	←

DB9 RS232 serial port on a PC.

- Typically, the connector is “male” for DTE equipment and “female” for DCE equipment.

RS232 DB9 pin D-SUB male connector



DB-9 9-Pin Connector

6.8 Summary

- This chapter introduced the RS232 serial communications standard and placed it in context with newer forms of serial communications. It also discussed the role of the UART and external transceiver circuits necessary to transmit bits of data at the proper voltage.
- On the software side, this chapter discussed how to configure the serial port using the special function registers and also discussed issues pertaining to baud rate generation. Finally, reading and writing to the serial port was addressed and both software and hardware handshaking concepts were introduced.

Interfacing Concepts

Prepared by

Prof.Mahesh P. Yanagimath

Introduction

- Overview of I/O operations
- Programmed I/O
  - Standard I/O
  - Memory Mapped I/O
- Device synchronization
- Readings: Scan Chapter 8

I/O operations

- Of Von Neumann's five computer building blocks, potentially the most important are the input and the output devices

- In this section we will look at the general techniques for performing I/O operations and their impact on the system performance

(Detailed discussions will be presented in follow-on sections)

CONTD

I/O operations

- *Basic I/O considerations:*
  - ➢ ***Timing***
    - » Typically the processor and the I/O device will not be operating at the same clock frequency
    - » As a result, we must have a means of synchronizing (at least momentarily) the two in order to effect the information transfer

I/O operations

- **Speed**
  - » During I/O operations, objective is to keep both the processor and the I/O device busy
  - » Not easy to do because of the range of operating speeds of the processor and the I/O device
- ***Coding***
  - » Information in the processor is held in a "machine readable" format (generally binary numbers)

contd

I/O operations

» The data representation is most likely not in a form suitable for external use

✓Externally, we like to think in terms of ASCII, 16-bit Unicode, EBCDIC, etc. --- Must make provisions for code conversion during I/O operations

➢From these three considerations, the I/O interface consists of two parts:

– The hardware interface -- the electrical connections and signal paths

– The software interface -- provides a means for manipulating the data

contd

I/O operations

- ***The I/O interface can be viewed as a "system" of processor registers***

  – Control -- defines the operational characteristics of the interface

  – Status -- tracks the use of the interface -- Is it busy now?

  – Data -- provides the actual data transfer mechanism

I/O operations

- Three categories of I/O operations, based on the control mechanism that is used:

– **Program controlled I/O**

– **Interrupt controlled** I/O -- I/O operations are a result of the processor's response to external

I/O interrupts that indicate the readiness of the

I/O device to transfer data

(More on this later!)

contd

I/O operations

– DMA controlled I/O -- I/O operations are initiated and controlled by hardware external to the processor -- operation and actual data transfer do not involve the processor (Not implemented in the 68HC11)

Programmed I/O

- ***In the program controlled I/O mode:***
  - – The I/O operations are completely supervised by and controlled by the processor
  - – The processor executes program segments that initiate, direct, and terminate the I/O operation
    - » Initialize I/O hardware
    - » Test and wait for I/O device to be “ready”
    - » Perform 1 transfer
    - » If not done, repeat the process

contd

Programmed I/O

- – The program segments can be part of the applications program or a lower-level operating system function

➢This type of operation is available on every computer system

- – Simple to implement
- – Requires very little special hardware or software
- – Primary disadvantage is the loss of processor efficiency -- it is slowed to the speed of the I/O device

Programmed I/O

- ***Two ways address I/O devices:***
  - **Isolated (standard) I/O**
    - » I/O devices have their own unique address space
    - » Individual devices are selected based on the

combined actions of:

- Valid device address being placed on the address bus
- IO/M signal indicates I/O operation
- Valid read or write pulse

contd

Programmed I/O

- ***Memory mapped I/O***
  - » If the I/O device address is part of the memory system addressing scheme, then any instruction that references memory can also be used toperform an I/O operation
  - » I/O device is treated like a memory location
  - » More flexibility in accessing the device, but tradeoff is a loss of real memory locations

Programmed I/O

- ➢ I/O device synchronization: under programmed I/O, data can be transferred using one of two methods:
  - – Normal “conditional” transfers
    - » Transfer can only take place after the processor determines that the I/O device is "ready“
    - » Processor "polls" the device and waits until

      it is ready

contd

Programmed I/O

» This handshaking guarantees that device will not be flooded by the processor (or that the processor won't read the same data more than once)

Programmed I/O

➢–Unconditional transfers

» An instruction transfers data to/from the device without determining if that device is actually ready to send or receive the data

» A "blind" transfer

» As a result of the speed differential between the processor and the device, unconditional transfers are generally used to exchange data with a port that is known to be "ready“

Transfer command (setup) information to a device

Receive status information from a device

Programmed I/O

- ***Parallel I/O***
  - – Each line carries 1 bit of data ord
  - – All 5 ports on the HC11 can be used for parallel I/O
    - » If not used for another I/O subsystem
  - – Ports B and C can be used for strobed I/O or full handshake I/O

contd

Programmed I/O

- *–Uses:*
  - » LED
  - » Keypad
  - » Printer interface
  - » Control relays, switches
  - » Sensor switch inputs

Programmed I/O

- ***Serial I/O***
  - ✓ Uses a single line to transmit bits one after the other
  - ✓ May be synchronous or asynchronous
  - ✓ Port D used for serial I/O
  - ✓ Often used for:
    - » Computer communication
    - » Modem
    - » Mouse
    - » Printer
    - » Network

Programmed I/O

- Programmable Timer
  - Port A used for timer functions
  - Uses:
    - » Generate time delays
    - » Generate pulse streams
    - » Measure period/frequency of input signals
    - » Measure pulse widths

contd

Programmed I/O

- **Analog/Digital Converter**
  - Converts an analog voltage into a binary number
  - Port E used for A/D conversions
  - Many physical quantities are represented by analog values
    - Temperature
    - Voltage
    - Light intensity
    - Pressure

Programmed I/O

- Summary of port functions:
  - ***Port A***:
    - » Timer operations or parallel I/O

      PA0-PA2 input only

      PA4-PA6 output only

      PA3 and PA7 input or output
  - ***Port B:***
    - » Upper 8 bits of address bus (expanded multiplexed mode) or parallel I/O

      PB0-PB7 output only

contd

Programmed I/O

– ***Port C:***

» Multiplexed address/data bus (expanded multiplexed mode) or parallel I/O

✓PC0-PC7 input or output

– ***Port D:***

» Asynchronous serial I/O (PD0-PD1), synchronous serial I/O (PD2-PD5), or parallel I/O

✓ PD0-PD5 input or output

– ***Port E:***

» A/D converter or parallel I/O PE0-PE7 input only

Presentation On

Real World interfacing with Microcontroller

By

Prof.Mahesh Yanagimath **M.Tech (VLSI &ES), MIEEE, MISTE.**

Faculty,Dept of Elertrical and Electronics Engg

Staff Advisor, IEEE Student Branch

Hirasugar Institute of Technology, Nidasoshi.

Content

- Microcontroller
- How to Interface devices
- Microcontroller Interfaces
- 8051 Serial communication
- Serial data transmission modes
- Microcontroller applications

Why do we need to learn Microprocessors/controllers?

- The microprocessor is the core of computer systems.
- Nowadays many communication, digital entertainment, portable devices, are controlled by them.
- A designer should know what types of components he needs, ways to reduce production costs and product reliable.

Microcontrollers

The prime use of a microcontroller :

► To control the operation of a machine using a fixed program that is stored in ROM and that does not change over the lifetime of the system

Typical Microcontrollers

- The most common microcontrollers are 8-bit.
- 4-bit are used in high volume very low cost applications
- 16 & 32 bit are used in high-end applications.
- Typical clock frequencies are 12 - 24 MHz

Different manufacturers of microcontroller

- Intel
- Atmel
- Philips
- Dallas Semiconductors
- Microchip
- Motorola

History of 8051

- ►1981, Intel MCS-51
- ►The 8051 became popular after Intel allowed other manufacturers to make and market an flavor of the 8051.
  - – different speed, amount of on-chip ROM
  - – code-compatible with the original 8051
  - – form a 8051 family

Criteria for Selecting microcontroller

- Meeting the computing needs of the task efficiently and cost effectively
  - speed, the amount of ROM and RAM, the number of I/O ports and timers, size, packaging, power consumption
  - easy to upgrade
  - cost per unit
- availability of software development tools
  - assemblers, debuggers, C compilers, emulator, simulator, technical support
- wide availability and reliable sources of the microcontrollers.

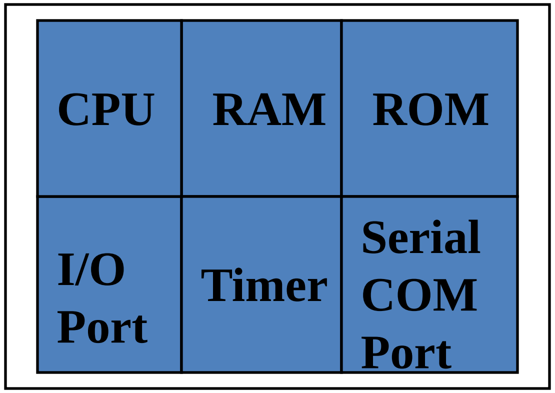
Different aspects of a microcontroller

► Hardware: Interface to the real world

► Software: order how to deal with inputs

Test case: 8051

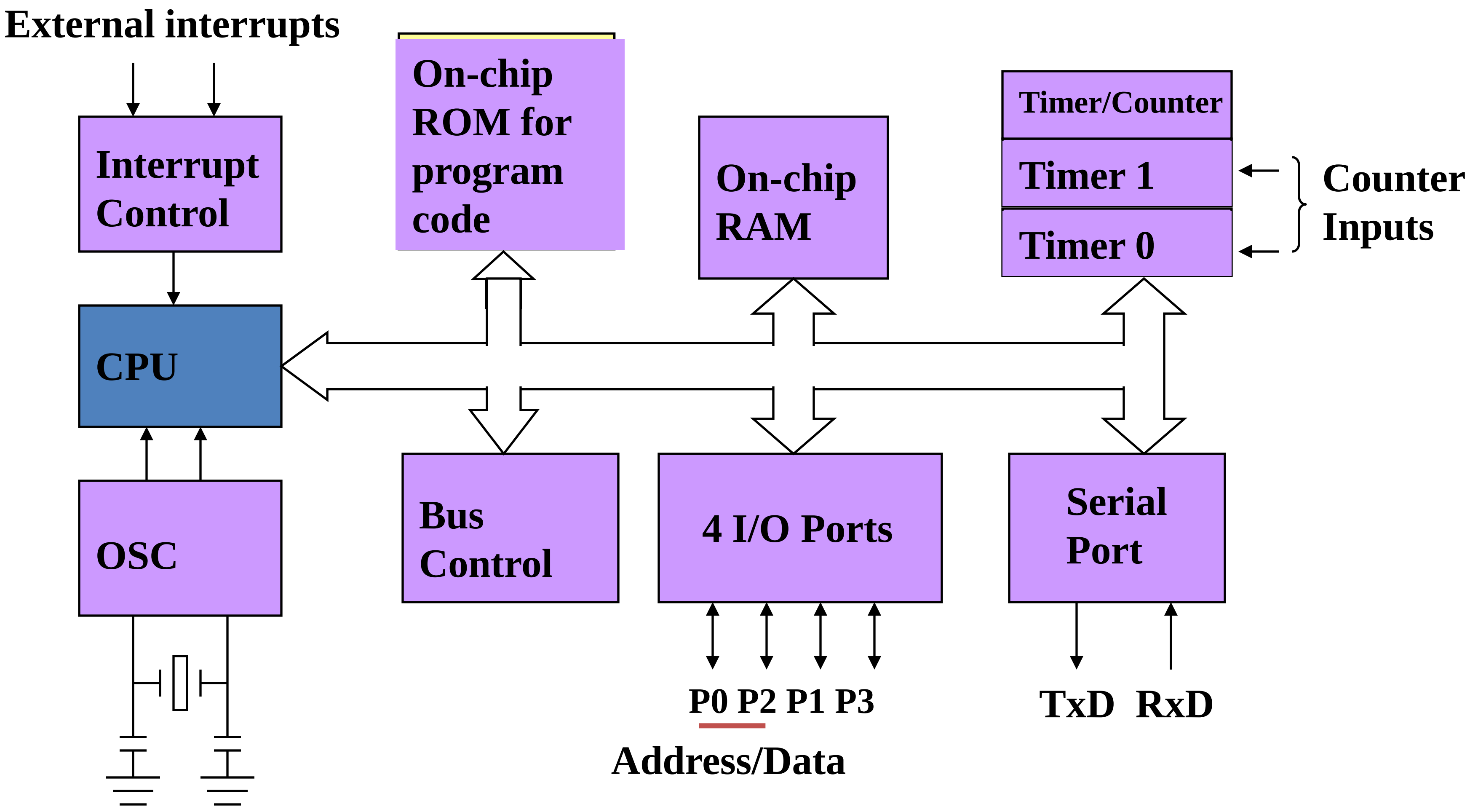
- A smaller computer
- On-chip RAM, ROM, I/O ports...



