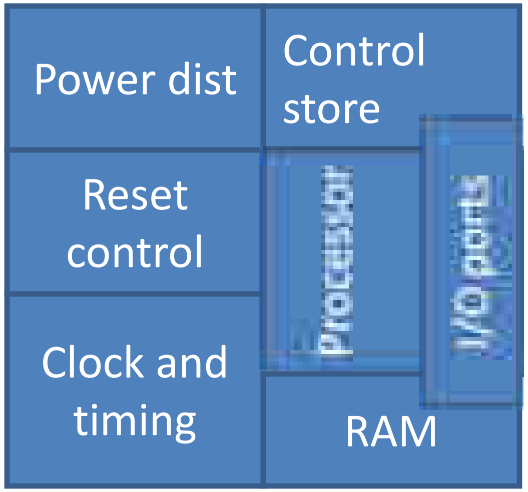
Microcontroller 8051

- An embedded microcontroller is a chip which is a computer processor with all it’s support functions (clocking and reset), memory, and i/O built into the device.



Microcontroller block diagram

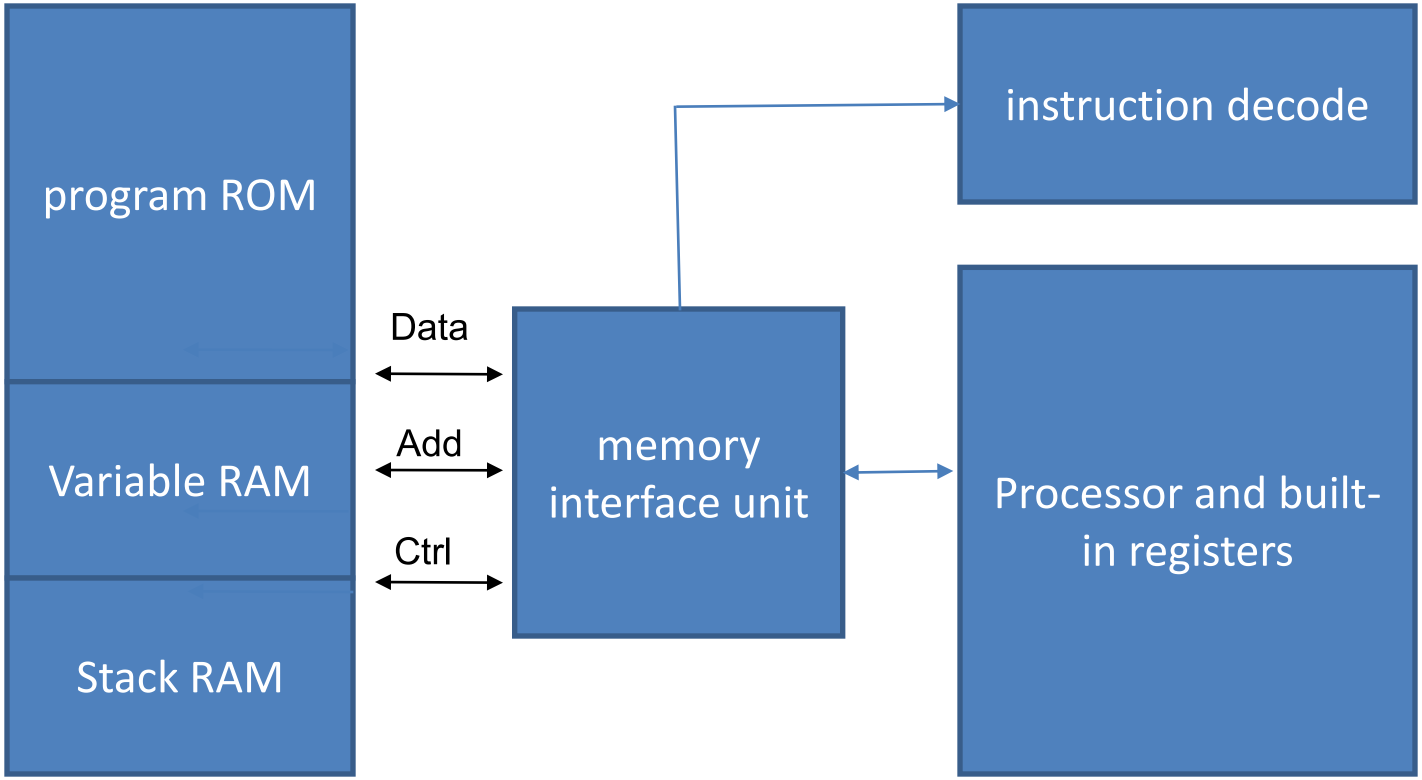
types of microcontrollers

- Embedded
  - All the hardware required to run the application is provided on the chip. typically: power, reset, clock, memory and IO.
- External memory
  - some microcontrollers allow the connection of external memory.

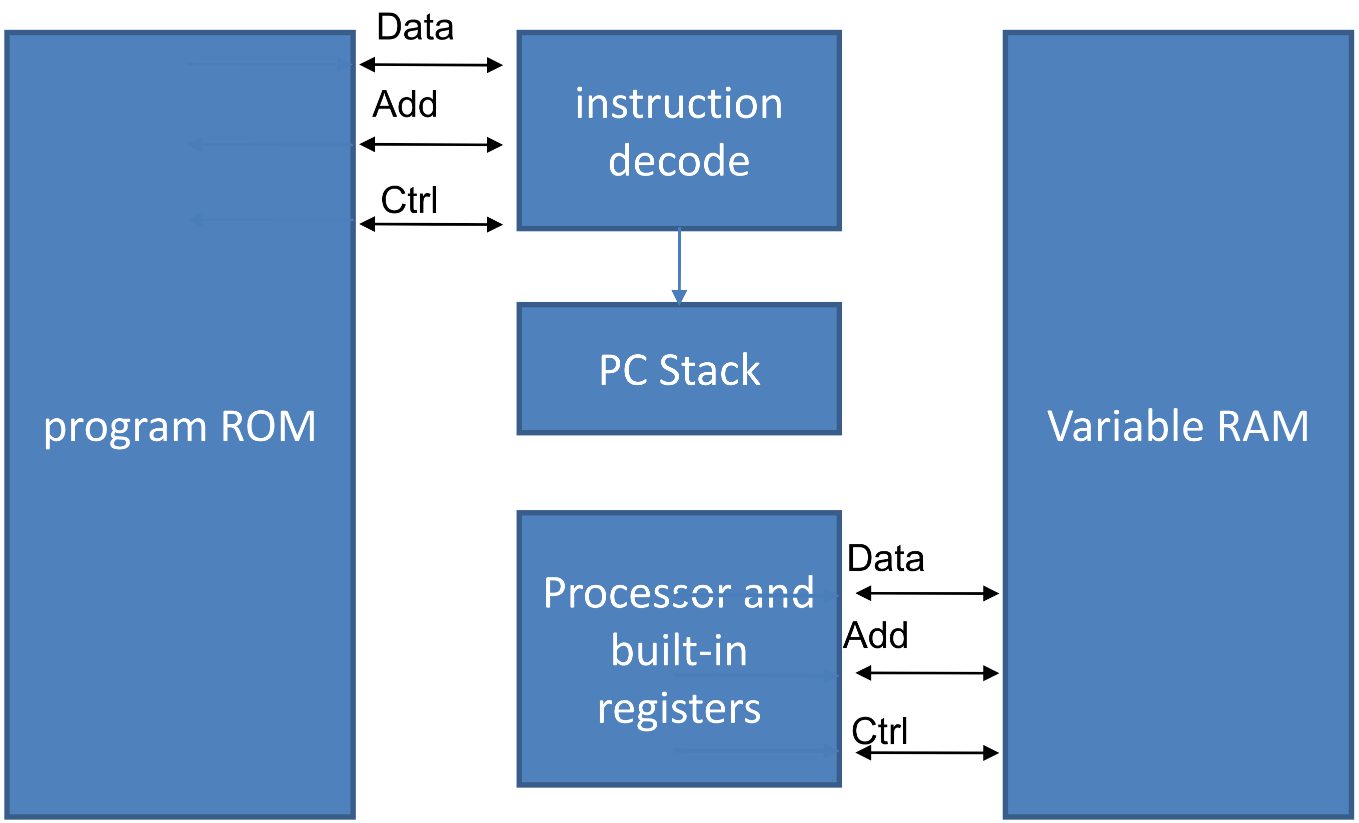
Processor Architecture

- Harvard and Princeton
  - US govt asked for computer to be used with naval shell distance for varying elevations and environmental conditions.
  - Princeton provided ‘Von Neumann’ architecture where common memory space are used for storing program and data. Memory unit is responsible for arbitrary access to memory space between reading instructions and passing data back and forth with processor and its internal registers.
    - Advantages: simple memory interfacing and management.
  - Harvard proposes a design that used separate memory banks for program storage, the processor stack, and variable RAM.
    - Advantage: execute instruction in fewer cycles than Von Neumann.

Princeton architecture block diagram



Harvard architecture block diagram



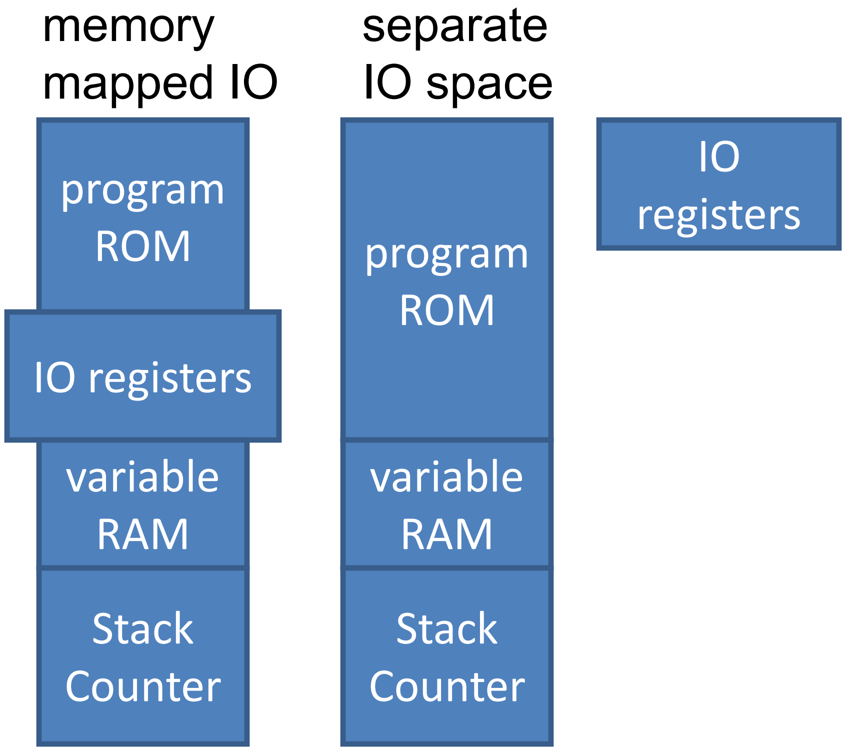
- CISC versus RISC
  - RISC stands for “Reduced Instruction Set Computers”. Instructions are as bare a minimum as possible to allow users to design their own operations.
  - CISC stands for “Complex Instruction Set Computers”. Large number of instructions, each carrying out a different permutation of the same operation.

Microcontroller memory types

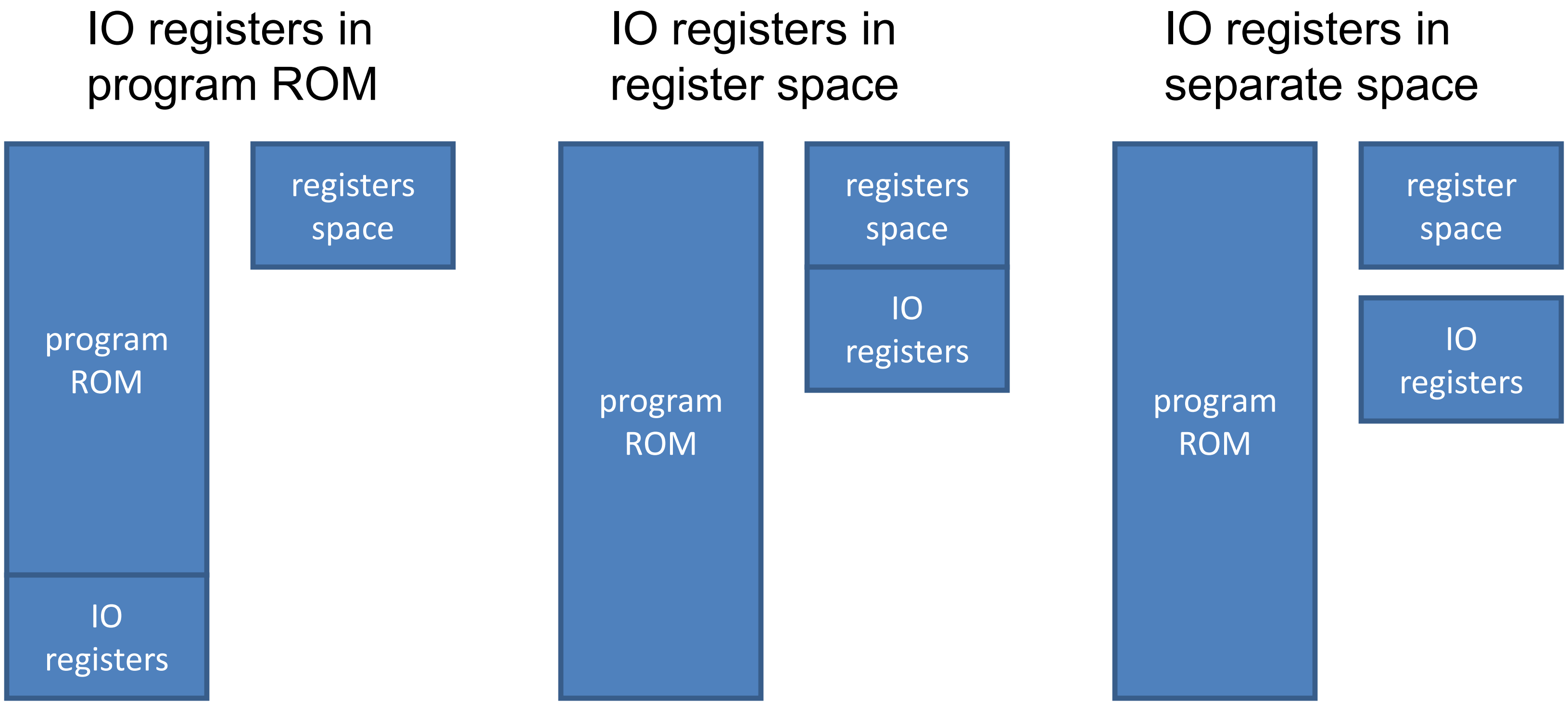
- Control store
  - program memory or firmware. this memory space is the maximum size of the application that can be loaded into the microcontroller and that the application also includes all the low-level code and device interface necessary to execute an application.
  - nonvolatile
  - 8051 has 5 different types of control store : none, mask ROM, PROM, EPROM and EEPROM/Flash

- Variable area (RAM)
  - 4 types variable data storage: bits, registers, variable RAM, PC stack.
  - in 8051 they are implemented as SRAM.
  - program counter stack
    - part of the RAM.
    - LIFO memory.
    - must be initialized by the starting address of the stack area.