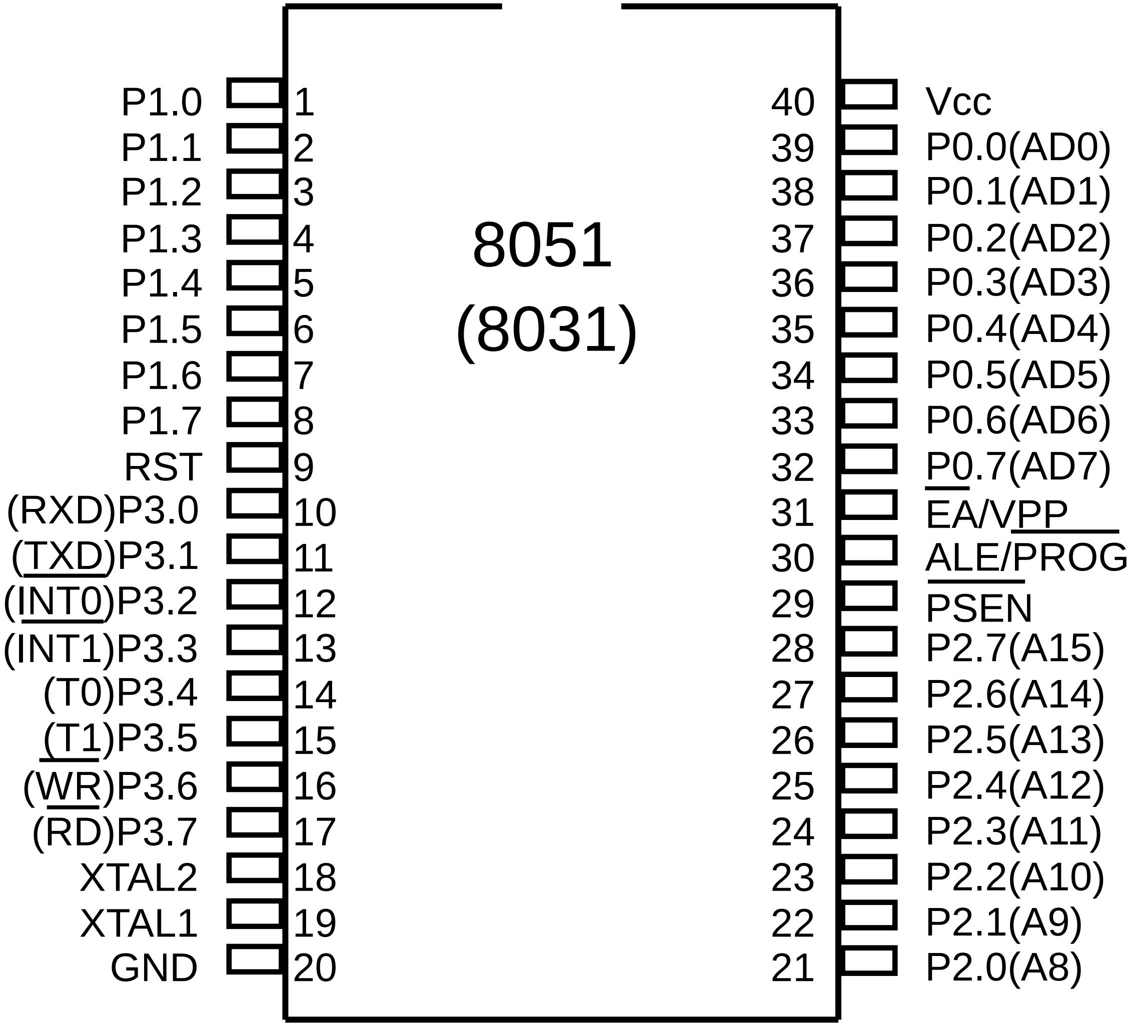
CPU
RAM
ROM
I/O Port
Timer
Serial COM Port

A single chip

Block Diagram



Pin Description of the 8051



8051

(8031)

P1.0 1

P1.1 2

P1.2 3

P1.3 4

P1.4 5

P1.5 6

P1.6 7

P1.7 8

RST 9

(RXD)P3.0 10

(TXD)P3.1 11

(INT0)P3.2 12

(INT1)P3.3 13

(T0)P3.4 14

(T1)P3.5 15

(WR)P3.6 16

(RD)P3.7 17

XTAL2 18

XTAL1 19

GND 20

40 Vcc

39 P0.0(AD0)

38 P0.1(AD1)

37 P0.2(AD2)

36 P0.3(AD3)

35 P0.4(AD4)

34 P0.5(AD5)

33 P0.6(AD6)

32 P0.7(AD7)

31 EA/VPP

30 ALE/PROG

29 PSEN

28 P2.7(A15)

27 P2.6(A14)

26 P2.5(A13)

25 P2.4(A12)

24 P2.3(A11)

23 P2.2(A10)

22 P2.1(A9)

21 P2.0(A8)

What is interfacing?

How to interface Devices

- ►Inputs and Outputs
- ►Compatibility of I/Os
- ►Selecting right microcontroller

Basic I/O considerations

1) Timing

--» Typically the processor and the I/O device will not be operating at the same clock frequency.

2) Speed

– » During I/O operations, objective is to keep both the processor and the I/O device busy

– » Not easy to do because of the range of operating speeds of the processor and the I/O device

3) Coding

– » Information in the processor is held in a "machine readable" format (generally binary numbers)

The I/O interface can be viewed as a "system" of processor registers

- – Control -- defines the operational characteristics of the interface

- – Status -- tracks the use of the interface

  -- Is it busy now?

- – Data -- provides the actual data transfer mechanism

➢Three categories of I/O operations, based on the control mechanism that is used:

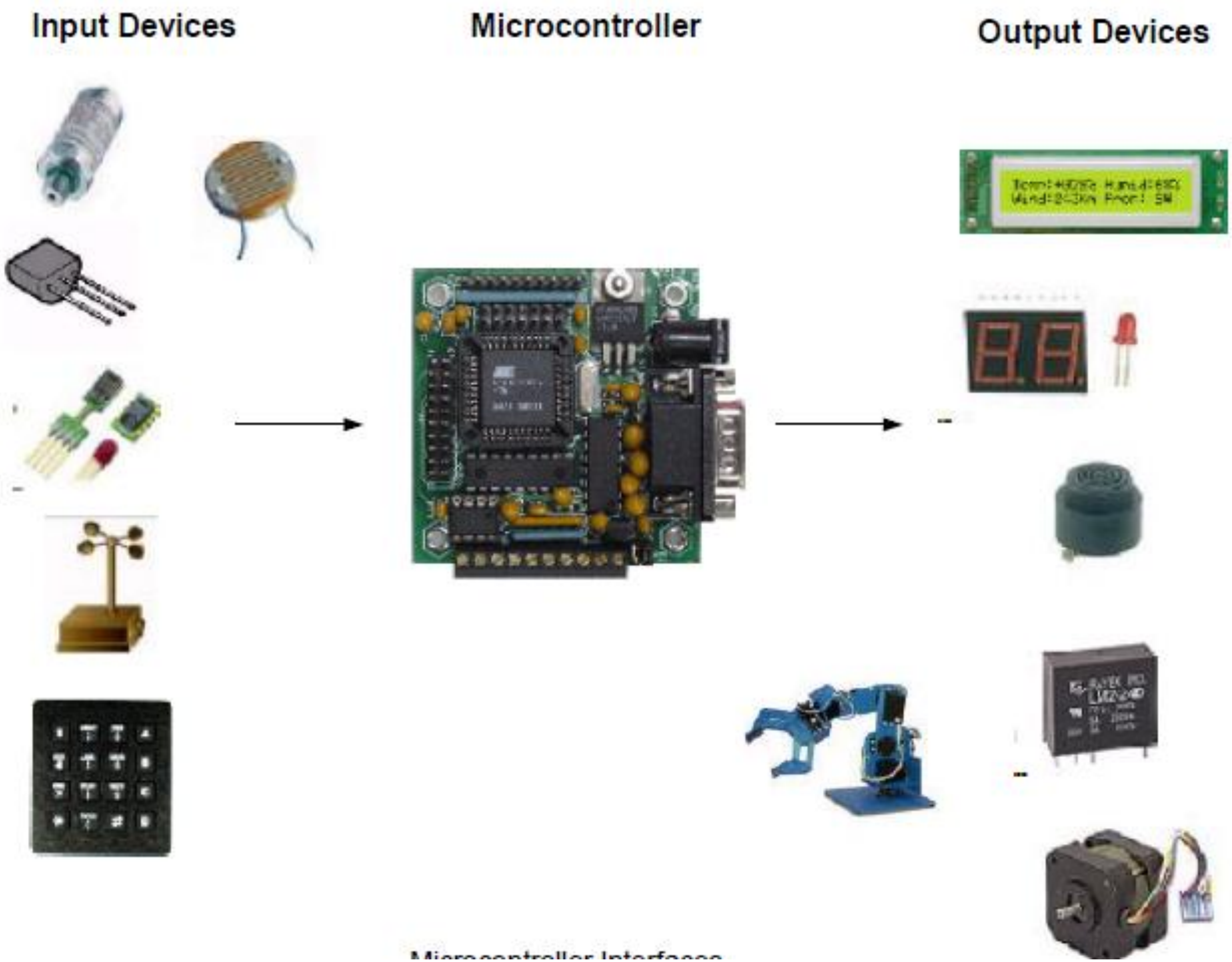
– **Program controlled I/O**

– **Interrupt controlled** I/O -- I/O operations are a result of the processor's response to external

I/O interrupts that indicate the readyness of the

I/O device to transfer data

-- DMA controlled I/O -- I/O operations are initiated and controlled by hardware external to the processor -- operation and actual data transfer do not involve the processor (Not implemented in the 68HC11)

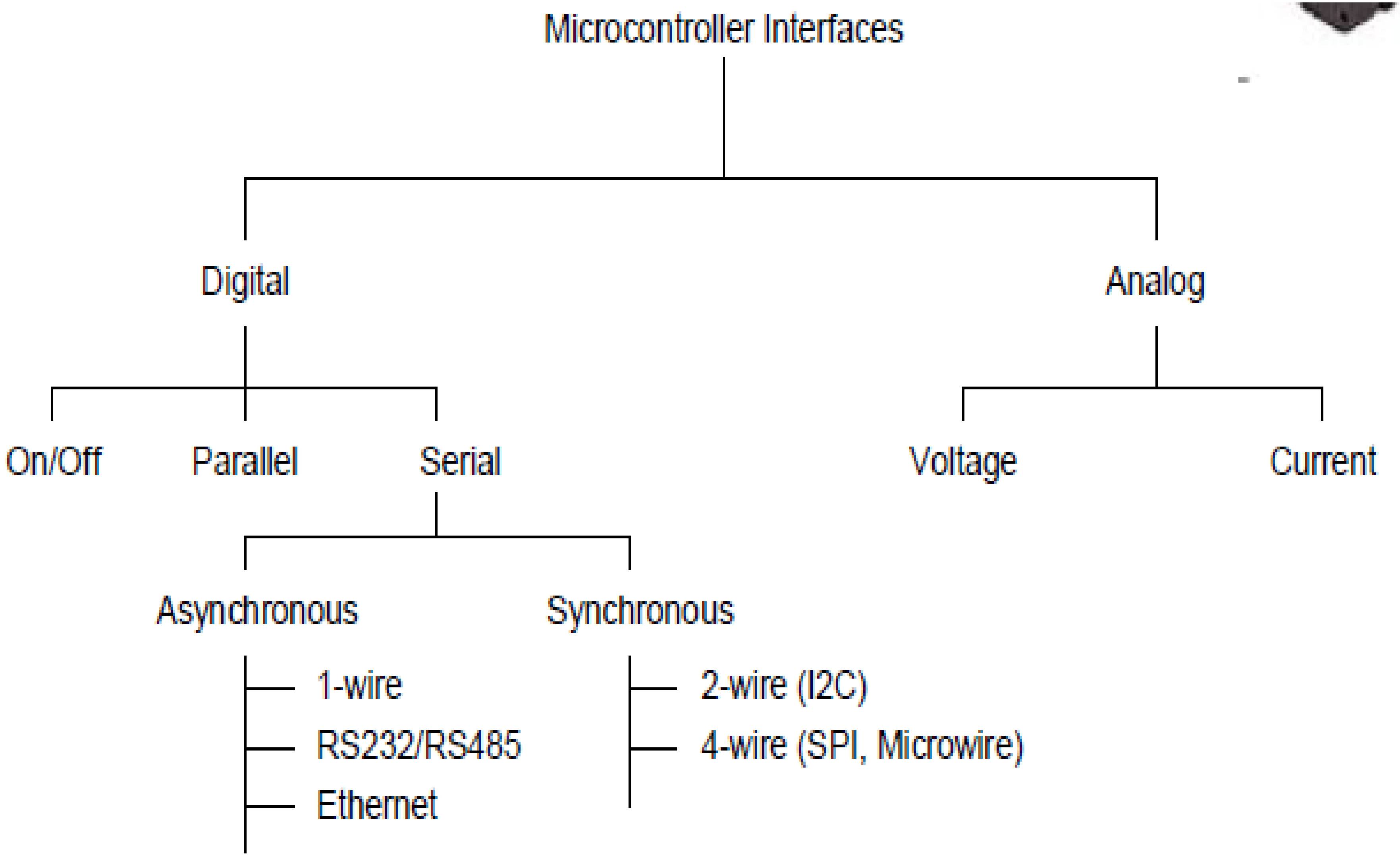


Input Devices

Microcontroller

Output Devices

Microcontroller Interfaces



Microcontroller Interfaces

Digital

Analog

On/Off

Parallel

Serial

Voltage

Current

Asynchronous

Synchronous

1-wire

RS232/RS485

Ethernet

2-wire (I2C)

4-wire (SPI, Microwire)

Digital Inputs/Outputs

On/OFF control and monitoring.

Advantages

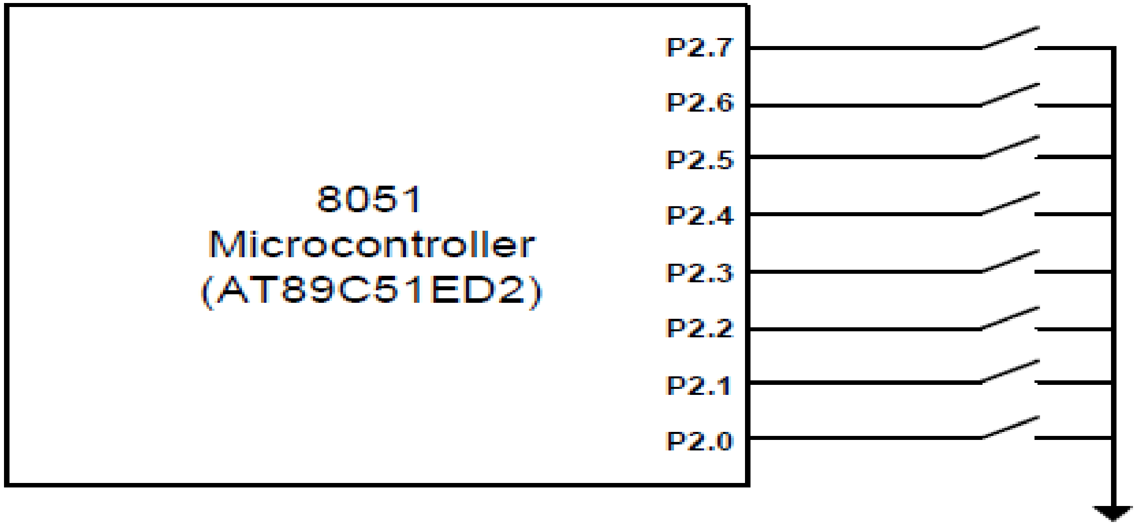
- Simplest interface
- Lowest-cost to implement (built into the microcontroller)
- High speed
- Low programming overhead

Disadvantages

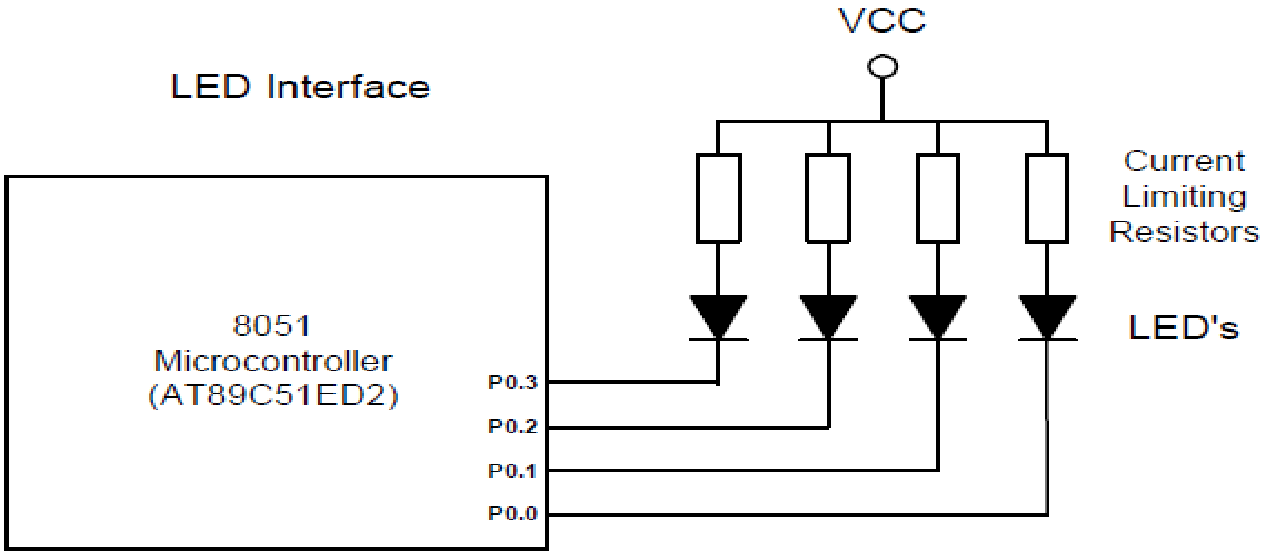
- Only on/off control/monitoring
- Short distance, few feet maximum.
- Single device control/monitoring

Digital Input Example: Reading the status of buttons or switches

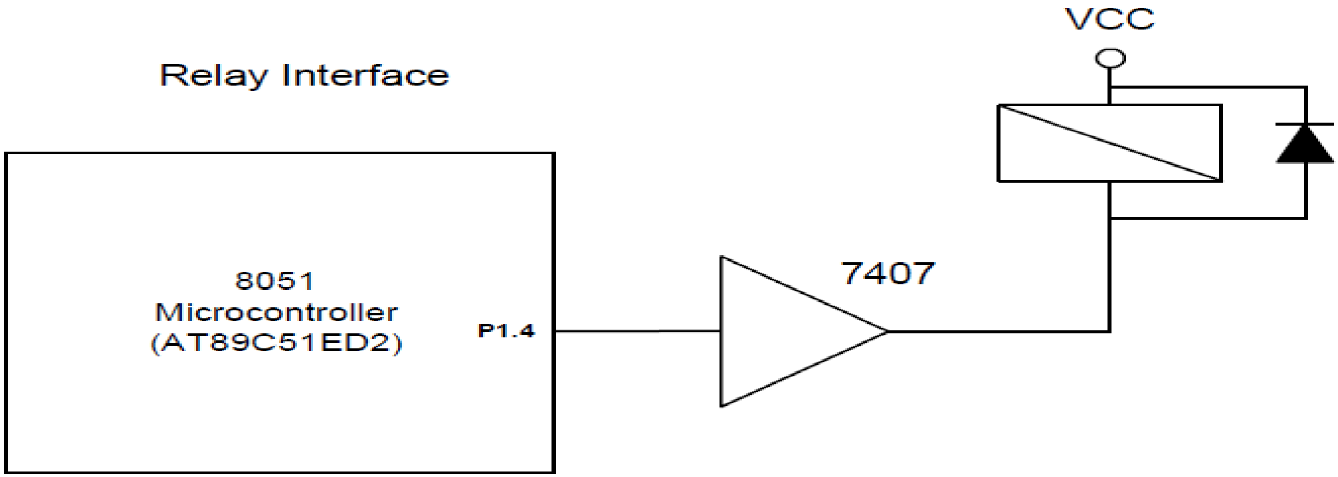
Single-ended
(non-matrix) switches



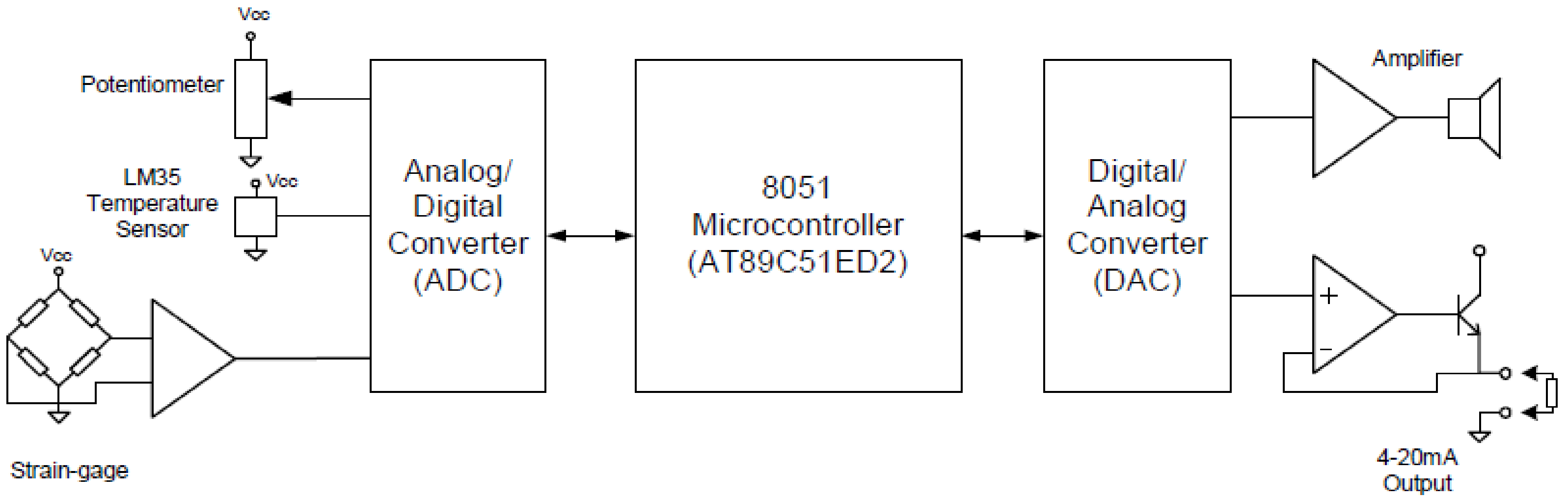
Digital Output Example: LED control



Digital Output Example: Relay control



Analog Interface



Analog Inputs/Outputs

Voltage-based control and monitoring.