- Hardware interface registers (I/O space)
  - could be memory mapped or IO mapped.
  - mostly in variable memory space.
  - IO in Princeton architecture



- IO in Harvard architecture



IO registers in program ROM

IO registers in register space

IO registers in separate space

Microcontroller features

- Clock/Oscillator
- IO pins
- interrupts
- timers

Basic features

- Peripherals
    - ADC inputs
    - DAC outputs
    - PWM outputs

Comparing µC with µP

- General-purpose microprocessors contains
  - No RAM
  - No ROM
  - No I/O ports

➢ Have the advantage of versatility on the amount of RAM, ROM, and I/O ports

- Microcontroller has
  - CPU (microprocessor)
  - RAM
  - ROM
  - I/O ports
  - Timer
  - ADC and other peripherals

➢ The fixed amount of on-chip ROM, RAM, and number of I/O ports and less computing power; suitable for very specific purpose with much less cost.

Applications

- Home
  - Appliances, intercom, telephones, security systems, garage door openers, answering machines, fax machines, TVs, cable TV tuner, VCR, camcorder, remote controls, video games, cellular phones, musical instruments, sewing machines, lighting control, paging, camera, pinball machines, toys, exercise equipment.
- Office
  - Telephones, security systems, fax machines, microwave, copier, laser printer, color printer, paging.
- Auto
  - Navigation system, engine control, air bag, ABS, instrumentation, security system, transmission control, entertainment, climate control, cellular phone, keyless entry.

Examples of 8-bit µC

- Motorola’s 6811
- Intel’s 8051
- Zilog’s Z8
- Microchip’s PIC

The 8051 family has the largest number of diversified (multiple source) suppliers:

- ○ Intel (original)
- ○ Atmel
- ○ Philips/Signetics
- ○ AMD
- ○ Infineon (formerly Siemens)
- ○ Matra
- ○ Dallas Semiconductor/Maxim

8051 µC features

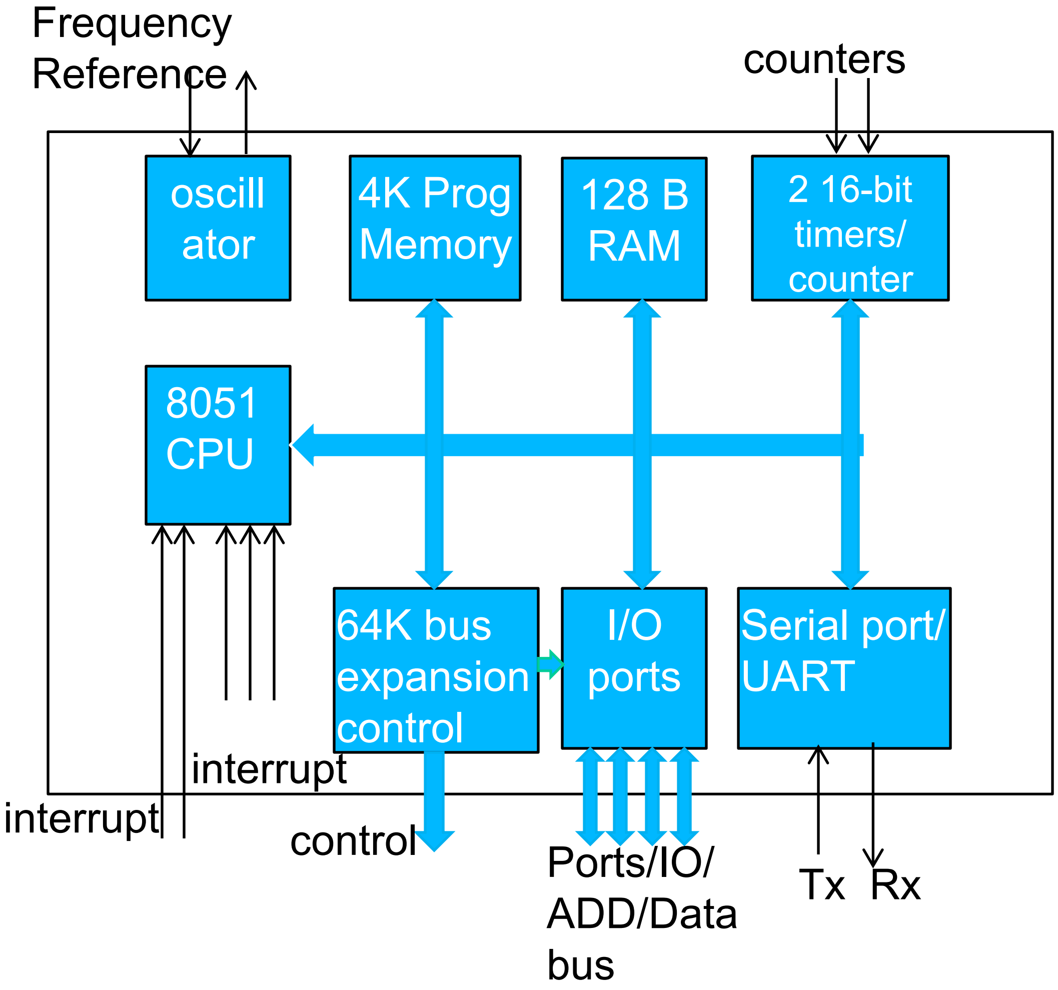
Intel introduced 8051, referred as MCS-51, in 1981

- The 8051 is an 8-bit processor
    - The CPU can work on only 8 bits of data at a time
    - 1 to 16 MHz clock
- The 8051 has
    - 128 bytes of RAM
    - 4K bytes of on-chip ROM
    - Two timers
    - One serial port
    - Four I/O ports, each 8 bits wide
    - 2 external and 3 internal interrupt sources

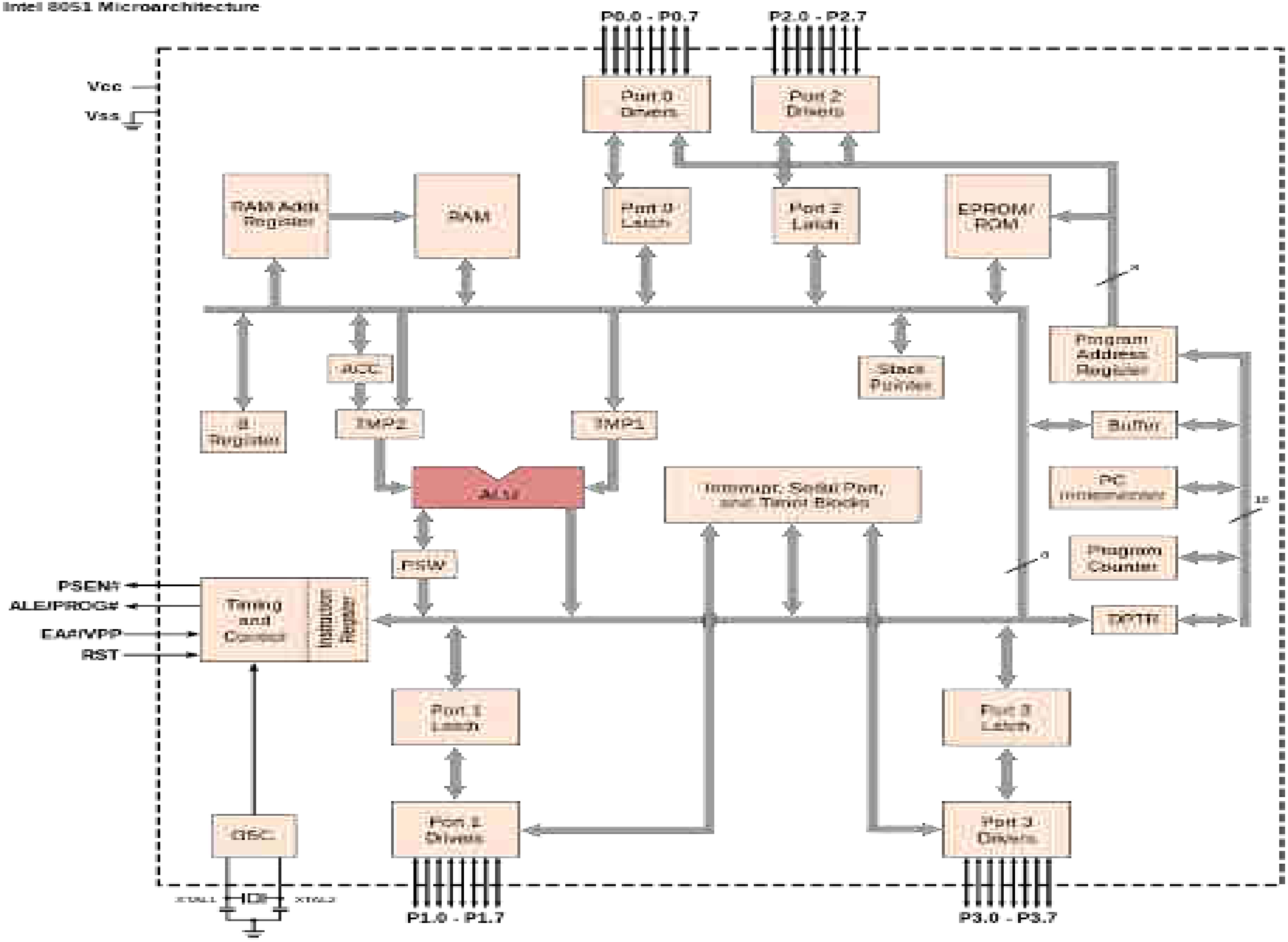
contd.

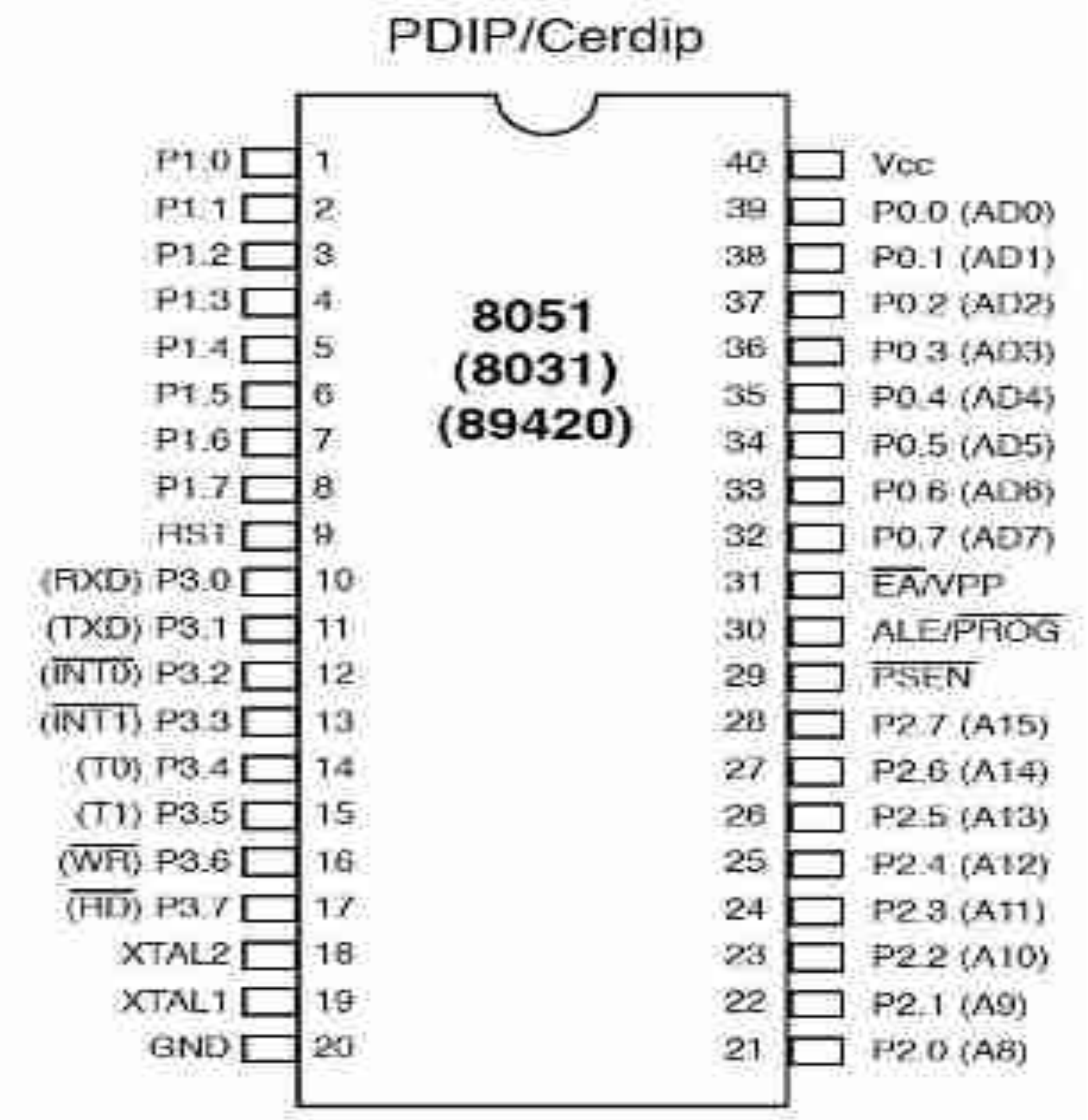
- 8051 instruction cycle consists of 12 clock cycles.
- Application should be run using slower clock speed to reduce power consumption.
- Dallas version of 8051 is 87C51 has EPROM as control store and CMOS device:
    - 24Mhz
    - 12 cycle per instruction
    - 4Kbyte of Control stote
    - 128 bytes of RAM
    - 32 I/O lines
    - Two 8/16-bit times
    - Multiple internal and external interrupts sources
    - Programmable serial ports
    - Interface upto 128Kbytes of external memory