Advantages

- Simple interface
- Low cost for low-resolutions
- High speed
- Low programming overhead

Disadvantages

- High cost for higher resolutions
- Not all microcontrollers have analog inputs/outputs built-in
- Complicates the circuit design when external ADC or DAC are needed.
- Short distance, few feet maximum.

Voltage type: Typical ranges

- 0 to 2.5V
- 0 to 4V
- 0 to 5V
- +/- 2.5V
- +/- 4V
- +/- 5V

Current type: Typical ranges

- 0-20mA
- 4-20mA

Analog Inputs

Physical effect produces an analog voltage or current

Microphone

In phones, cameras, voice recorders, …

Accelerometer

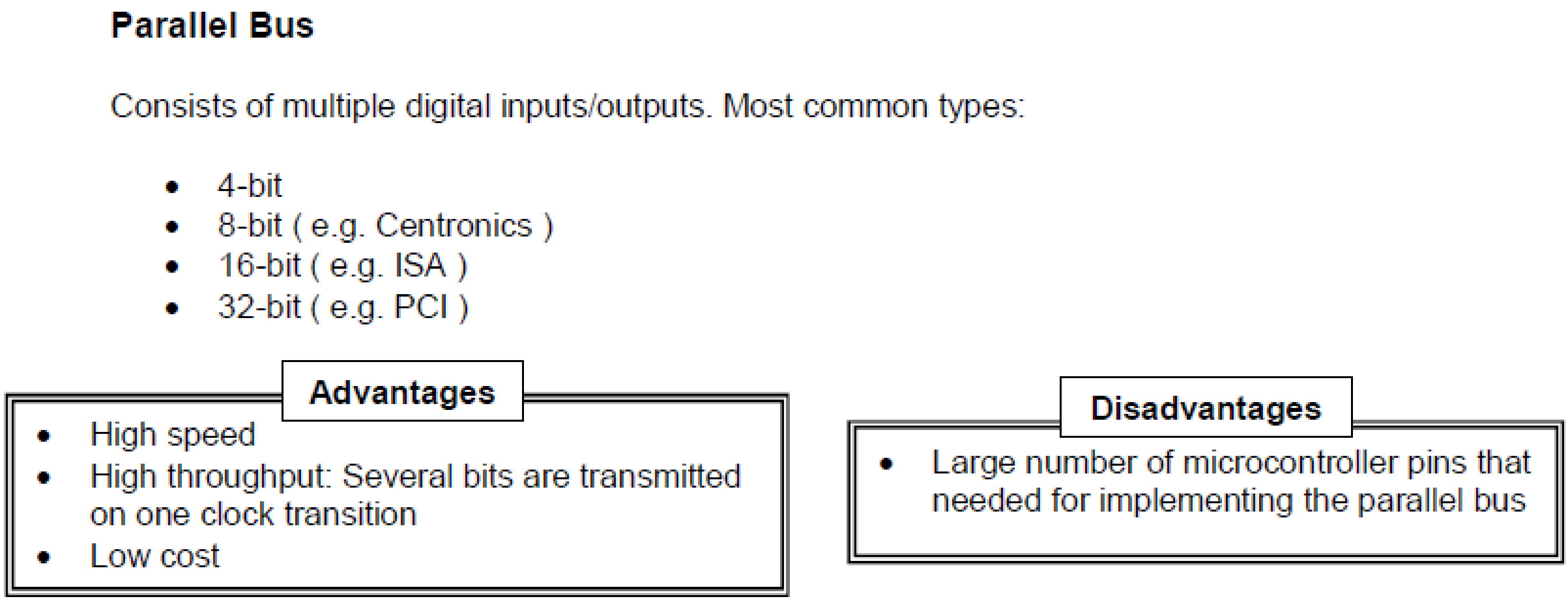
In airbag controllers

Fluid-flow sensors

In industrial machines, coffee machines, …

Gas detectors

In safety equipment



Parallel Bus

Consists of multiple digital inputs/outputs. Most common types:

- 4-bit
- 8-bit (e.g. Centronics)
- 16-bit (e.g. ISA)
- 32-bit (e.g. PCI)

Advantages

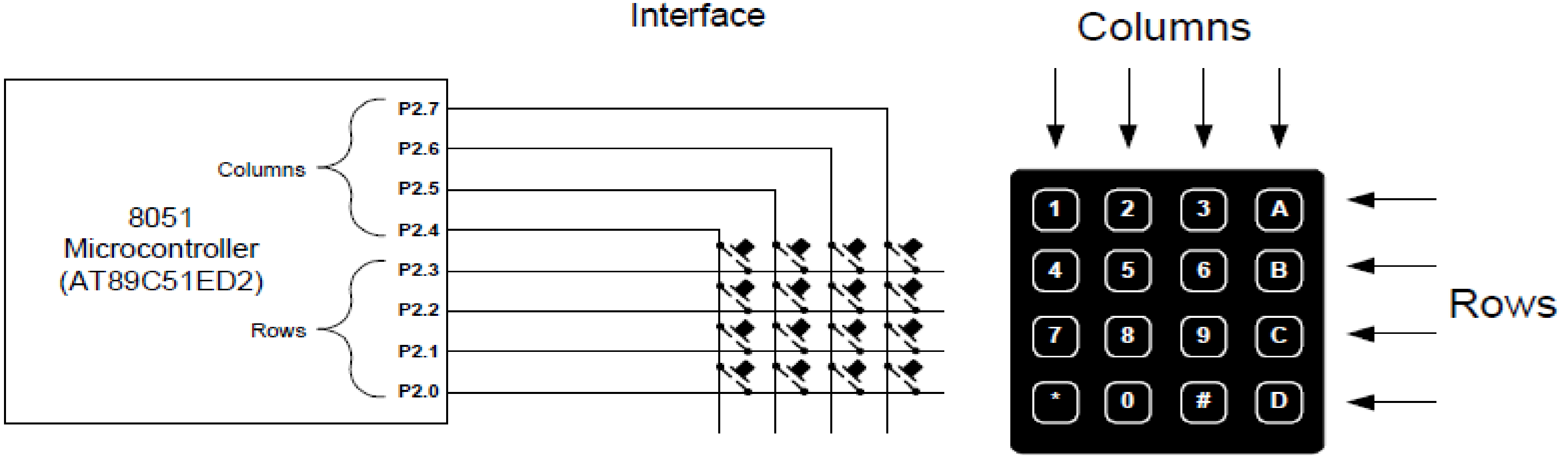
- High speed
- High throughput: Several bits are transmitted on one clock transition
- Low cost

Disadvantages

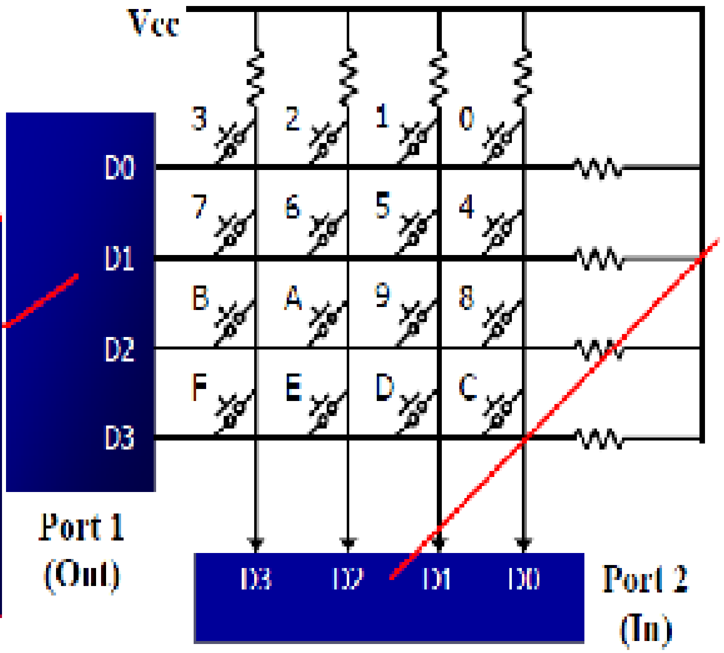
- Large number of microcontroller pins that needed for implementing the parallel bus

Digital Input Example: Keypad Interface

4X4 Matrix Keypad Interface

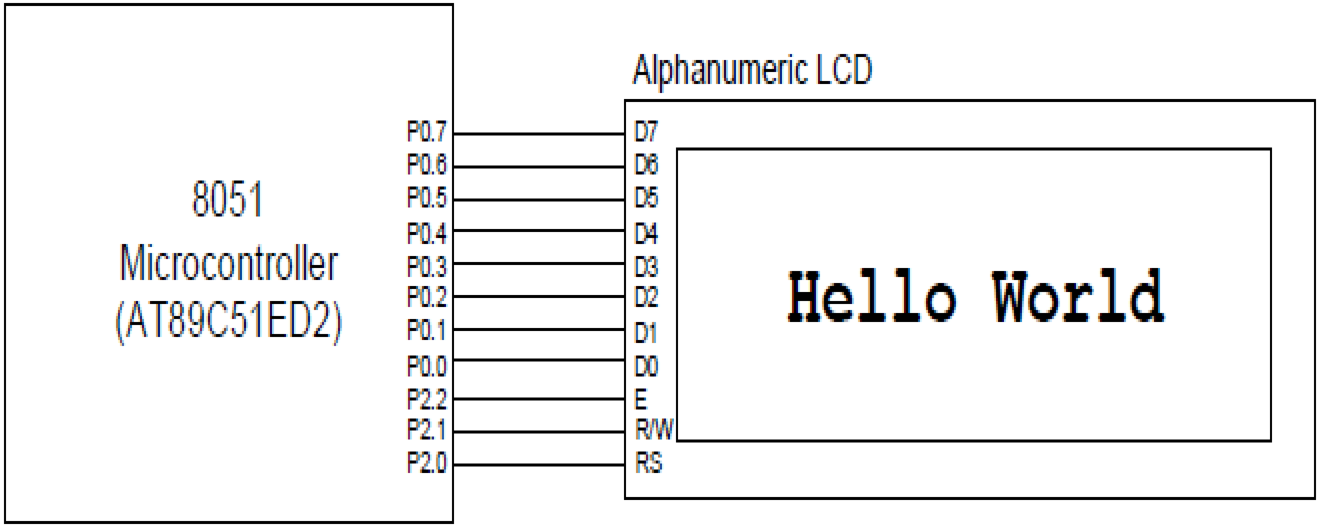


If all the rows are grounded and a key is pressed, one of the columns will have 0 since the key pressed provides the path to ground



If no key has been pressed, reading the input port will yield 1s for all columns since they are all connected to high (V_{cc})