8051 Block Diagram

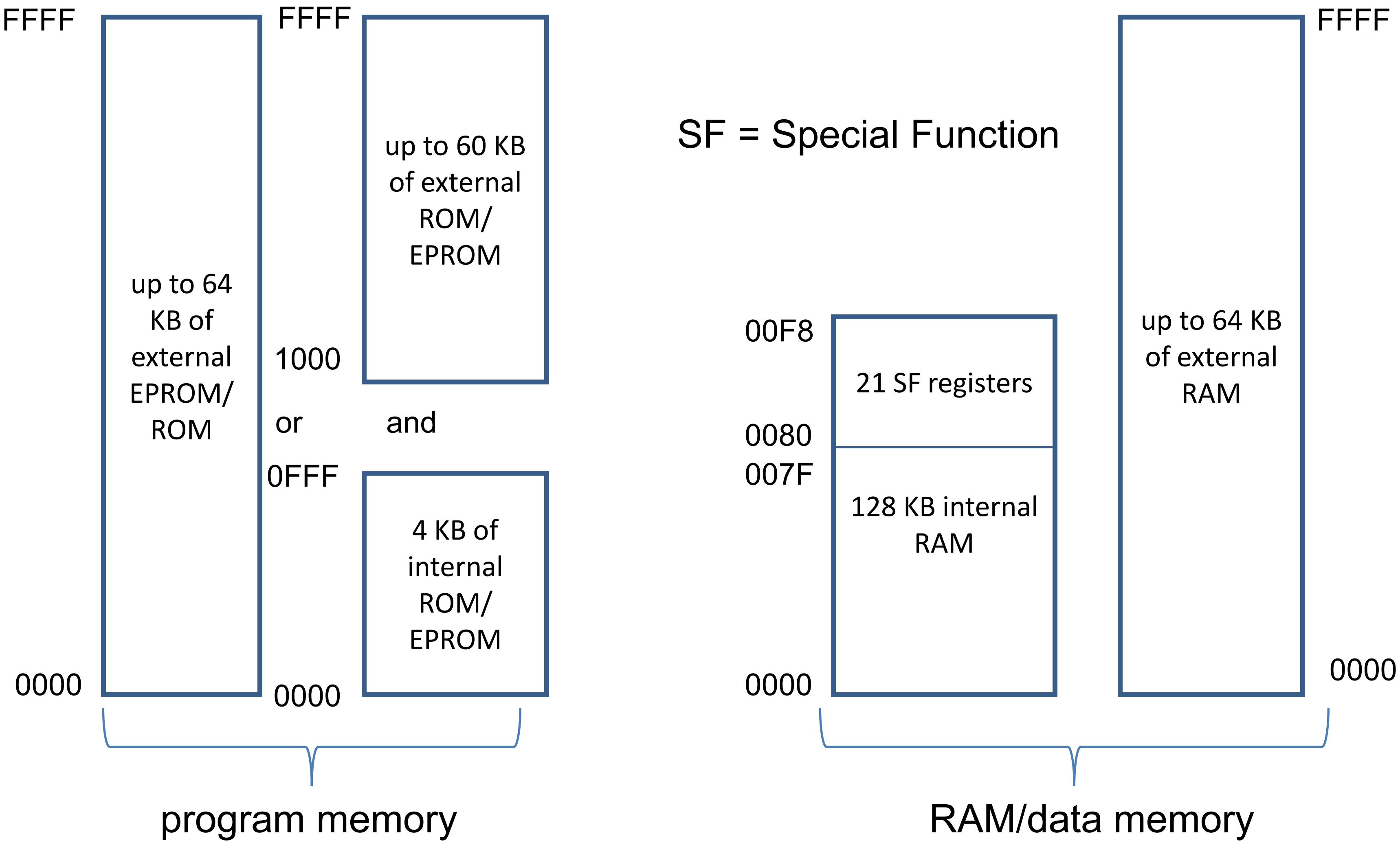


Intel 8051 Microarchitecture




PDIP/Cerdip
8051
(8031)
(89420)
P1.0 1
P1.1 2
P1.2 3
P1.3 4
P1.4 5
P1.5 6
P1.6 7
P1.7 8
RS1 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)

8051 memory-register map



FFFF

up to 64 KB of external EPROM/ ROM

0000

FFFF

up to 60 KB of external ROM/ EPROM

1000

or

and

0FFF

4 KB of internal ROM/ EPROM

0000

program memory

SF = Special Function

00F8

21 SF registers

0080

007F

128 KB internal RAM

0000

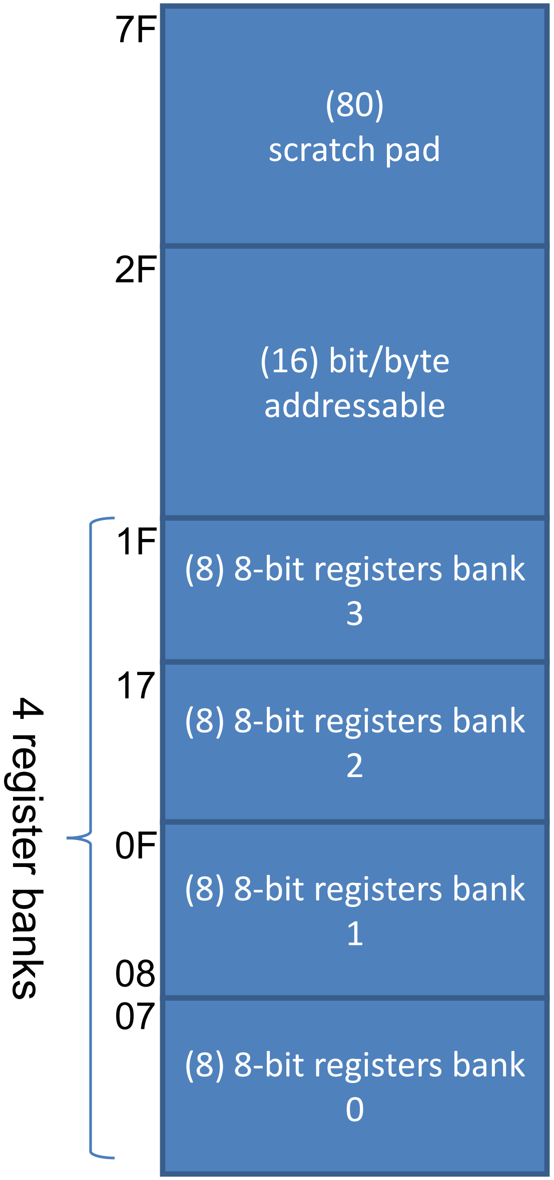
FFFF

up to 64 KB of external RAM

0000

RAM/data memory

128 bytes internal RAM



7F

(80)
scratch pad

2F

(16) bit/byte
addressable

1F

(8) 8-bit registers bank
3

17

(8) 8-bit registers bank
2

0F

(8) 8-bit registers bank
1

08
07

(8) 8-bit registers bank
0

4 register banks

F0H	F7							F0	B
E0H	E7							E0	ACC
D0H	D7							D0	PSW
B8H	BF							B8	IP
B0H	B7							B0	P3
A8H	AF							A8	IE
A0H	A7							A0	P2
99H									SBUF
98H	9F							98	SCON
90H	97							90	P1
8DH									TH1
8CH									TH0
8BH									TL1
8AH									TL0
89H									TMOD

88H	8F							88	TCON
87H									PCON
83H									DPH
82H									DPL
81H									SP
80H	87							80	P0

CY	AC	F0	RS1	RS0	OV		P	PSW

CY – carry flag

AC – auxiliary carry

F0 – general purpose indicator

RS1 – register bank selector bit 1

RS0 – register bank selector bit 0

About RAM

- 128 bytes of internal RAM (00H to 7FH)is general R/W storage.
- Part of this RAM is used as general purpose registers.
- 21 Special-Function Registers (SFR) which are not part of 128 bytes of RAM at 80H to F8H locations of the RAM space.
- 64 KB External RAM can be used fully in addition to 128 internal RAM.
- Although 8051 normally operates with separate program and data memory space, there are applications where it can be used as one 64 KB of memory. When this is done, 8051 can input a block of data through its serial communication port, load it into memory, and then execute that data as a program.

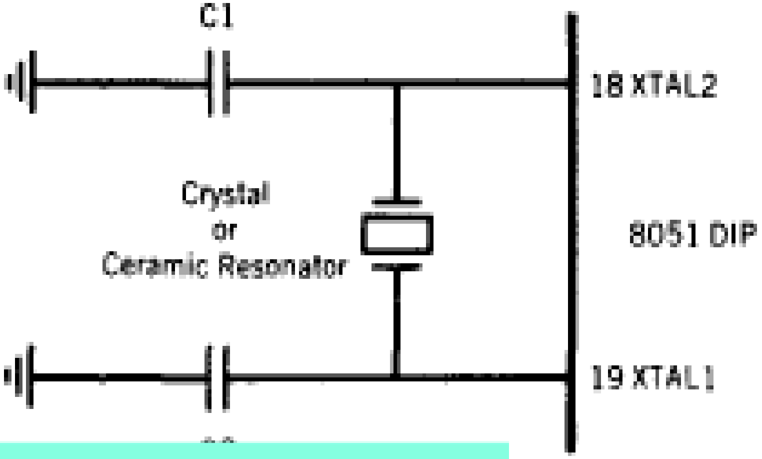
- SFRs are accessed as if they were normal Internal RAM. The only difference is that Internal RAM is from address 00h through 7Fh whereas SFR registers exist in the address range of 80h through F8h.
- B register is used during multiply and divide operations as to hold higher 8-bit source. Otherwise it can used as a simple scratch-pad register.
- ACC and PSW are like microprocessor’s accumulator and flag. PSW does not have a zero flag. RS1 and RS0 indicates the current register bank.
- DPH and DPL is used as 2-byte data pointer DTPR when addressing external memory. Can be used as 8-bit or 16 bit memory pointers.
- SP incremented just before data is stored by using push or call instruction or the interrupt. 8051 SP initialized to 07H on reset. This means first data put on the stack is loaded into memory location 08H.
- The 8051 has four I/O ports of 8 bits, for a total of 32 I/O lines. P0,P1,P2 and P3.
- PCON is power control register. Can put mp into hibernation and conserve power.

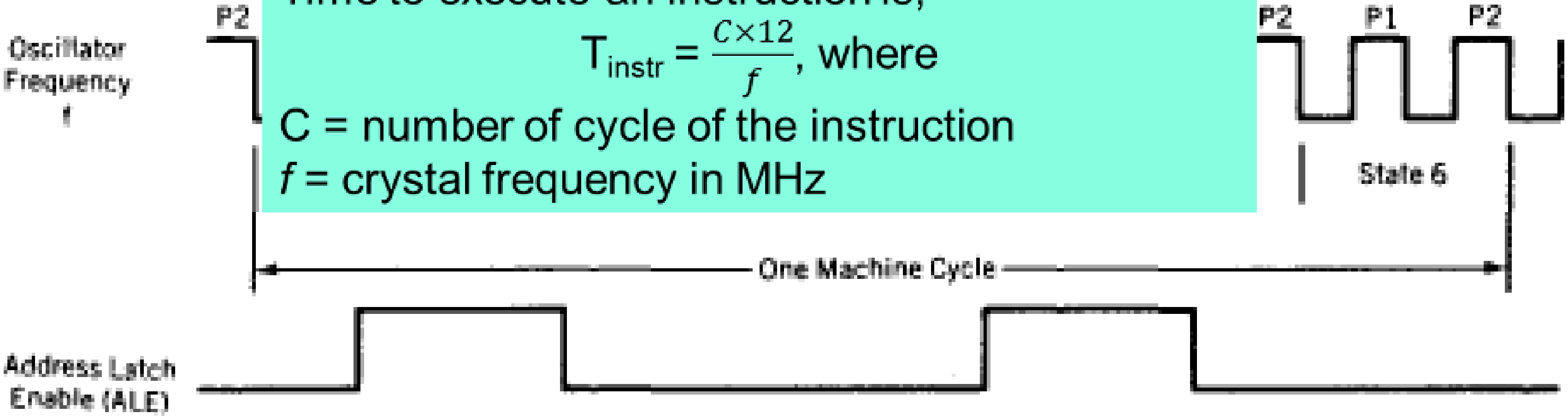
About Program memory

- if more program memory is needed, internal 4 KB memory can be expanded by an additional 60 KB, giving a full 64 KB of program memory space.
- if EA (active low) is asserted, the 8051 does not use the internal 4K ROM. The external memory must start from location 0000H.

Clock/Oscillator

- Clock and communication requ
- Ceramic or crystal?
- State=2 pulses
- Machine cycle=6 states
- Instruction cycle=1/2/4 MC





Time to execute an instruction is,

$T_{instr} = \frac{C\times12}{f}$, where

C = number of cycle of the instruction

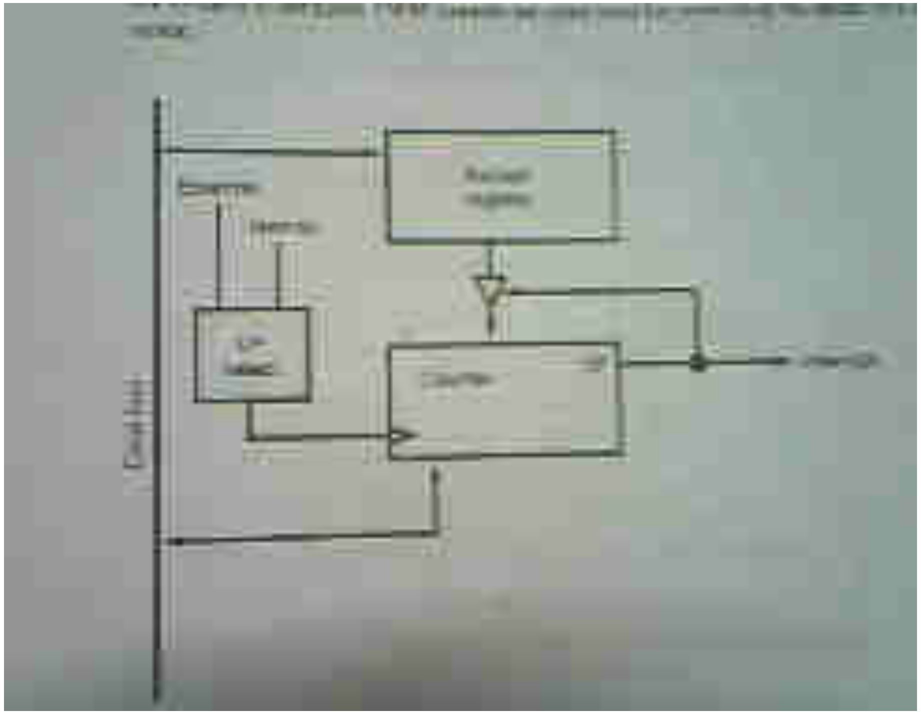
f = crystal frequency in MHz

8051 Timing

Timers/counters

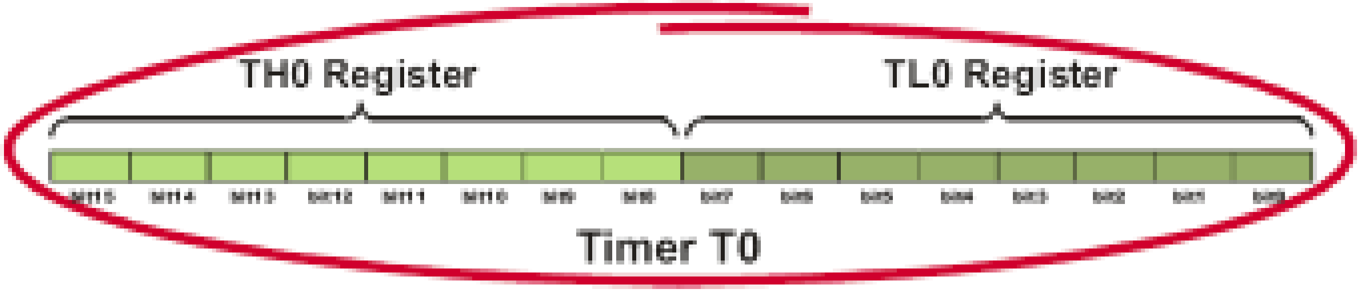
- Event detection, timed control signal generation, counter etc.
- Reads from or written to by the processor and is given by some constant frequency source. Generates an interrupt at the overflow. Run by µC clock or external clock.

$$\text{Interval} = \frac{(\text{CountMax} - \text{Reload})}{\text{ClockFreq}}$$



Timer/counter contd..

- A machine cycle instruction lasts for 12 quartz oscillator periods, which means that by embedding quartz with oscillator frequency of 12MHz, a number stored in the timer register will be changed million times per second, i.e. each microsecond.
- 2 timers/counters called T0 and T1



- If the timer contains for example number 1000 (decimal), then the TH0 register (high byte) will contain the number 3, while the TL0 register (low byte) will contain decimal number 232.
- Formula: TH0 × 256 + TL0 = T
so, 3 × 256 + 232 = 1000

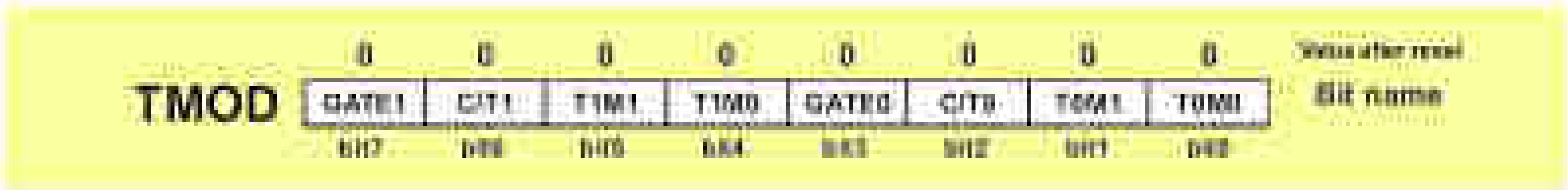
TH0 = 1000/256 = 3 (00000011)
TL0 = 1000 – 3x256 = 232 (11101000)

Timer/counter contd..

- The largest value it can store is 65 535
- In case of exceeding this value, the timer will be automatically cleared and counting starts from 0. This condition is called an overflow.

TMOD Register (Timer Mode)

- There are 4 operational modes.



	0	0	0	0	0	0	0	0	Value after reset
TMOD	GATE1	C/T1	T1M1	T1M0	GATE0	C/T0	T0M1	T0M0	Bit name
	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	

- The low 4 bits (bit0 - bit3) refer to the timer 0, while the high 4 bits (bit4 - bit7) refer to the timer 1.

GATE1 enables and disables Timer 1 by means of a signal brought to the INT1 pin (P3.3):

1 - Timer 1 operates only if the INT1 bit is set.

0 - Timer 1 operates regardless of the logic state of the INT1 bit.

C/T1 selects pulses to be counted up by the timer/counter 1:

1 - Timer counts pulses brought to the T1 pin (P3.5).

0 - Timer counts pulses from internal oscillator.

T1M1,T1M0 These two bits select the operational mode of the Timer 1.

T1M1	T1M0	MODE	DESCRIPTION
0	0	0	13-bit timer
0	1	1	16-bit timer
1	0	2	8-bit auto-reload
1	1	3	Split mode

GATE0 enables and disables Timer 1 using a signal brought to the INT0 pin (P3.2):

1 - Timer 0 operates only if the INT0 bit is set.

0 - Timer 0 operates regardless of the logic state of the INT0 bit.

C/T0 selects pulses to be counted up by the timer/counter 0:

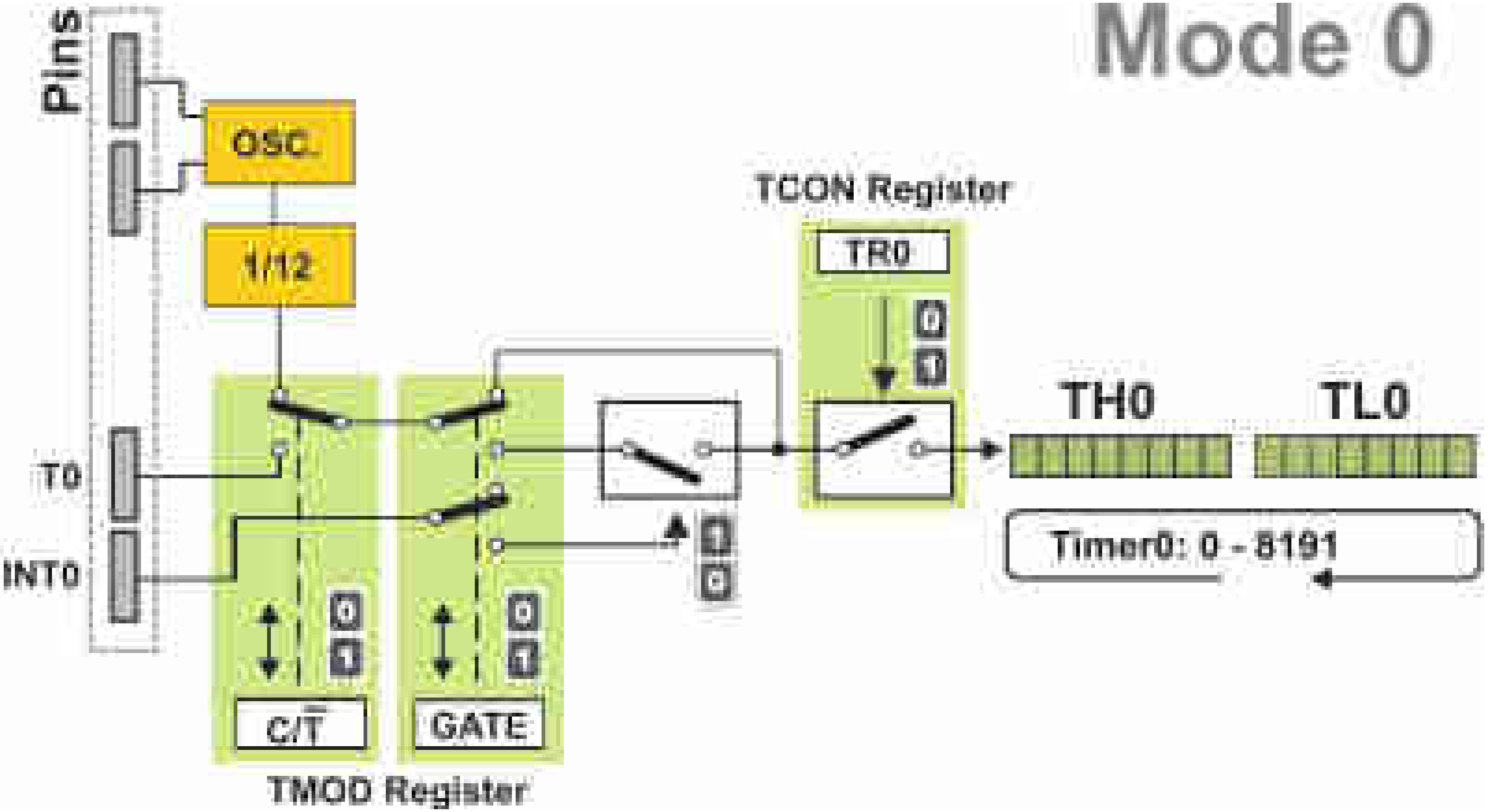
1 - Timer counts pulses brought to the T0 pin (P3.4).

0 - Timer counts pulses from internal oscillator.

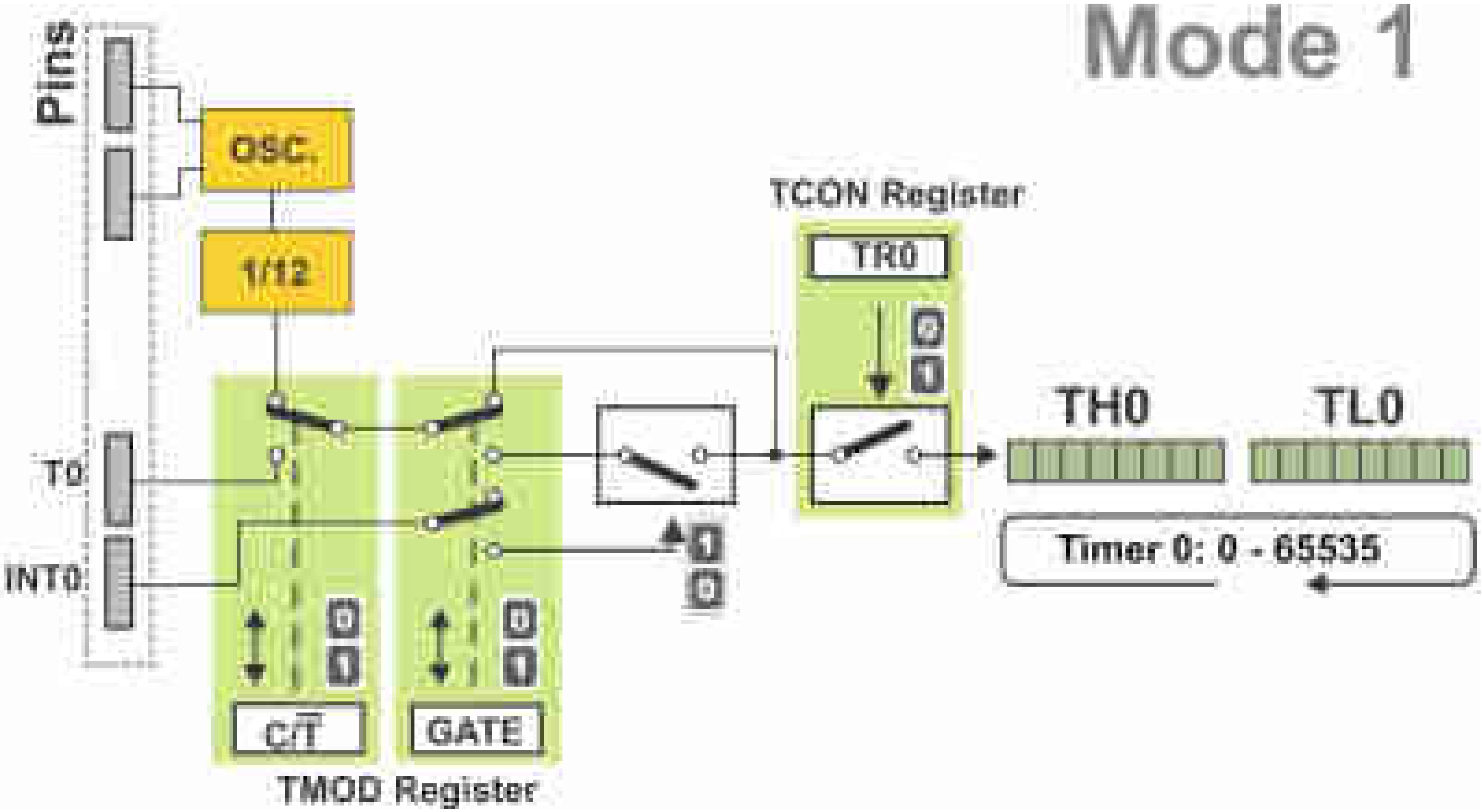
T0M1,T0M0 These two bits select the operational mode of the Timer 0.

T0M1	T0M0	MODE	DESCRIPTION
0	0	0	13-bit timer
0	1	1	16-bit timer
1	0	2	8-bit auto-reload
1	1	3	Split mode

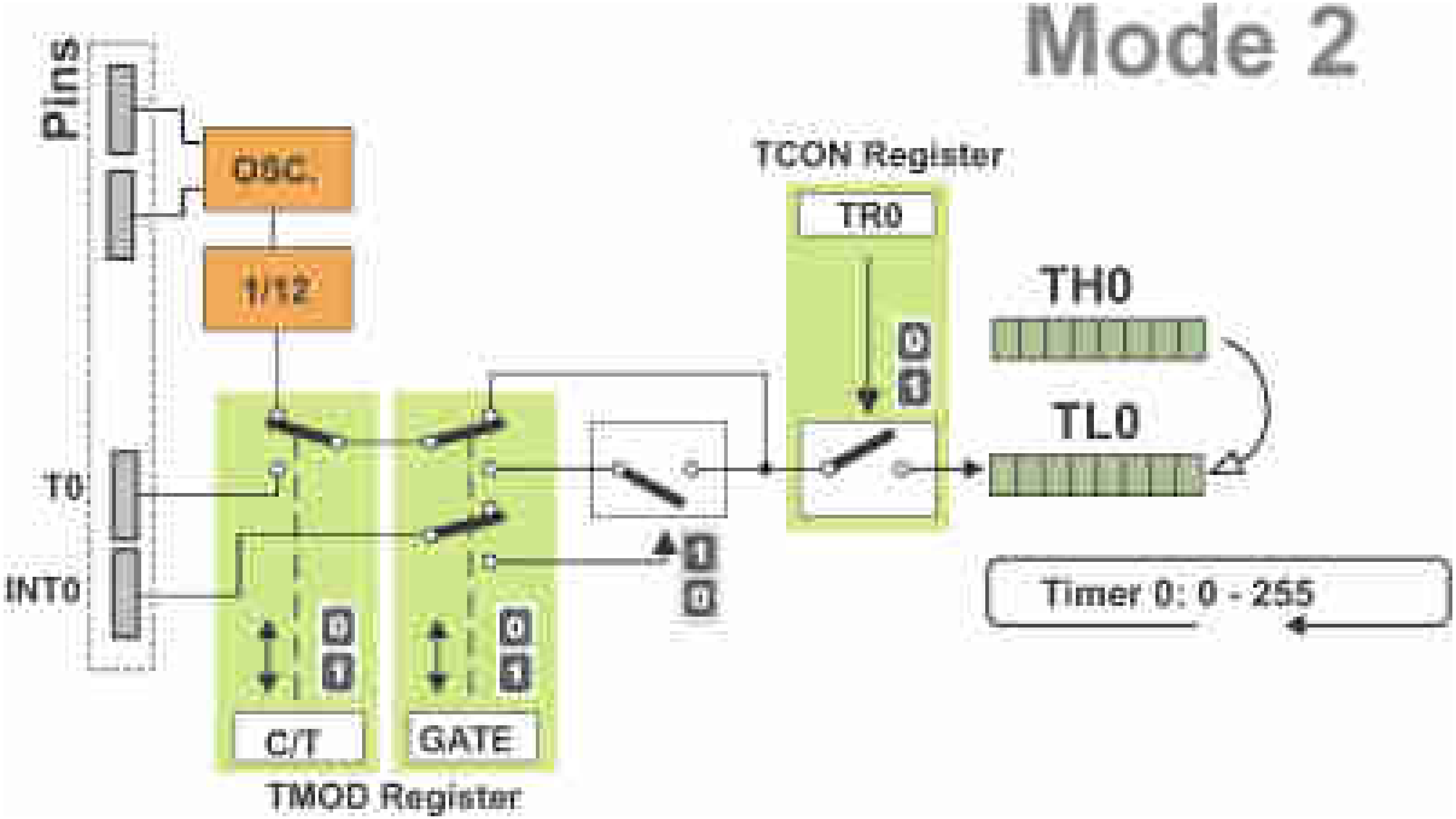
- Timer 0 in mode 0 (13-bit timer):
  - This mode configures timer 0 as a 13-bit timer which consists of all 8 bits of TH0 and the lower 5 bits of TL0. As a result, the Timer 0 uses only 13 of 16 bits. Each coming pulse causes the lower register bits to change their states. After receiving 32 pulses, this register is loaded and automatically cleared, while the higher byte (TH0) is incremented by 1. This process is repeated until registers count up 8192 pulses. After that, both registers are cleared and counting starts from 0.