8-bit LCD Interface

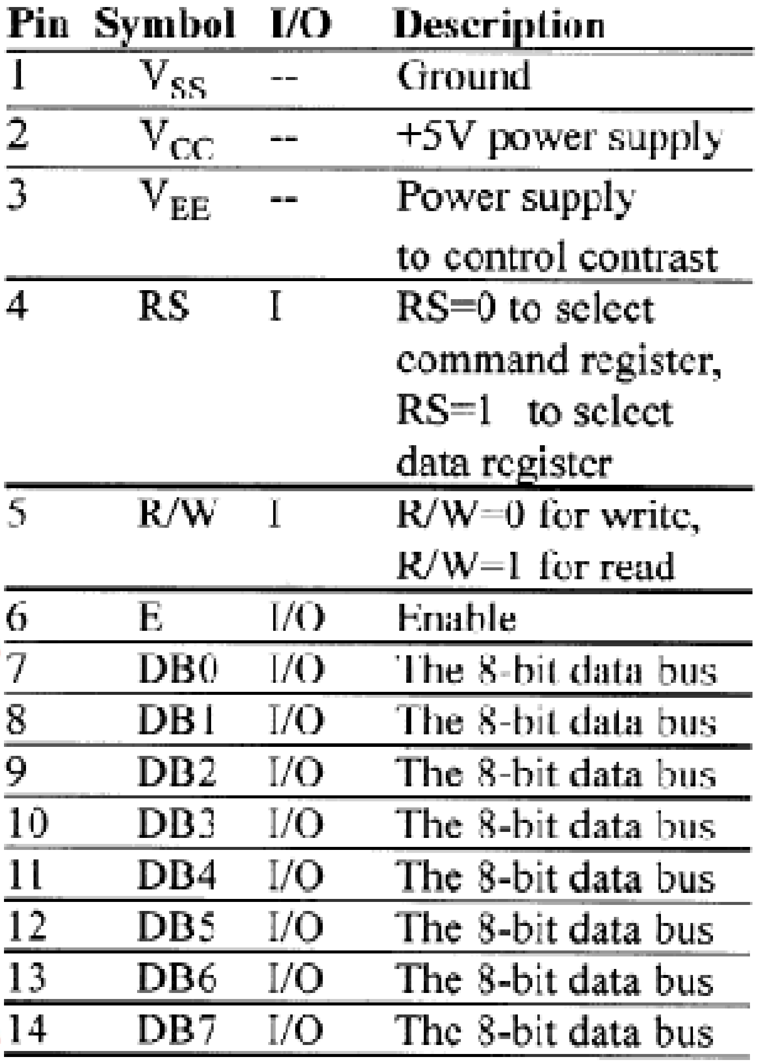


LCD Interfacing

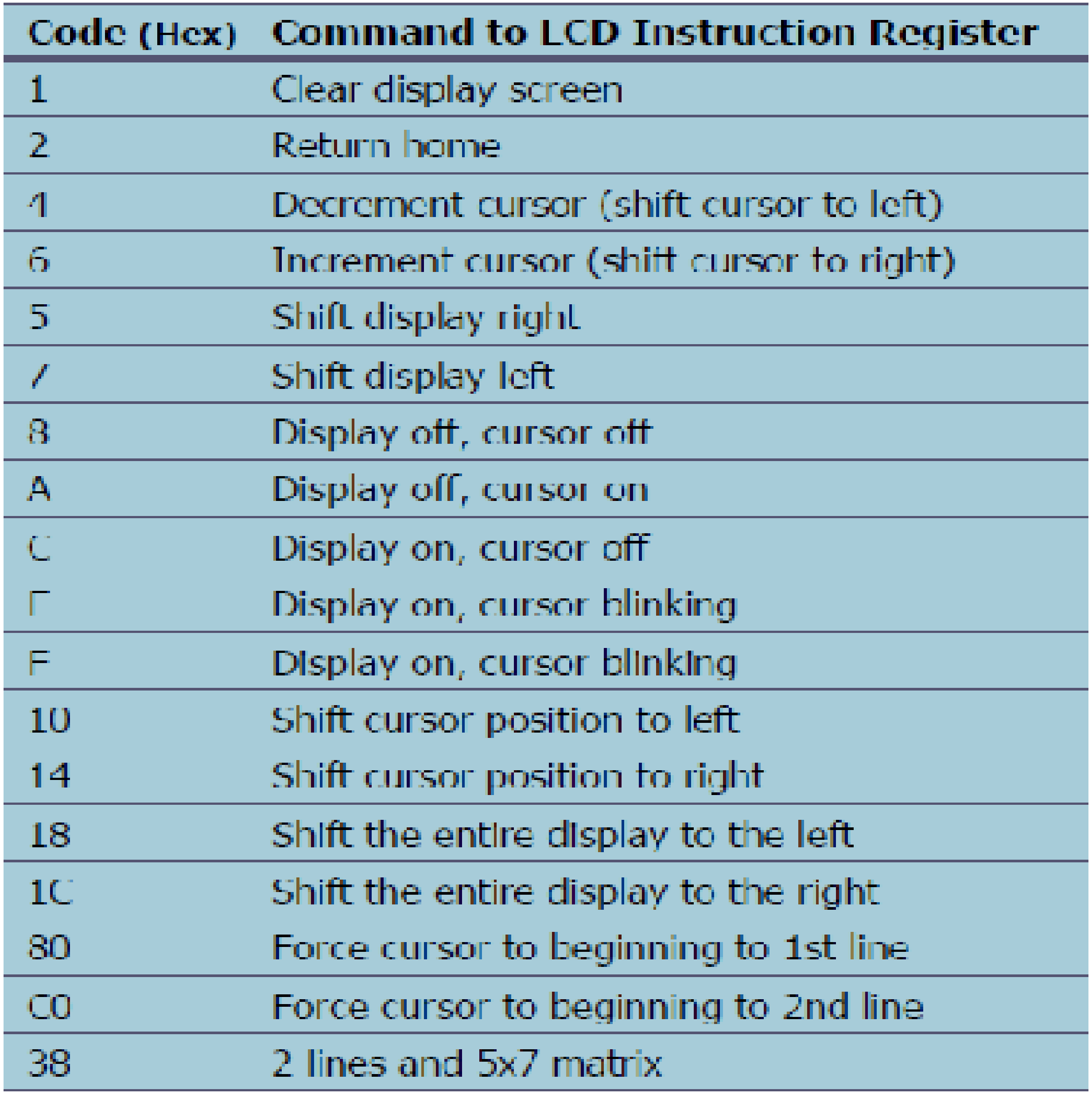
LCD is finding widespread use replacing LEDs for the following reasons:

- The declining prices of LCD
- The ability to display numbers, characters, and graphics
- Incorporation of a refreshing controller into the LCD, thereby relieving the CPU of the task of refreshing the LCD
- Ease of programming for characters and graphics

- Send displayed information or instruction command codes to the LCD
- Read the contents of the LCD's internal registers

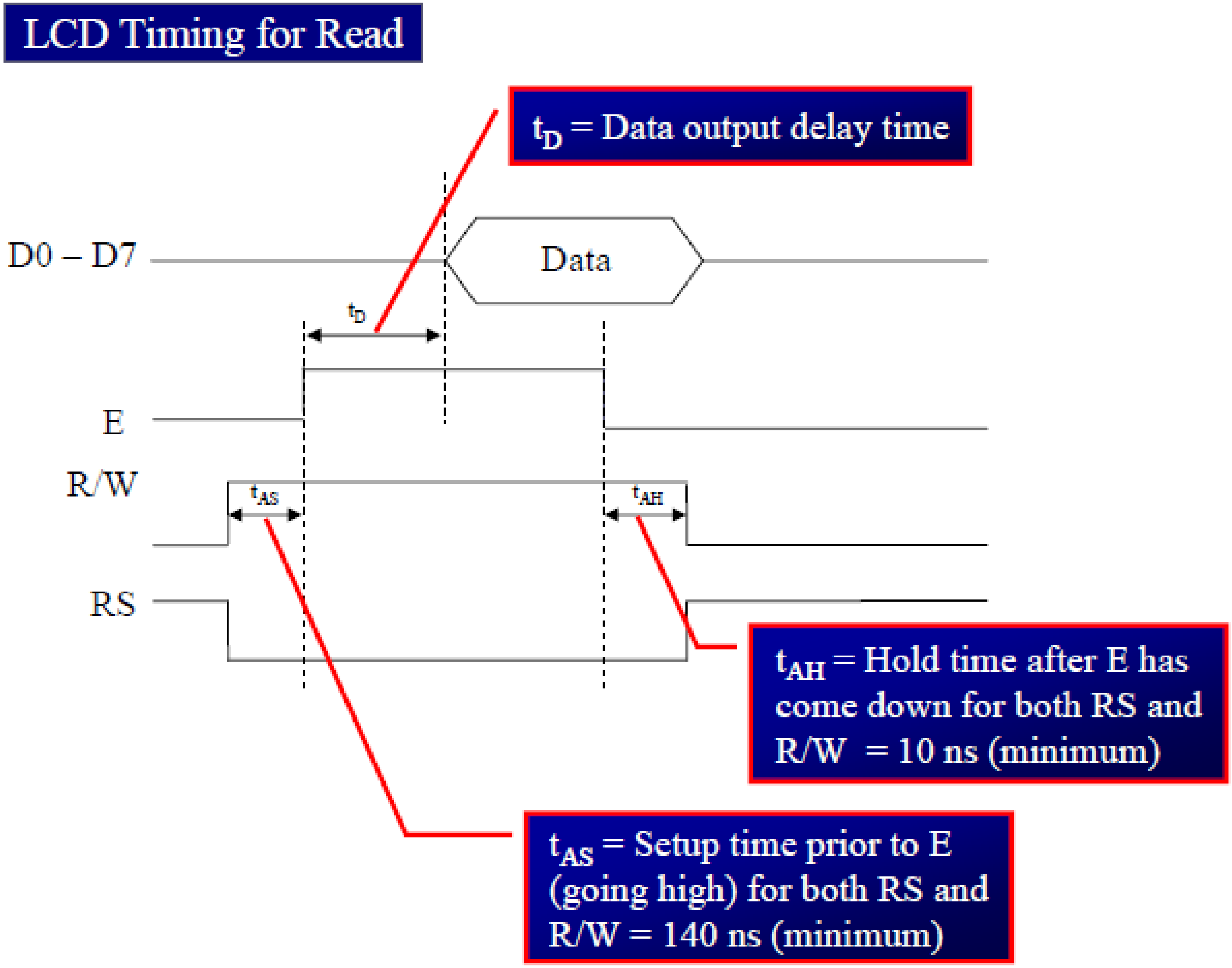


Pin	Symbol	I/O	Description
1	V_{SS}	--	Ground
2	V_{CC}	--	+5V power supply
3	V_{EE}	--	Power supply to control contrast
4	RS	I	RS=0 to select command register, RS=1 to select data register
5	R/W	I	R/W=0 for write, R/W=1 for read
6	E	I/O	Enable
7	DB0	I/O	The 8-bit data bus
8	DB1	I/O	The 8-bit data bus
9	DB2	I/O	The 8-bit data bus
10	DB3	I/O	The 8-bit data bus
11	DB4	I/O	The 8-bit data bus
12	DB5	I/O	The 8-bit data bus
13	DB6	I/O	The 8-bit data bus
14	DB7	I/O	The 8-bit data bus



Code (Hex)	Command to LCD Instruction Register
1	Clear display screen
2	Return home
4	Decrement cursor (shift cursor to left)
6	Increment cursor (shift cursor to right)
5	Shift display right
7	Shift display left
8	Display off, cursor off
A	Display off, cursor on
C	Display on, cursor off
E	Display on, cursor blinking
F	Display on, cursor blinking
10	Shift cursor position to left
14	Shift cursor position to right
18	Shift the entire display to the left
1C	Shift the entire display to the right
80	Force cursor to beginning to 1st line
C0	Force cursor to beginning to 2nd line
38	2 lines and 5x7 matrix

LCD timing diagram for read operation



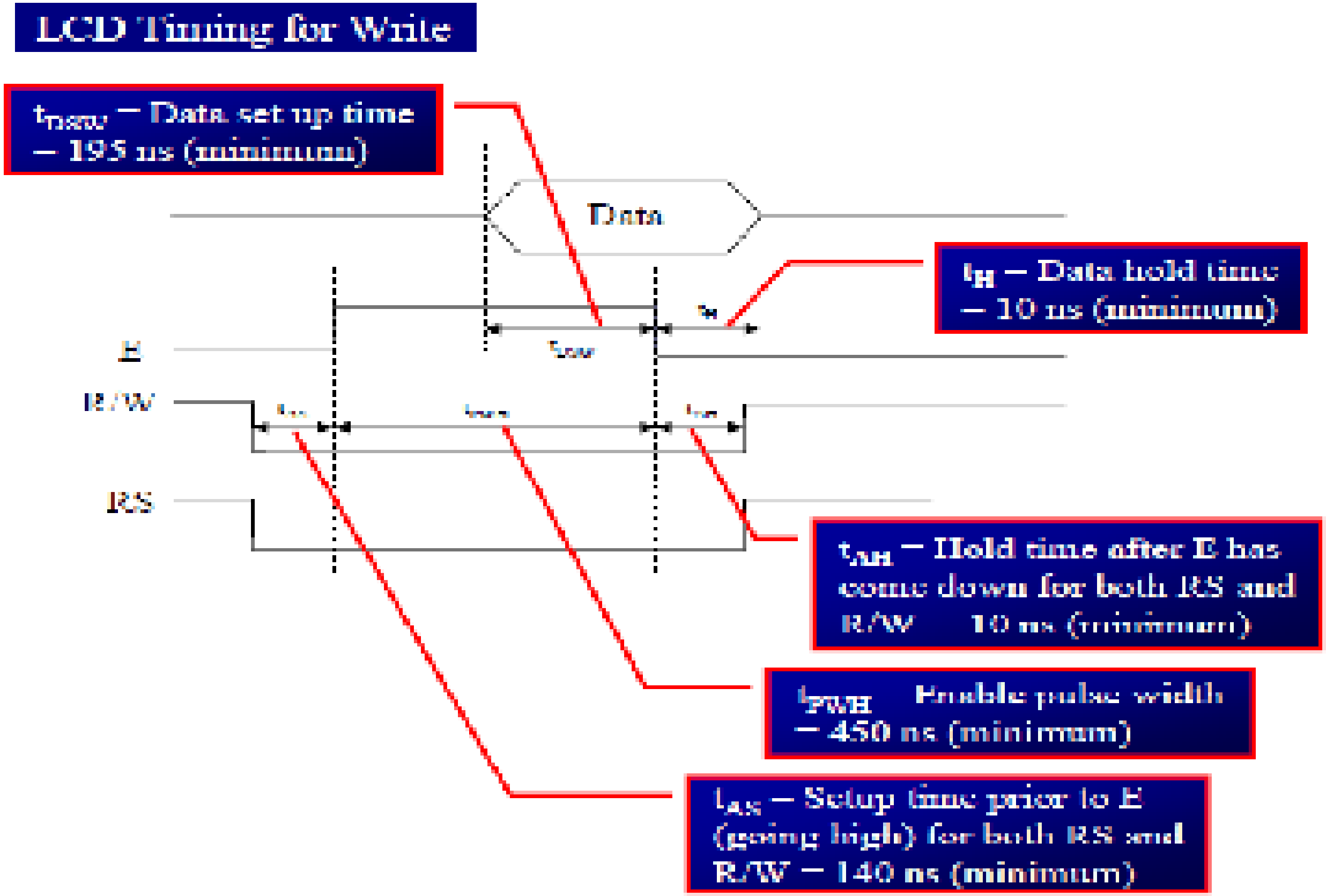
LCD Timing for Read

t_D = Data output delay time

t_{AH} = Hold time after E has come down for both RS and R/W = 10 ns (minimum)

t_{AS} = Setup time prior to E (going high) for both RS and R/W = 140 ns (minimum)

LCD timing diagram for write Operation



LCD Timing for Write

t_{DSW} – Data set up time – 195 ns (minimum)

t_H – Data hold time – 10 ns (minimum)

t_{AH} – Hold time after E has come down for both RS and R/W 10 ns (minimum)

t_{PWH} Enable pulse width – 450 ns (minimum)

t_{AS} – Setup time prior to E (going high) for both RS and R/W – 140 ns (minimum)

Serial Buses

I2C (Inter Integrated Circuit bus)

2-wire interface with one master and multiple slaves (multi-master configurations possible). Originated by Philips Semiconductor in the early 80's to connect a microcontroller to peripheral devices in TV sets.

Signals: DATA (**SDA**), CLOCK (**SCL**) and Ground. **SDA** is always bi-directional; SCL is bi-directional only in multi-master mode.

Maximum allowable capacitance on the lines is 400 pF. Typical device capacitance is 10 pF.

To start the communications, the bus master (typically a microcontroller) places the address of the device with which it intends to communicate (the slave) on the bus. All slave devices monitor the bus to determine if the master device is sending their address. Only the device with the correct address communicates with the master

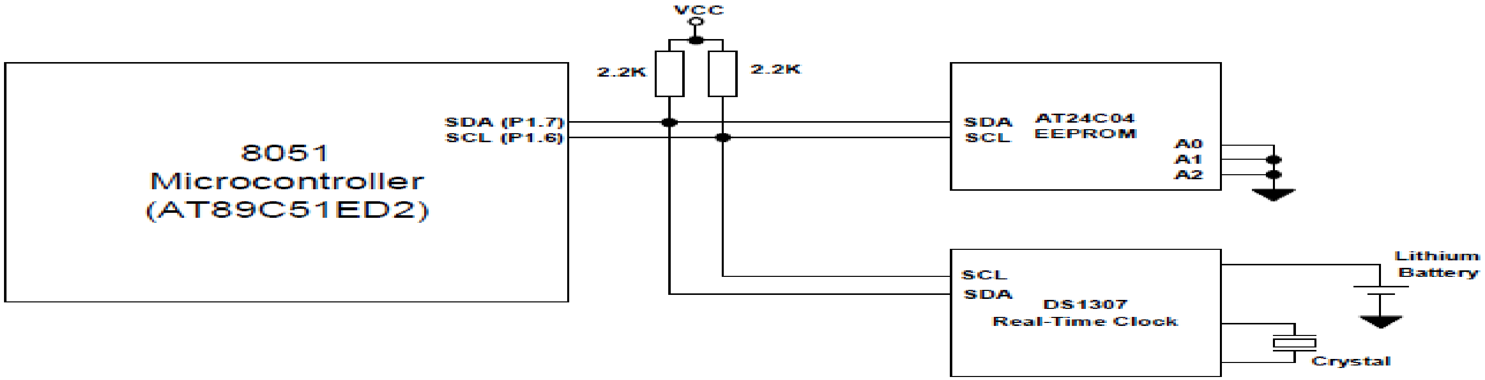
Advantages

- Multiple slave devices can be accessed with only 3 wires
- Low-cost to implement
- Implemented in hardware or software
- Ease to implement, many examples
- Supports multi-master configuration

Disadvantages

- Short distance
- Slow speed: 100 KHz although 400 KHz and 1 MHz slave devise exist. These can not coexist with slower devices.
- Limited device addresses

2-wire (I2C) interface



SPI (Serial Peripheral Interface)

4-wire interface with one master and multiple slaves. Signals: DATA IN, DATA OUT, CLOCK, CS (Chip Select)

Originated by Motorola, SPI bus is a relatively simple synchronous serial interface for connecting low speed external devices using minimal number of wires. A synchronous clock shifts serial data into and out of the microcontrollers in blocks of 8 bits.

SPI bus is a master/slave interface. Whenever two devices communicate, one is referred to as the "master" and the other as the "slave" device. The master drives the serial clock. SPI is full duplex: Data is simultaneously transmitted and received.

Advantages

- Multiple slave devices can be accessed with only few wires
- Low-cost
- Implemented in hardware or software
- Ease to implement, many examples
- Can be high speed (e.g. 4MHz or higher if implemented in hardware)

Disadvantages

- Short distance
- Data and clock lines can be shared but each device requires a separate Chip Select signal, limiting the number of devices in limited I/O systems

RS232

Asynchronous communications

Advantages

- Popular interface with many examples
- Many compatible legacy devices
- Relatively long distance, 50 feet maximum for low baud rates although longer distances work in practice, with low baud rates and error correction
- Immune to noise due to +/-5 Volts or higher voltage levels for logic “0” and “1”
- Implemented in hardware or software
- Ease to implement, many examples

Disadvantages

- More suitable for system to system communications, not so much for chip to chip or chip to sensor
- Low speed for long distance, 115200 baud can be achieved with small microcontrollers using short distances
- Requires transceiver chips which add to system cost (TTL/CMOS level RS232 can be used without transceiver chips).
- Single master/single slave

RS485

Asynchronous communications

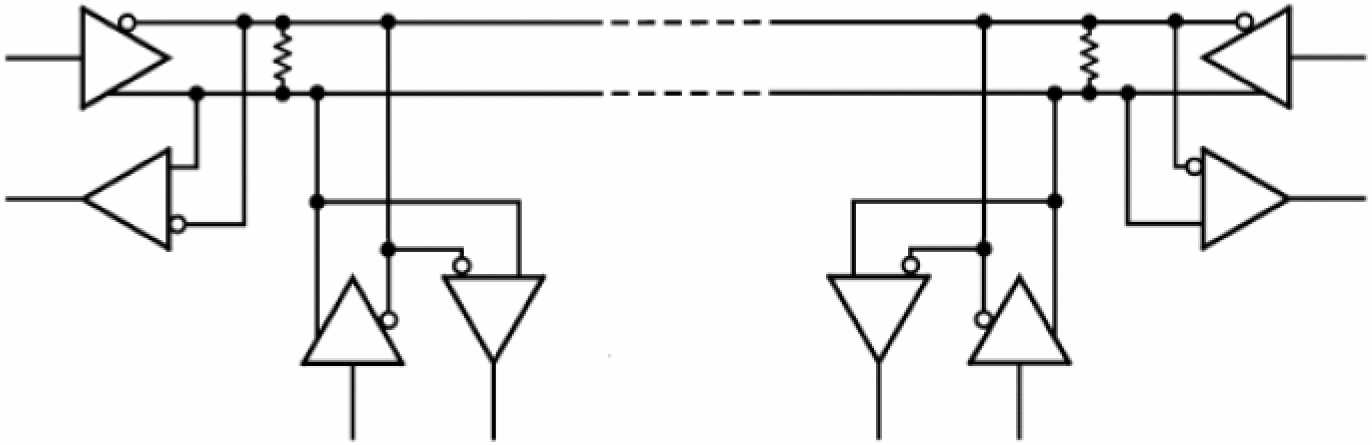
Advantages

- Popular interface with many examples
- Very long distance, thousands of feet
- Immune to noise due to differential voltage
- Implemented in hardware or software
- Ease to implement, many examples
- Widely used in industrial automation
- Higher speeds beyond 115200 baud

Disadvantages

- More suitable for system to system communications, not so much for chip to chip or chip to sensor
- Requires transceiver chips and twisted pair cable with terminating resistors which add to system cost.

RS485 Network Topology: Any station can communicate with any other station, but not at the same time.