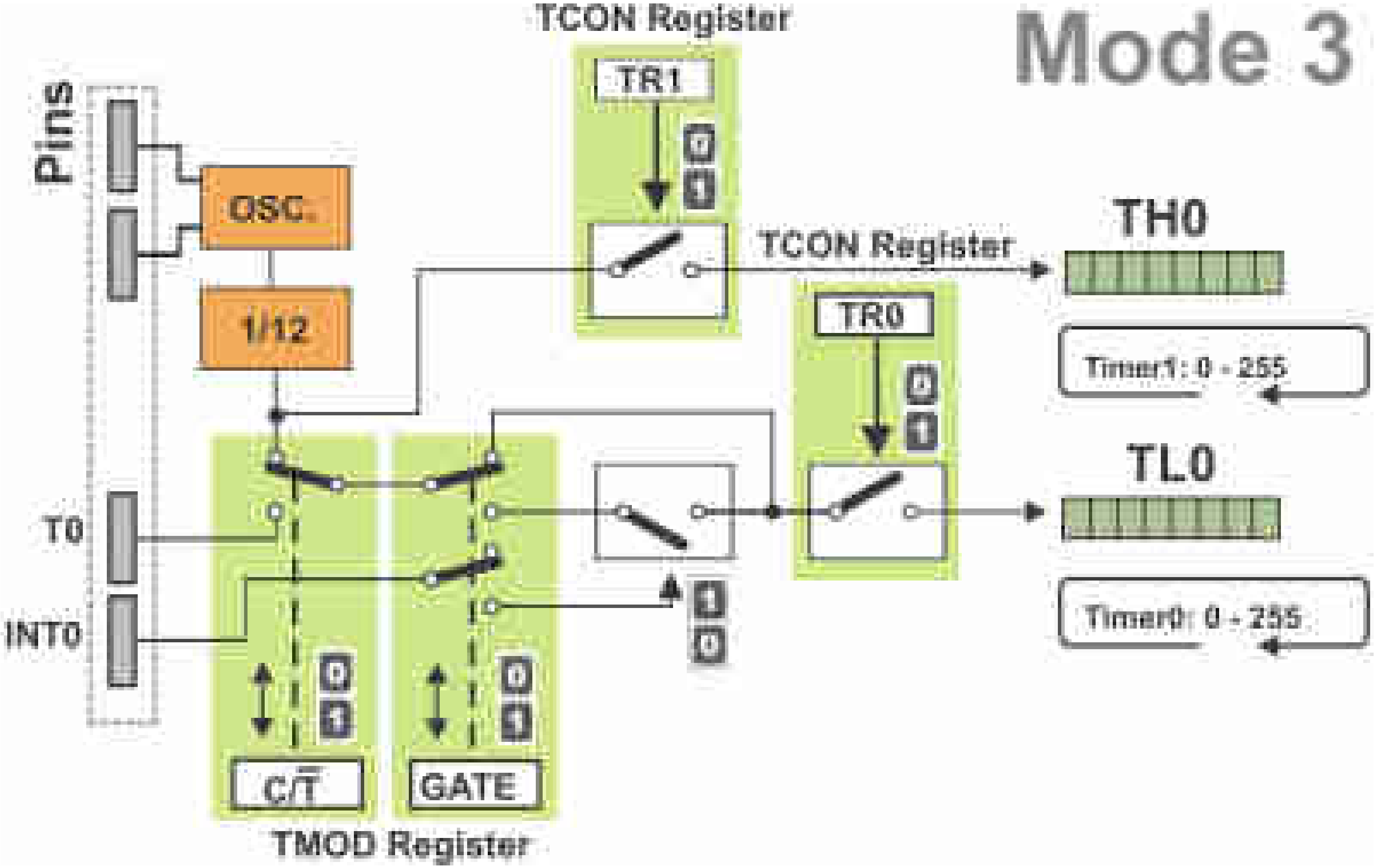
- Timer 0 in mode 1 (16-bit timer)
    - Mode 1 configures timer 0 as a 16-bit timer comprising all the bits of both registers TH0 and TL0. That's why this is one of the most commonly used modes. Timer operates in the same way as in mode 0, with difference that the registers count up to 65 536 as allowable by the 16 bits.



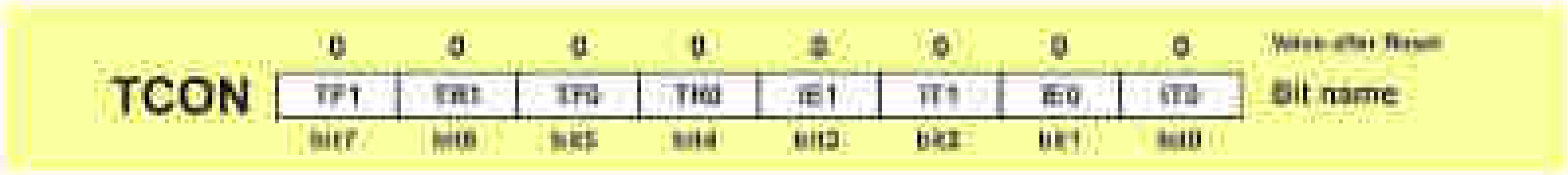
- Timer 0 in mode 2 (Auto-Reload Timer)
  - Mode 2 configures timer 0 as an 8-bit timer. Actually, timer 0 uses only one 8-bit register for counting and never counts from 0, but from an arbitrary value (0-255) stored in another (TH0) register.



- **Timer 0 in Mode 3 (Split Timer)**
  - Mode 3 configures timer 0 so that registers TL0 and TH0 operate as separate 8-bit timers. In other words, the 16-bit timer consisting of two registers TH0 and TL0 is split



Timer Control (TCON) Register

- Only 4 bits of this register are used for this purpose, while rest of them is used for interrupt control.
- 

TCON	0	0	0	0	0	0	0	0	Value after Reset
	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	Bit name
	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	

TF1 bit is automatically set on the Timer 1 overflow.

TR1 bit enables the Timer 1.

1 - Timer 1 is enabled.

0 - Timer 1 is disabled.

TF0 bit is automatically set on the Timer 0 overflow.

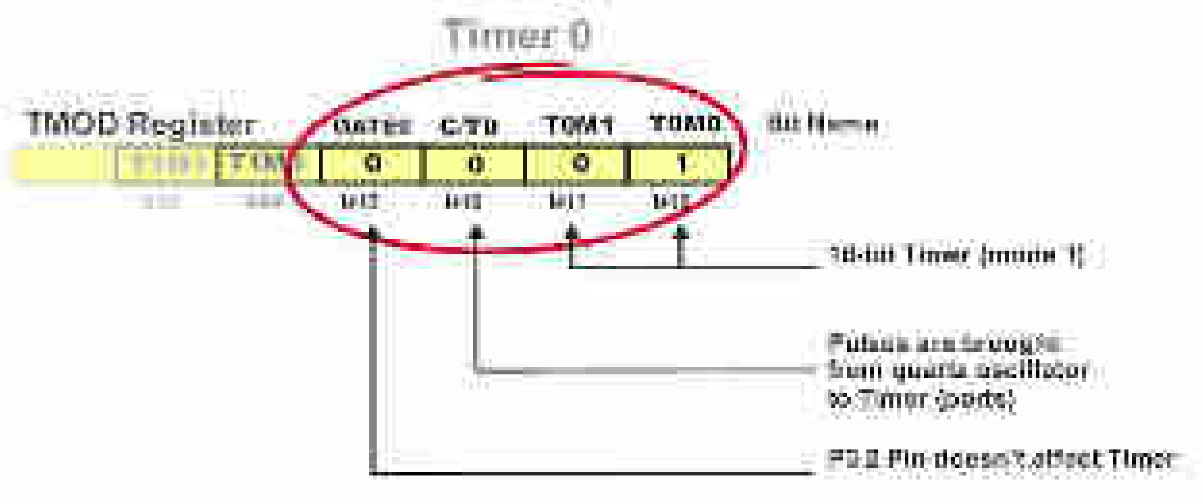
TR0 bit enables the timer 0.

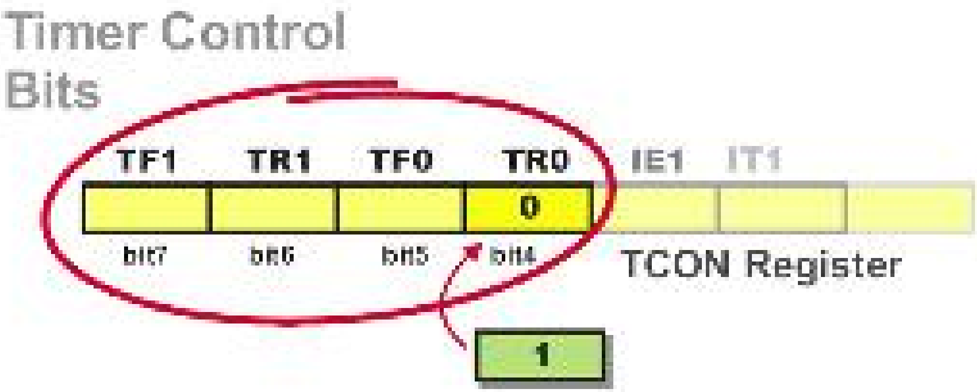
1 - Timer 0 is enabled.

0 - Timer 0 is disabled.

How to use the Timer 0 ?

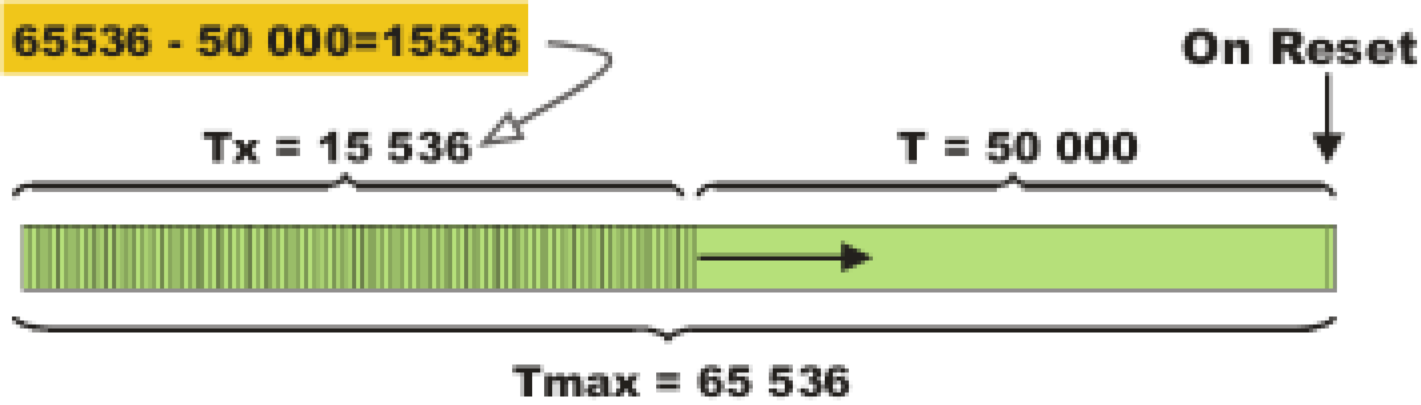
- the timer 0 operates in mode 1 and counts pulses generated by internal clock the frequency of which is equal to 1/12 the quartz frequency.

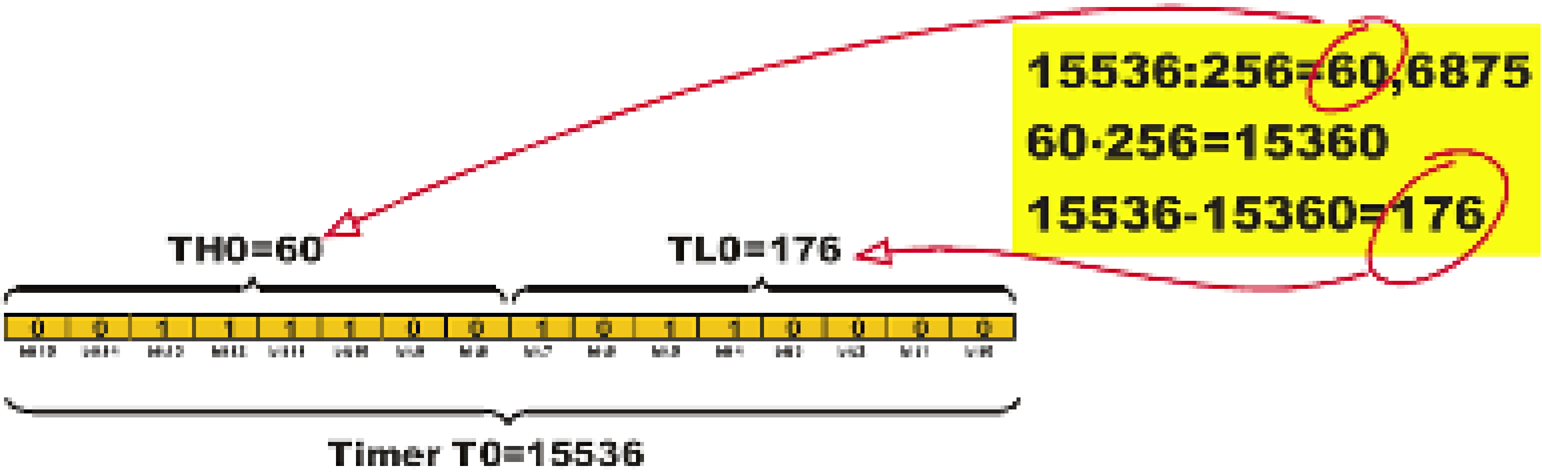

Timer 0
TMOD Register
GATE0
C/T0
T0M1
T0M0
Bit Name
0
0
0
1
bit3
bit2
bit1
bit0
16-bit Timer (mode 1)
Pulses are brought from quartz oscillator to Timer (parts)
P3.2 Pin doesn't affect Timer