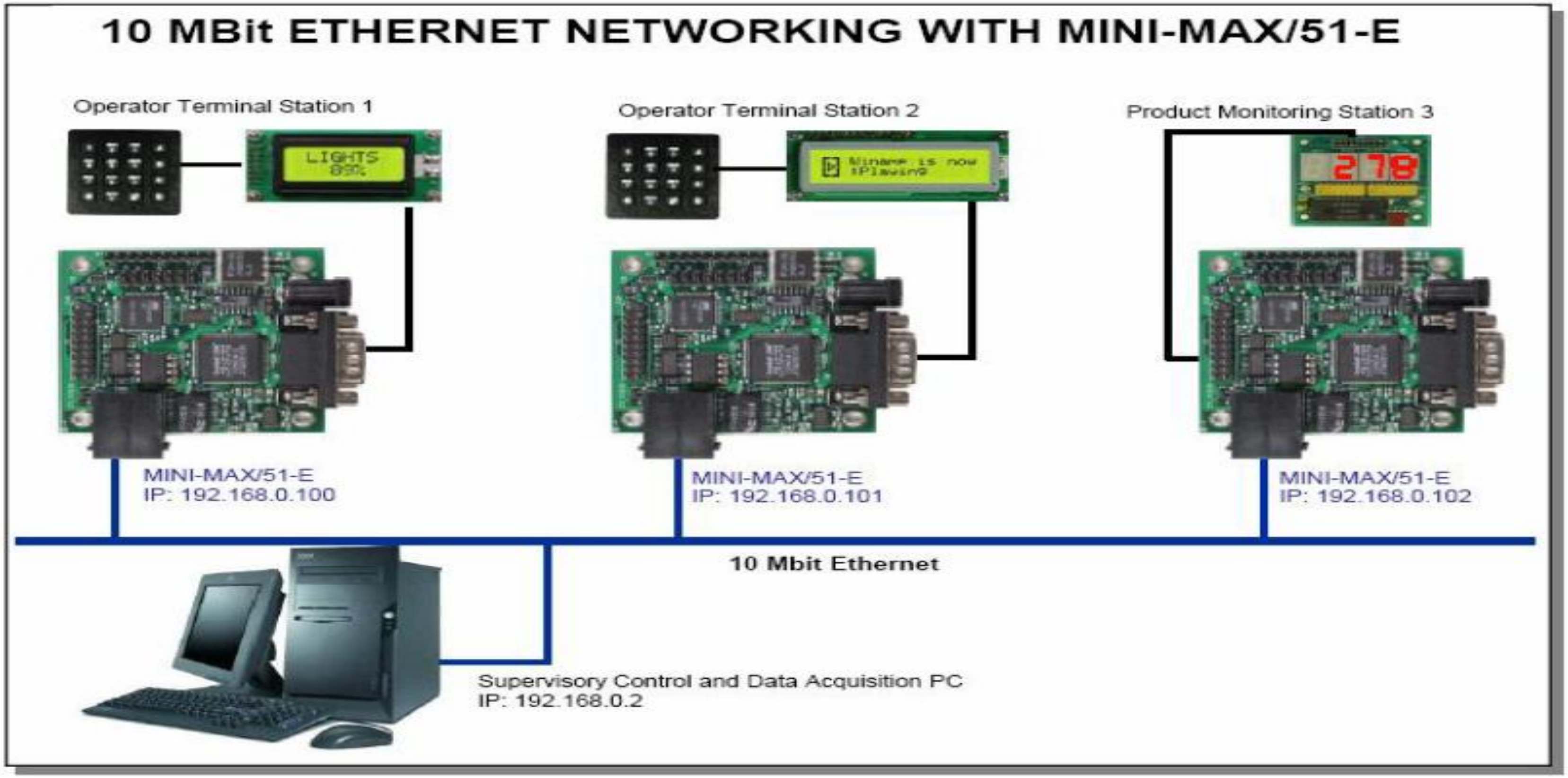
Ethernet

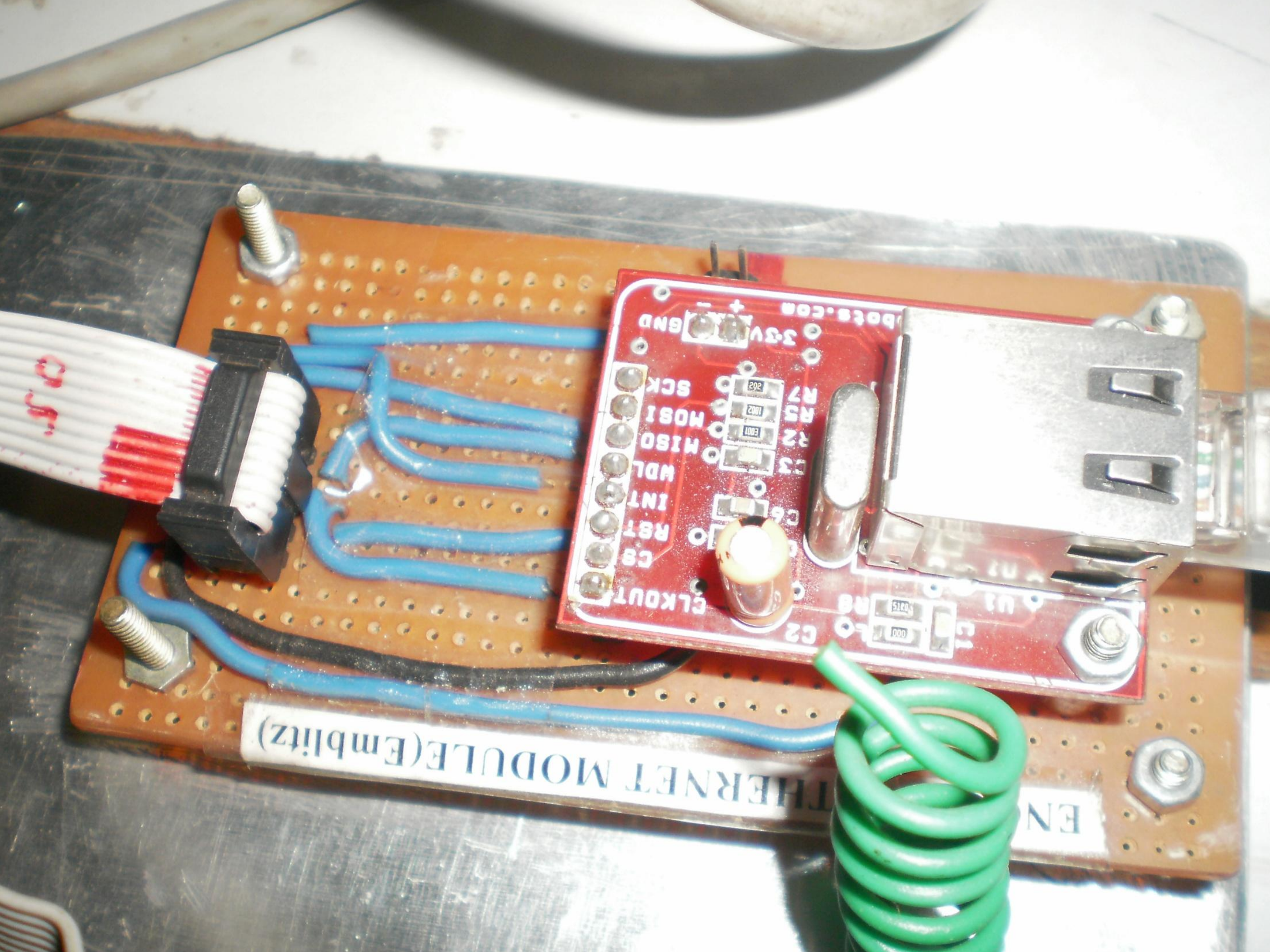
Advantages

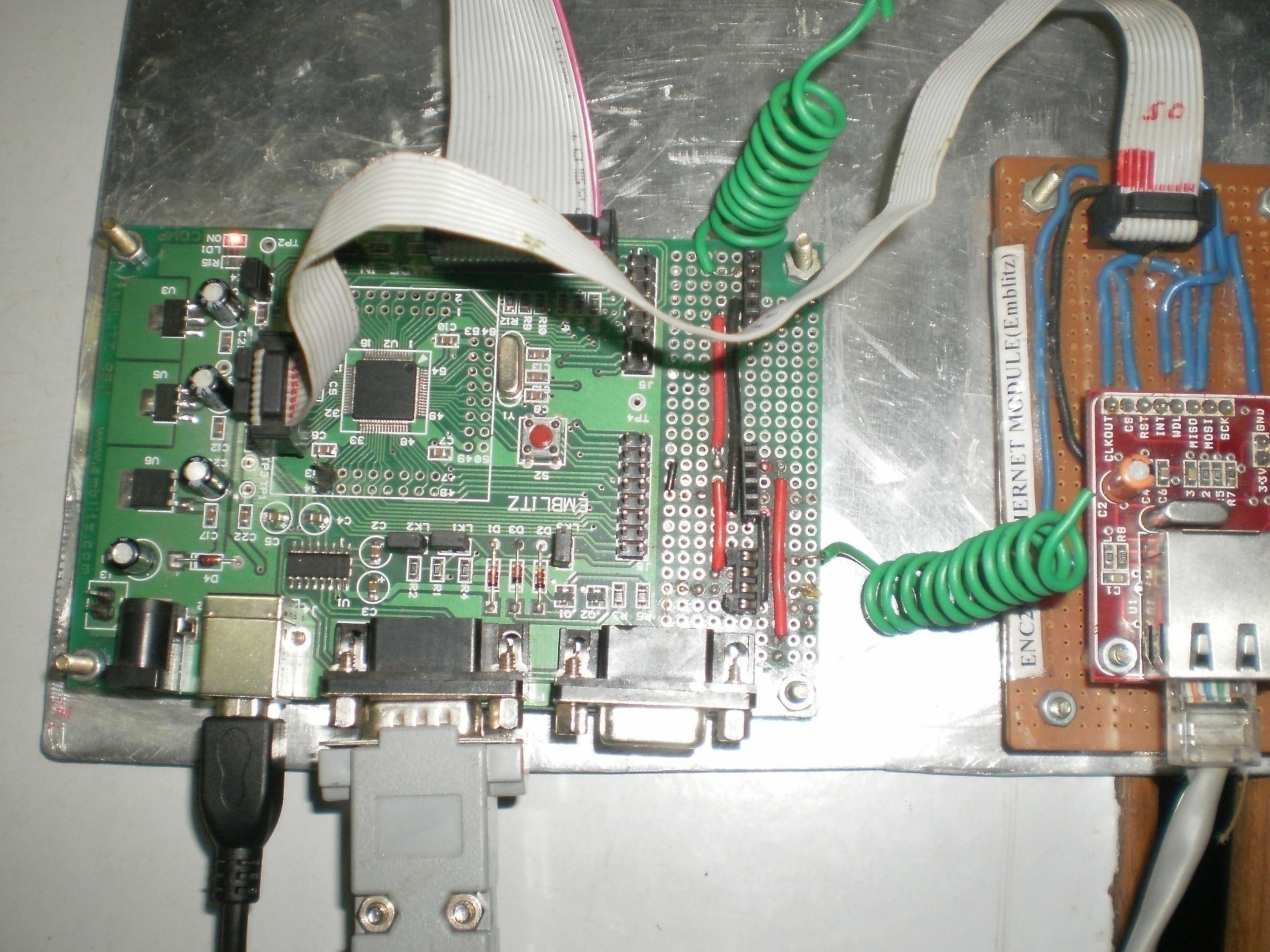
- Very high speed (10Mbit to 100Mbit/s)
- Very long distance, hundreds of feet can be achieved, more with hubs and switches
- Immune to noise
- Widely used in industrial automation due to noise immunity