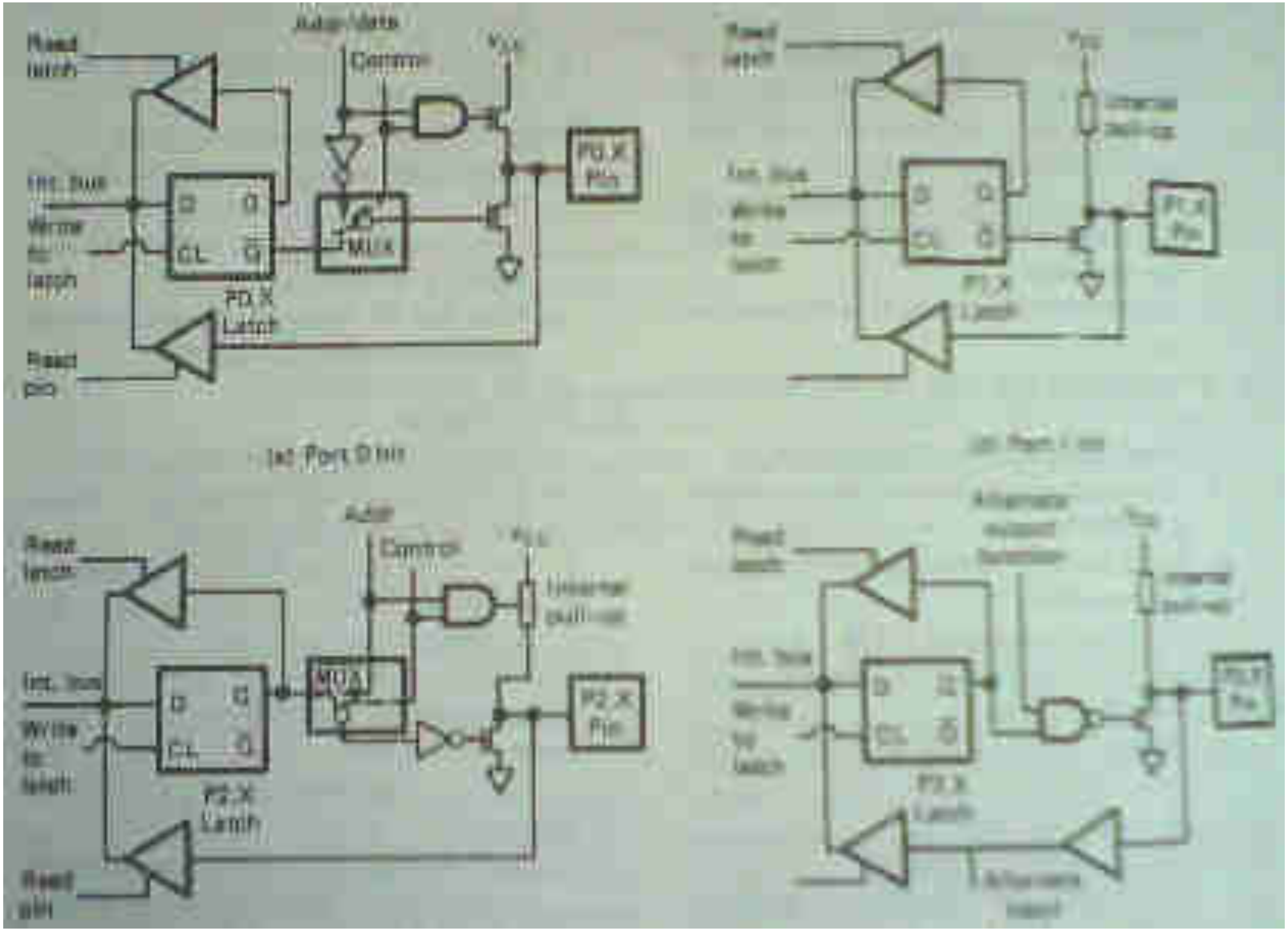
Timer 0 Overflow Detection

- When it occurrs, the TF0 bit of the TCON register will be automatically set. The state of this bit can be constantly checked from within the program or by enabling an interrupt which will stop the main program
-





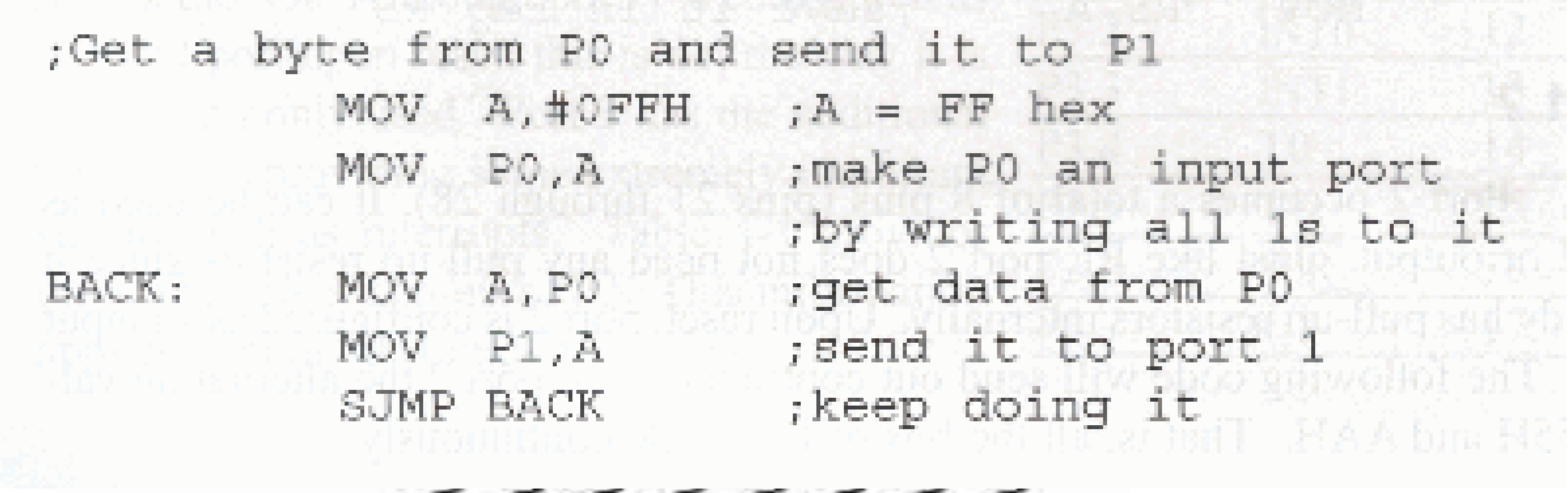
A Pin of IO ports



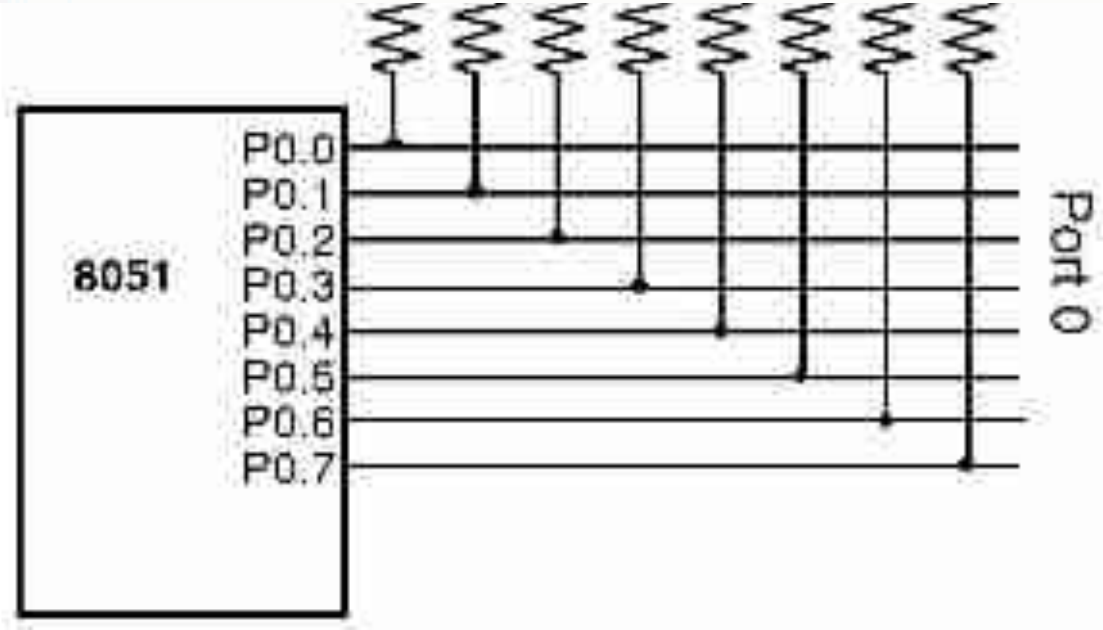
IO ports

- All four 8 bit ports are bidirectional.
- Port latch (D-type FF) allows u to store data going out of the port or coming into the port. The latch can be set by data on the data bus or at the port pin. Also the latch can place data on the mc bus or send it to the port pin. However, port pins may have different value than port latches.
- When 1s are written to port, pins are pulled high (or floats) by the internal pull-ups and can be used as inputs.
- In order to configure a microcontroller pin as an output, it is necessary to apply a logic zero (0) to appropriate I/O port bit. In this case, voltage level on appropriate pin will be 0.

Port 0



```
;Get a byte from P0 and send it to P1
            MOV  A,#0FFH     ;A = FF hex
            MOV  P0,A        ;make P0 an input port
                             ;by writing all 1s to it
BACK:       MOV  A,P0        ;get data from P0
            MOV  P1,A        ;send it to port 1
            SJMP BACK        ;keep doing it
```

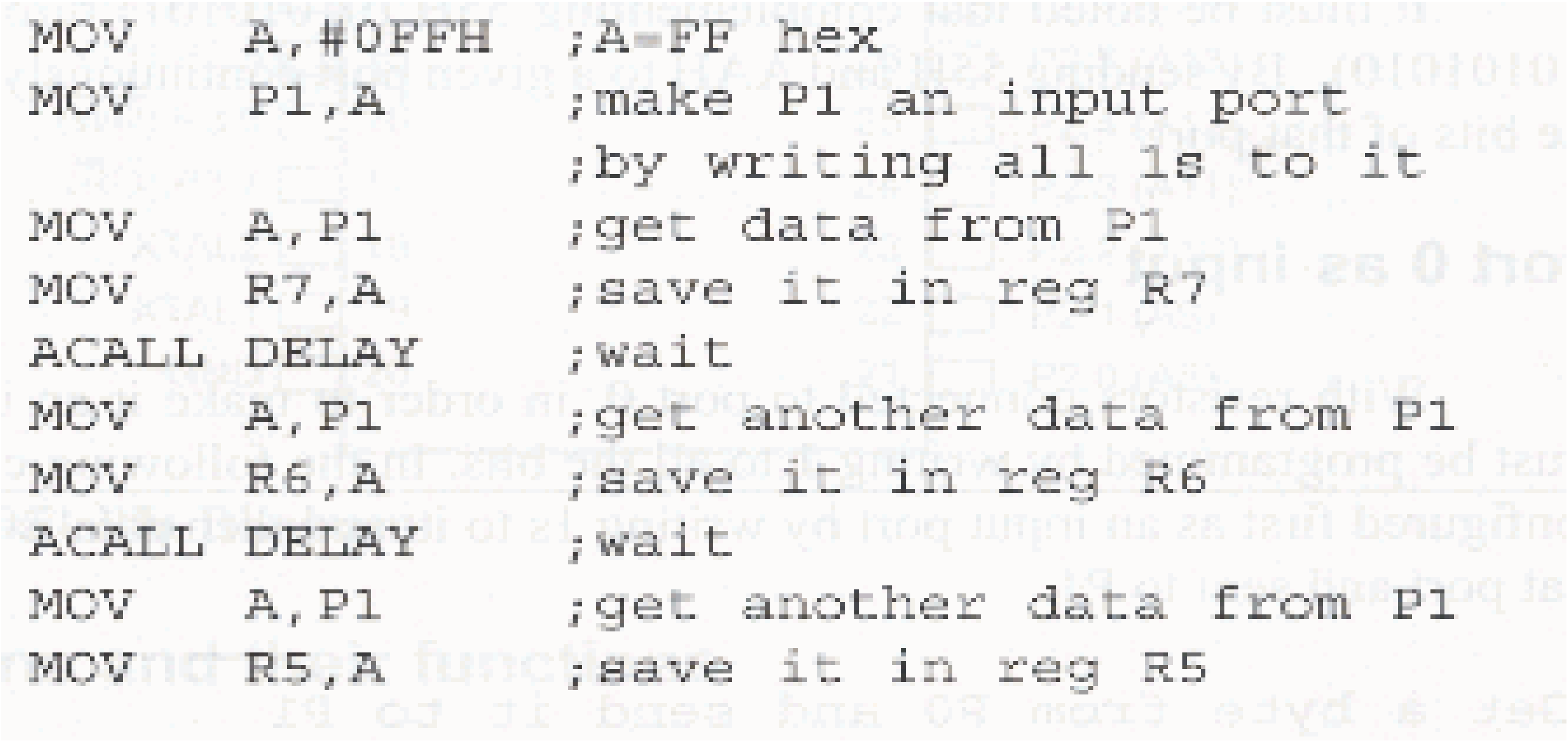


Dual role of P0

- Port 0 and 2 together can be used to address the external memory. Port 0 can also be used to exchange data from the external port. accessing 64K bytes of external memory, it needs a path for the 16 bits of the address. P0 gives lower Byte of Address.

Port 1

- This port does not need any pull-up resistors since it already has pull-up resistors internally.
- If port 1 is configured as an output port, to make it an input port again, it must programmed as such by writing 1 to all its



```
MOV    A,#0FFH    ;A=FF hex
MOV    P1,A       ;make P1 an input port
                  ;by writing all 1s to it
MOV    A,P1       ;get data from P1
MOV    R7,A       ;save it in reg R7
ACALL  DELAY      ;wait
MOV    A,P1       ;get another data from P1
MOV    R6,A       ;save it in reg R6
ACALL  DELAY      ;wait
MOV    A,P1       ;get another data from P1
MOV    R5,A       ;save it in reg R5
```

Port 2

- It can be used as input or output.
- Internal pull-up resistor.
- Upon reset, port 2 is configured as an input port.
- Alternate use
  - provide higher byte address when external memory is connected.

Port 3

- Port 3 can be used as input or output. On Reset input port.
- Has internal pull-up resistors.
- Alternate use
    - providing some extremely important signals such as interrupts, serial I/O, timer/counter and read/write control for external memory.

PIN	ALT. USE		Control SFR
P3.0	RXD	Serial Data In	SBF
P3.1	TXD	Serial Data out	SBF
P3.2	$\overline{\mathrm{INT0}}$	Ext. Interrupt 0	TCON.1
P3.3	$\overline{\mathrm{INT1}}$	Ext. Interrupt 1	TCON.3
P3.4	IT0	Ext. Timer 0 In	TMOD.3
P3.5	IT1	Ext. Timer 1 In	TMOD.7
P3.6	$\overline{\mathrm{WR}}$	Write Control	For ext. memory
P3.7	$\overline{\mathrm{RD}}$	Read Control	

Interrupts

- Two SFRs controls the function of interrupts in 8051 microcontroller.
  - IE, Responsible for disable/enable the function.
  - IP, Responsible for priority assignment:

    The priority list offers 3 levels of interrupt priority:

    - ➢ Reset! The absolute master. When a reset request arrives, everything is stopped and the microcontroller restarts.
    - ➢ Interrupt priority 1 can be disabled by Reset only.
    - ➢ Interrupt priority 0 can be disabled by both Reset and interrupt priority 1.

	0	X	0	0	0	0	0	0	Value after Reset
IE	EA		ET2	ES	ET1	EX1	ET0	EX0	Bit name
	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	

EA - global interrupt enable/disable:

0 - disables all interrupt requests. 1 - enables all individual interrupt requests.

ES - enables or disables serial interrupt:

0 - UART system cannot generate an interrupt. 1 - UART system enables an interrupt.

ET1 - bit enables or disables Timer 1 interrupt:

0 - Timer 1 cannot generate an interrupt. 1 - Timer 1 enables an interrupt.

EX1 - bit enables or disables external 1 interrupt:

0 - change of the pin INT0 logic state cannot generate an interrupt.

1 - enables an external interrupt on the pin INT0 state change.

ET0 - bit enables or disables timer 0 interrupt:

0 - Timer 0 cannot generate an interrupt. 1 - enables timer 0 interrupt.

EX0 - bit enables or disables external 0 interrupt:

0 - change of the INT1 pin logic state cannot generate an interrupt.

1 - enables an external interrupt on the pin INT1 state change.

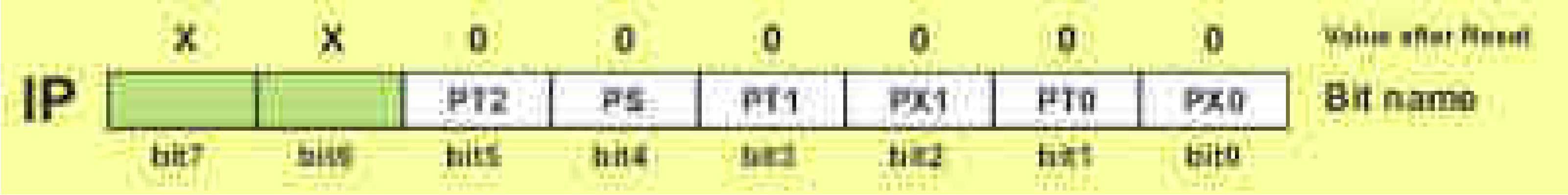
*bit6 is not implemented.

* ET2 is reserved for future use.

Interrupt Priority

If an interrupt of higher priority arrives while an interrupt is in progress, it will be immediately stopped and the higher priority interrupt will be executed first.

- If the both interrupt requests, at the same priority level, occur one after another, the one which came later has to wait until routine being in progress ends.
- If two interrupt requests, at different priority levels, arrive at the same time then the higher priority interrupt is serviced first.
- If two interrupt requests of equal priority arrive at the same time then the interrupt to be serviced is selected according to the following priority list:
    - External interrupt INT0, i.e. IE0
    - Timer 0 interrupt, i.e. TF0
    - External Interrupt INT1, i.e. IE1
    - Timer 1 interrupt, i.e. TF1
    - Serial Communication Interrupt, i.e. RI, TI



IP	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	
	X	X	0	0	0	0	0	0	Value after Reset
			PT2	PS	PT1	PX1	PT0	PX0	Bit name

PS - Serial Port Interrupt priority bit

Priority 0 or Priority 1

PT1 - Timer 1 interrupt priority

Priority 0 or Priority 1

PX1 - External Interrupt INT1 priority

Priority 0 or Priority 1

PT0 - Timer 0 Interrupt Priority

Priority 0 or Priority 1

PX0 - External Interrupt INT0 Priority

Priority 0 or Priority 1

Bit7 and bit6 are not implemented. PT2 is reserved for future use.

Handling Interrupt

When an interrupt request arrives the following occurs:

1. Instruction in progress is ended.
2. The address of the next instruction to execute is pushed on the stack.
3. Depending on which interrupt is requested, one of 5 vectors (addresses) is written to the program counter in accordance to the table below:

INTERRUPT SOURCE	VECTOR (ADDRESS)
IE0	3 h
TF0	B h
IE1	13h
TF1	1B h
RI, TI	23 h
All addresses are in hexadecimal format	

4. When an interrupt routine is executed, the address of the next instruction to execute is poped from the stack to the program counter and interrupted program resumes operation from where it left off.

8051 Instruction Set

- 8 bit opcode, 255 codes implemented
- 111 truly different instructions
  – 49 single bytes, 45 two bytes, and 17 three-bytes.
- 4 different types of instructions.
  1. data transfer instructions.
     - General purpose: MOV, PUSH, POP
     - Accumulator specific: MOVX, MOVC, XCH
     - Address-object transfer: to load 16 bit address data into data pointer, e.g. MOV #data 16
  2. Mathematical operations.
     - Add, subtract, multi, divi, rotate, swap, and, or etc.
  3. Control operations.
     - Call, jump, ret etc.
  4. Bit operations.
     - CLR, SETB, CPL, ANL

Addressing modes

- 8 addressing mode
  1. Bank addressing, MOV A, Rn
     - Rn is R0 to R7 of the currently selected bank.
  2. Direct addressing, MOV A, direct
     - “direct” is an internal data memory location pointed by 8 bit
  3. Register indirect addressing, MOV A, @Ri
     - Ri is either R0 or R1 of the currently selected bank
  4. Immediate addressing
     - 8 bit addressing, MOV A, #data
  5. 16-bit absolute addressing, LCALL *addr 16*
     - Addr 16 is a 16 bit absolute address in 64K locations
  6. 11 bit relative addressing, ACALL *addr 11*
     - Addr 11 is a 11 bit address relative to current PC value
  7. 8 bit offset addressing, JZ *rel*
     - rel is an 8 bit offset added to current PC value.
  8. Bit addressing, SETB bit

Bank Addressing

- Data can be moved among the registers of the current Bank, A and B.
  - MOV A, B
  - MOV A, R0
  - MOV R0, R5
  - MOV R1, A
  - ADD A, R3

Direct Addressing Mode

MOV A, direct

- “direct” is an internal data memory location pointed by 8 bit.
- All 128 bytes of internal RAM and SFRs may be accessed.

MOV A, 7Fh ; /* copy data from RAM location 7F to Accumulator */

MOV A, 0Dh ; /* note leading 0, instead of writing Dh */

Similarly,

MOV address, A

MOV register, address

MOV address, register

MOV address1, address2

MOV address, #value; /* value is 8 bit number */

Indirect Addressing Mode

- Sometimes called register indirect addressing mode.
- MOV A, @Ri
- Uses register R0 or R1 only.

  MOV A, @R1;   /* copy data from memory location pointed to by R1 */

  MOV @R0, #n; /* copy n to memory location pointed to by R0 */

  MOV @R1, address; /*copy between address and address in R1 */

  MOV @Ri, B;

Note: content of R0 or R1 must be an address of RAM locations or SFR.

Invalid instructions:

MOV @Ri, @Rj

Immediate addressing

- Values can be directly loaded to any of the registers A, B or R0 – R7. # is a must.
  - MOV R0, #9Fh
  - MOV A, #0F1h ; /* note preceding 0 of the value */
  - MOV A, #5h ; /* it is actually read as #05h */
  - MOV A, #56;
  - MOV A, #0x15;

External Data Moves

- From External RAM

  MOVX A, @Rp; /* copy data to A from external RAM location in Rp */

  MOVX A, @DPTR; /* copy data from location in DPTR */

  - Must involve A register.
  - Rp can address 256 bytes
  - DPTR can address 64K bytes

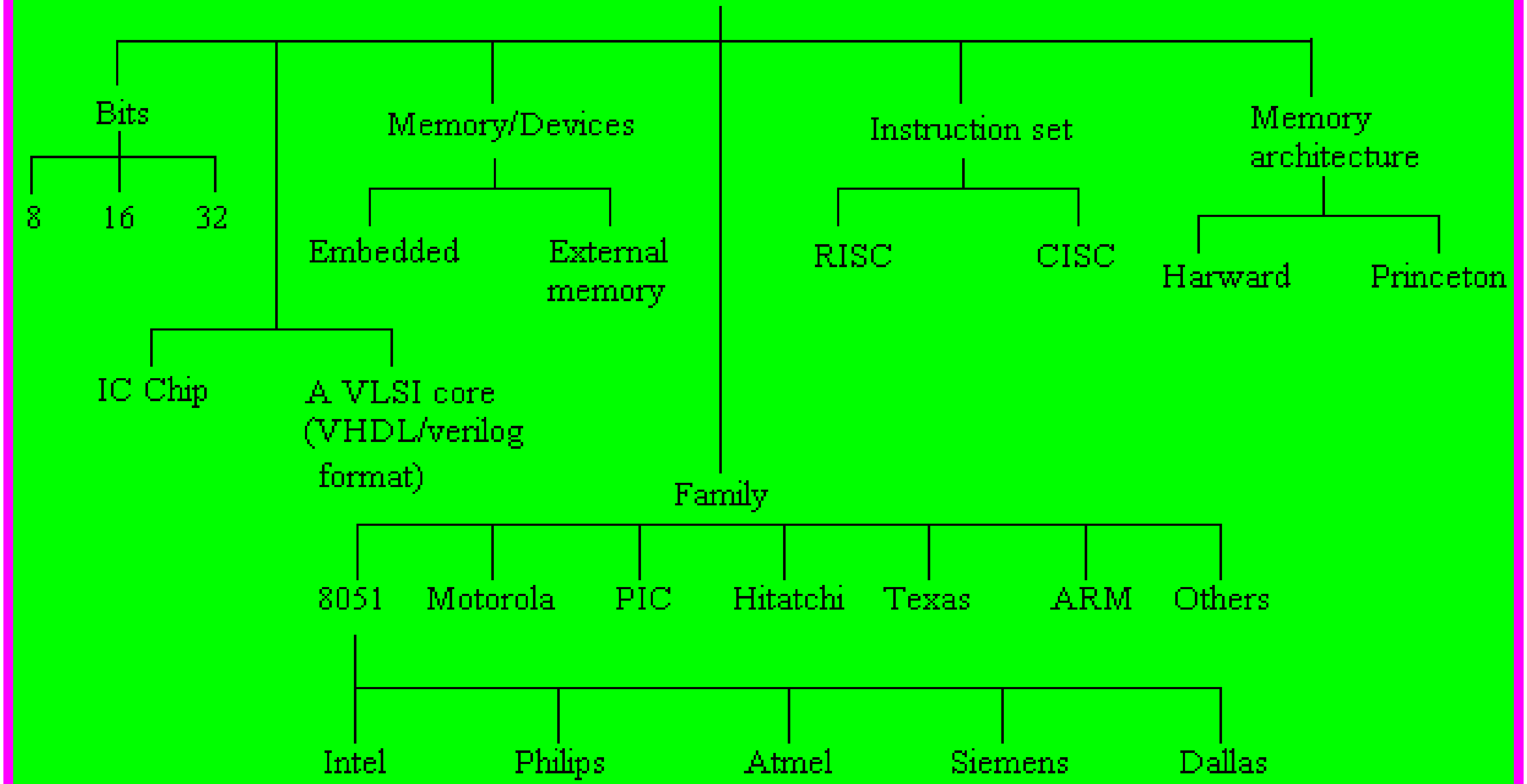
- From external code memory Read-only moves

  MOVC A, @A+DPTR; /* copy the code byte found at address formed by adding A and DPTR, to A */

  MOVC A, @A+PC ; /* address formed by A and program counter */

Note: must involve A register

TYPES OF MICROCONTROLLERS



Bits

8 16 32

IC Chip

A VLSI core (VHDL/verilog format)

Memory/Devices

Embedded

External memory

Family

8051

Intel

Philips

Atmel

Siemens

Dallas

Motorola

PIC

Hitatchi

Texas

ARM

Others

Instruction set

RISC

CISC

Memory architecture

Harward

Princeton

Code Writing Style

```
       ORG 0H; means program starts at 0000H location
Again: MOV R5, # 25H; load 25H to R5
       ADD A, R5; Add the R5 with Acc
       SJMP Again; short jump to again
       END;
```

Types of Instruction

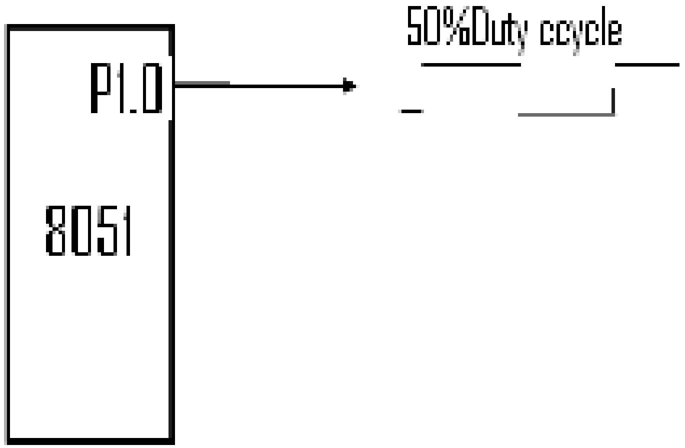
- Arithmetic
  - ADD, SUBB, INC, DEC, MUL AB, DIV AB
- Branch
  - ACALL addr16; absolute subroutine call with a 16bit address
  - SJMP rel; short jump from -128 to +127, relative present location
  - JC rel; Jump if Carry is set, short jum
  - JB bit, rel; Jump if direct bit is set; short jump
  - CJNE A,direct,rel; Compares direct byte to the accumulator and jumps if not equal.
- Data Transfer
  - MOV direct, direct;
  - MOVX A, @Ri ; get content from external RAM pointed to by Ri
  - XCH A,Rn; Exchanges the register with the accumulator
  - XCH A,@Ri; Exchanges the indirect RAM with the accumulator

- Logic
  - ANL A,Rn; AND register to accumulator
  - ANL A,direct; AND direct byte to accumulator
  - ANL A,#data; AND immediate data to accumulator
  - ORL, XRL
  - CLR A; Clears the accumulator
  - CPL A; Complements the accumulator (1=0, 0=1)
  - RL A Rotates bits in the accumulator left
  - RR A
  - RLC, RRC

- Bit-oriented
  - CLR C; Clears the carry flag
  - CLR bit; Clears the direct bit
  - SETB C, SETB bit etc.
  - CPL C; Complements the carry flag
  - CPL bit; Complements the direct bit
  - ANL C,bit. ORL C,bit

Generating Square-Wave

```
HERE : SETB    P1.0      (Make bit of Port 0 High)
       LCALL DELAY
       CLR P1.0
       LCALL DELAY
       SJMP  HERE   : Keep doing
```