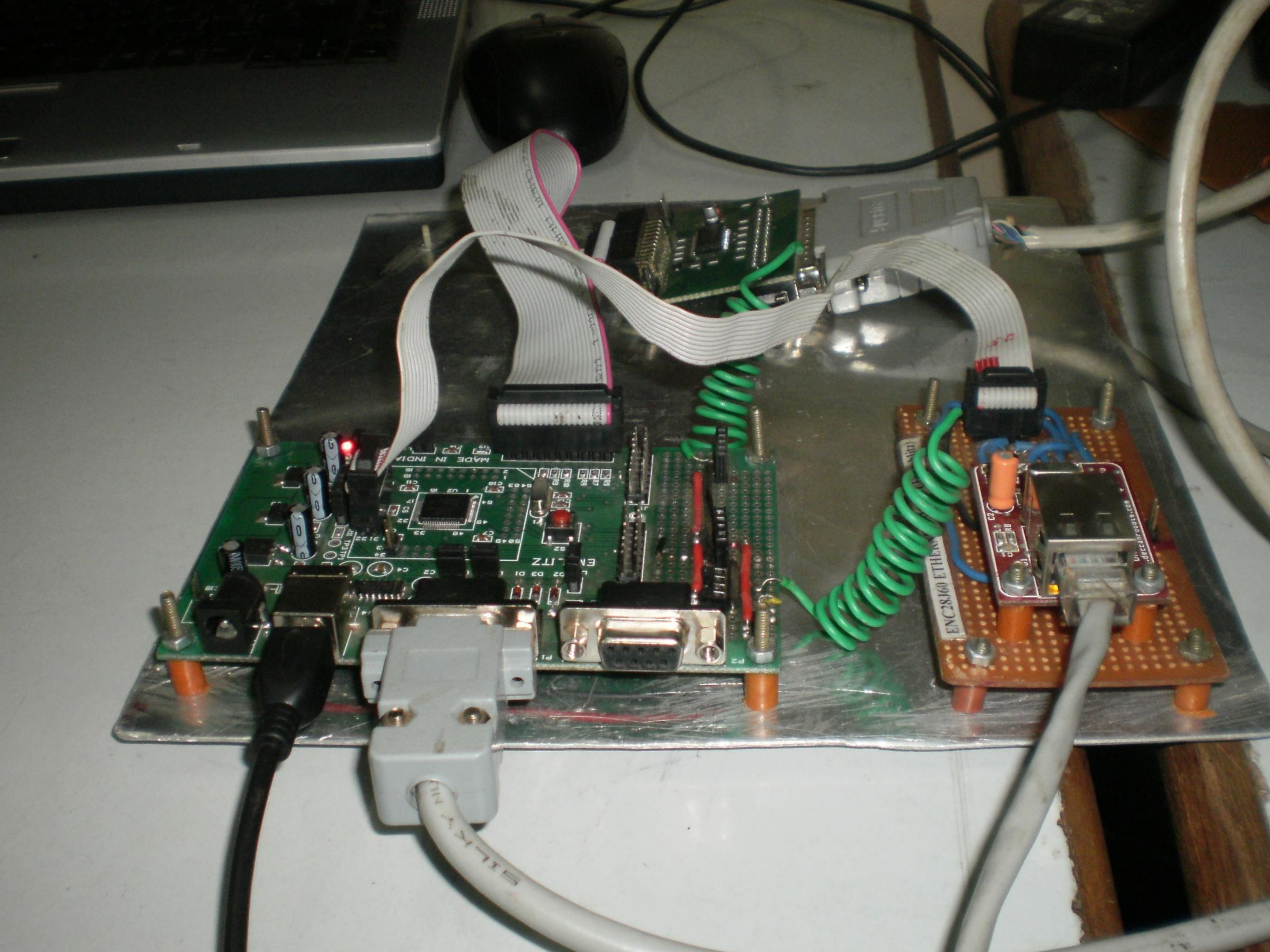
Disadvantages

- Cost
- More suitable for system to system communications, not so much for chip to chip/sensor
- Requires Ethernet chipset, transformer, jack and special cabling that add to system cost.
- Complicated to implement
- High code footprint


10 MBit ETHERNET NETWORKING WITH MINI-MAX/51-E
Operator Terminal Station 1
LIGHTS 89%
Operator Terminal Station 2
Product Monitoring Station 3
278
MINI-MAX/51-E
IP: 192.168.0.100
MINI-MAX/51-E
IP: 192.168.0.101
MINI-MAX/51-E
IP: 192.168.0.102
10 Mbit Ethernet
Supervisory Control and Data Acquisition PC
IP: 192.168.0.2


EN THERNET MODULE(Embitz)
GND
3.3V
SCK
MOSI
MISO
WDL
INT
RST
CS
CLKOUT
R7
R5
R2
C3
C6
C2
R8
C1
U1
J.P


EMBLITZ
ENC2
MODULE(Emblitz)
CLKOUT
CS
RST
INT
WOL
MISO
MOSI
SCK
GND
3.3V


MADE IN INDIA
ENC28J60 ETHER

8051 SERIAL COMMUNICATION

Types of Serial Communication

- Synchronous serial Data Communication

  Transfer Block of data at a time

- Asynchronous Serial Data Communication

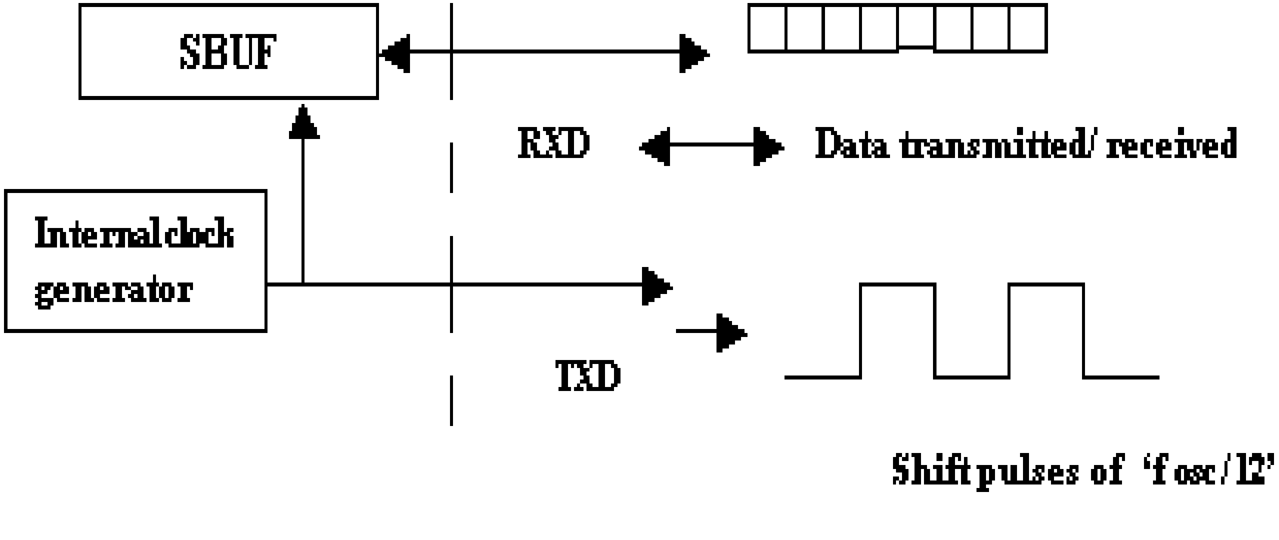
  Transfers single byte at a time

Half Duplex Data Transfer(One way at a time)

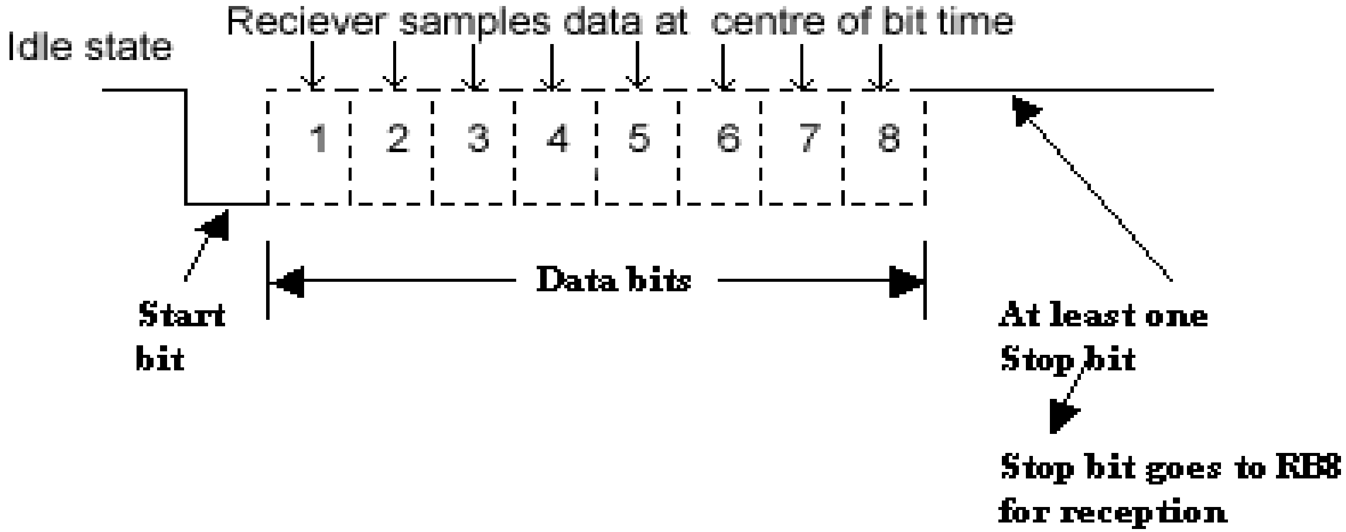
Full Duplex Data Transfer(Both way at a time)

Serial Data Transmission Modes:

- Mode-0**:** In this mode, the serial port works like a shift register and the data transmission works synchronously with a clock frequency of f_{osc} /12. Serial data is received and transmitted through RXD. 8 bits are transmitted/ received at a time.



- Mode-1, the serial port functions as a standard Universal Asynchronous Receiver Transmitter (UART) mode. 10 bits are transmitted through TXD or received through RXD. The 10 bits consist of one start bit (which is usually '0'), 8 data bits (LSB is sent first/received first), and a stop bit (which is usually '1').
- The following figure shows the way the bits are transmitted/ received.


Reciever samples data at centre of bit time
Idle state
1
2
3
4
5
6
7
8
Data bits
Start bit
At least one Stop bit
Stop bit goes to RB8 for reception

Serial Data Mode-2 - Multiprocessor Mode :

- In this mode 11 bits are transmitted through TXD or received through RXD. The various bits are as follows: a start bit (usually '0'), 8 data bits (LSB first), a programmable 9th (TB8 or RB8)bit and a stop bit (usually '1').
- While transmitting, the 9th data bit (TB8 in SCON) can be assigned the value '0' or '1'. For example, if the information of parity is to be transmitted, the parity bit (P) in PSW could be moved into TB8. On reception of the data, the 9th bit goes into RB8 in 'SCON', while the stop bit is ignored.

Mode-3 - Multi processor mode with variable baud rate

- In this mode 11 bits are transmitted through TXD or received through RXD. The various bits are: a start bit (usually '0'), 8 data bits (LSB first), a programmable 9 th bit and a stop bit (usually '1').
- Mode-3 is same as mode-2, except the fact that the baud rate in mode-3 is variable (i.e., just as in mode-1).
- $f_{baud} = (2^{SMOD} /32) * (f_{osc} / 12 (256\text{-}TH1))$.
- This baudrate holds when Timer-1 is programmed in Mode-2.

Applications of microcontroller

►Personal information products: Cell phone, pager, watch, pocket recorder, calculator

►Laptop components: mouse, keyboard, modem, fax card, sound card, battery charger

►Home appliances: door lock, alarm clock, thermostat, air conditioner, TV remote, VCR, small refrigerator, exercise equipment, washer/dryer, microwave oven

►Industrial equipment: Temperature/pressure controllers, Counters, timers, RPM Controllers

►Toys: video games, cars, dolls, etc.

Any Questions ?

Thank You

Contact address

Prof.Mahesh Yanagimath **M.Tech (VLSI &ES), MIEEE, MISTE.**

Faculty,Dept of Elertrical and Electronics Engg

Staff Advisor, IEEE Student Branch

Hirasugar Institute of Technology, Nidasoshi.

Mail Address: maheshyanagimath.eee@hsit.ac.in

Contact No: 09341449466