Here same delay is used for both

High & low. DELAY subroutine is assume

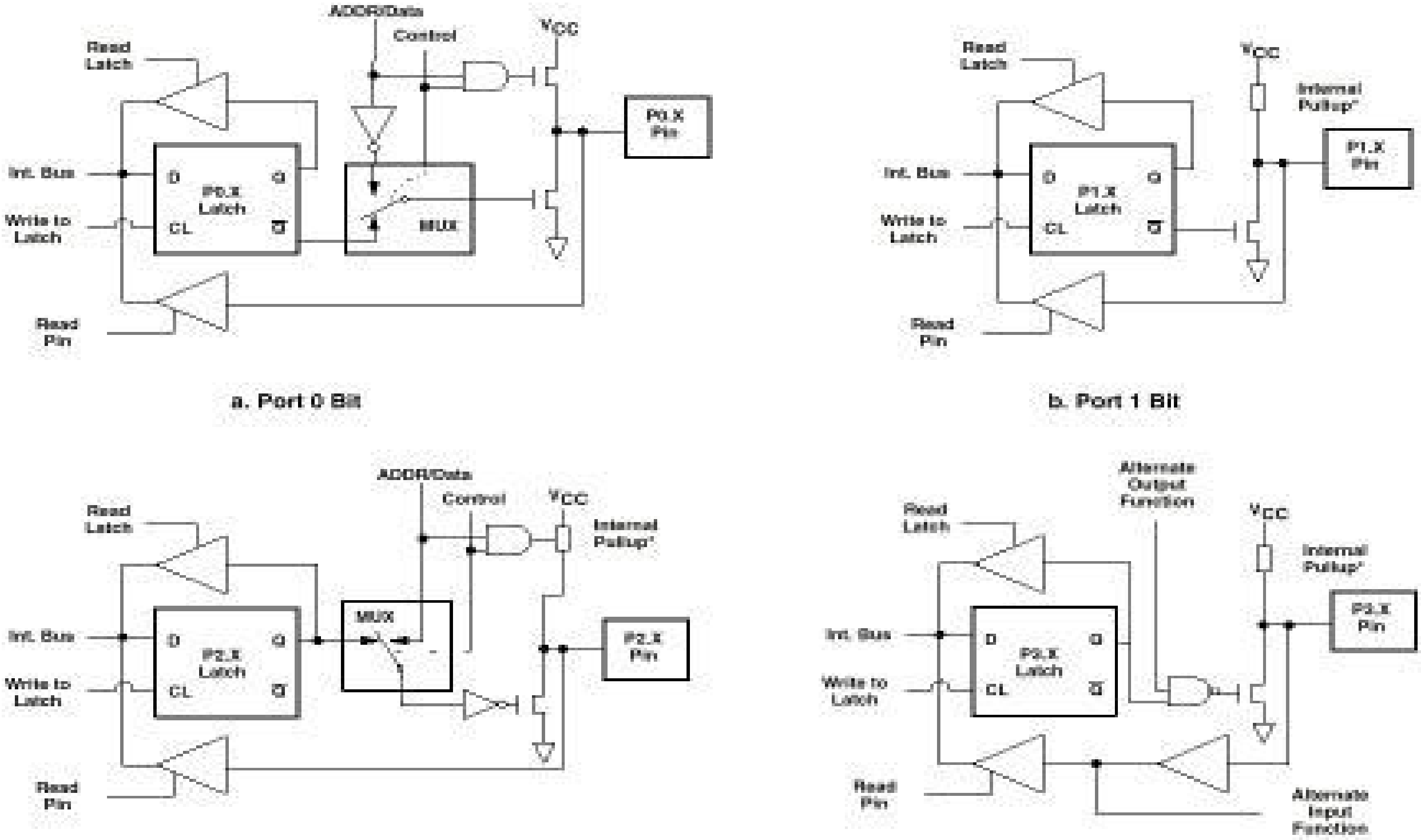
Bank Selection

- Bank 0 is default.
- RS1 and RS0 in PSW are used for Bank selected. PSW.4 ->RS1, PSW.3->RS0
- SETB PSW.4 and SETB PSW.3 commands are used.
- If PSW.4=0 and PSW.3=1 Bank 1 is selected.

Ports

- Port 0 - external memory access
  - low address byte/data
- Port 2 - external memory access
  - high address byte
- Port 1 - general purpose I/O
  - pins 0, 1 for timer/counter 2
- Port 3 - Special features
  - 0 - RxD: serial input
  - 1 - TxD: serial output
  - 2 - INT0: external interrupt

Ports



a. Port 0 Bit

b. Port 1 Bit

c. Port 2 Bit

d. Port 3 Bit

Ports

- Port 0 - true bi-directional
- Port 1-3 - have internal pullups that will source current
- Output pins:
  - Just write 0/1 to the bit/byte
- Input pins:
  - Output latch must have a 1 (reset state)
    - Turns off the pulldown
    - pullup must be pulled down by external driver
  - Just read the bit/byte

Program Status Word

- Register set select
- Status bits

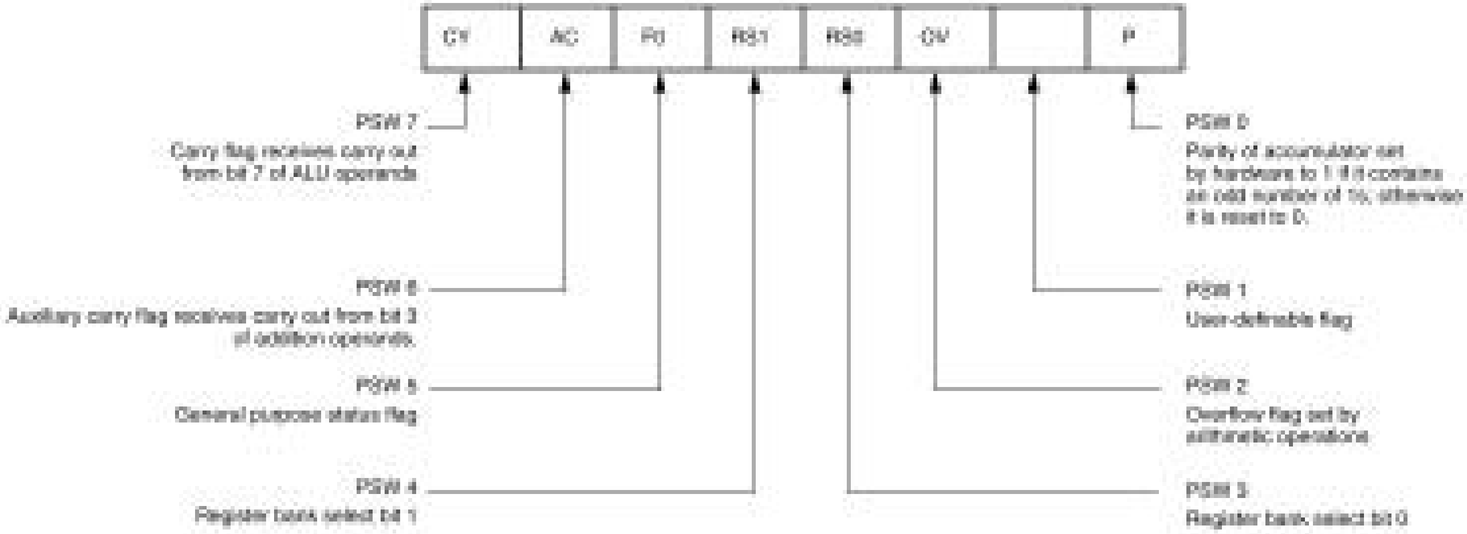
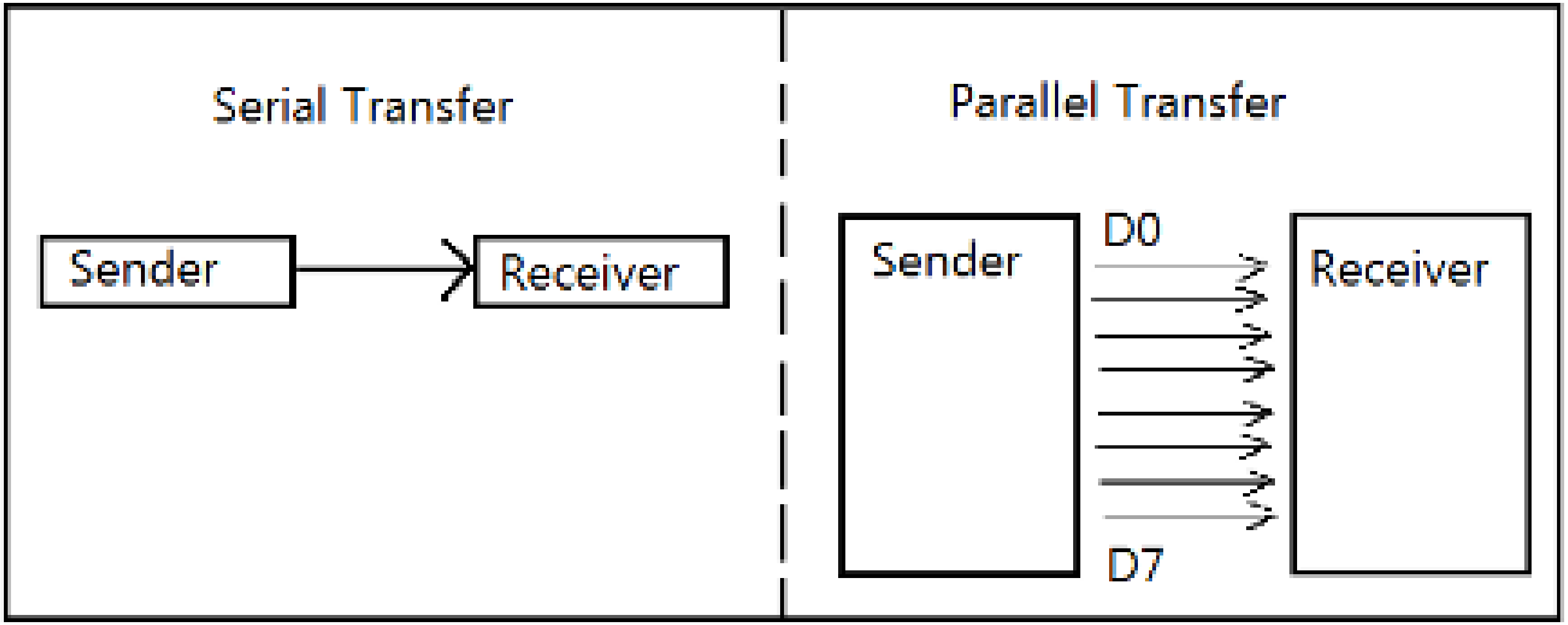

CY
AC
F0
RS1
RS0
OV
P
PSW 7
Carry flag receives carry out from bit 7 of ALU operands
PSW 6
Auxiliary carry flag receives carry out from bit 3 of addition operands.
PSW 5
General purpose status flag
PSW 4
Register bank select bit 1
PSW 0
Parity of accumulator set by hardware to 1 if it contains an odd number of 1s; otherwise it is reset to 0.
PSW 1
User-definable flag
PSW 2
Overflow flag set by arithmetic operations
PSW 3
Register bank select bit 0

Figure 10. PSW (Program Status Word) Register in 80C51 Devices

Basics of serial communication

Basics of serial communication



Serial versus Parallel Data Transfer

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.

UARTs and Transceivers

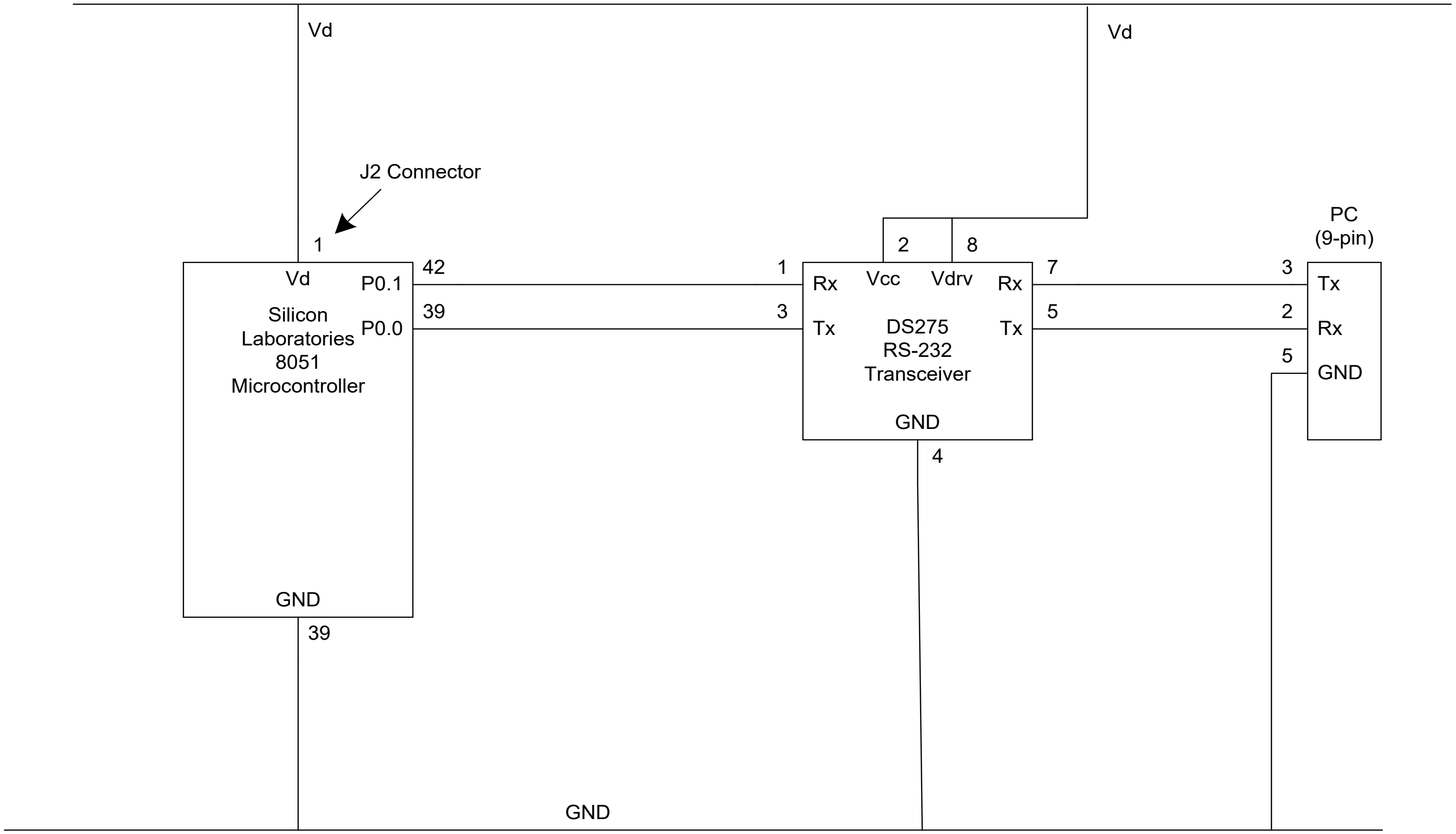
- UART (pronounced “You Art”) is an industry acronym that stands for Universal Asynchronous Receiver Transmitter. It is the interface circuitry between the microprocessor and the serial port. This circuitry is built in to the 8051 microcontroller.

- 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.

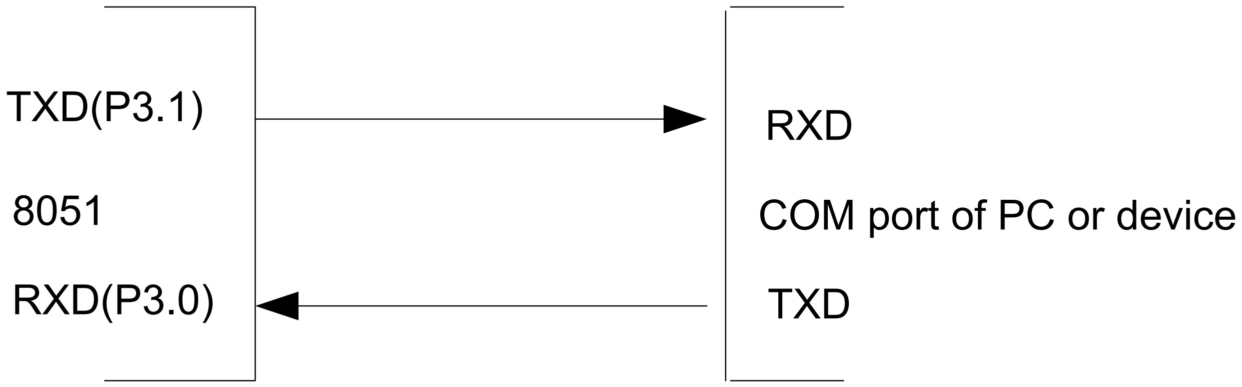
Configuring the Serial Port

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

4 UART operational modes

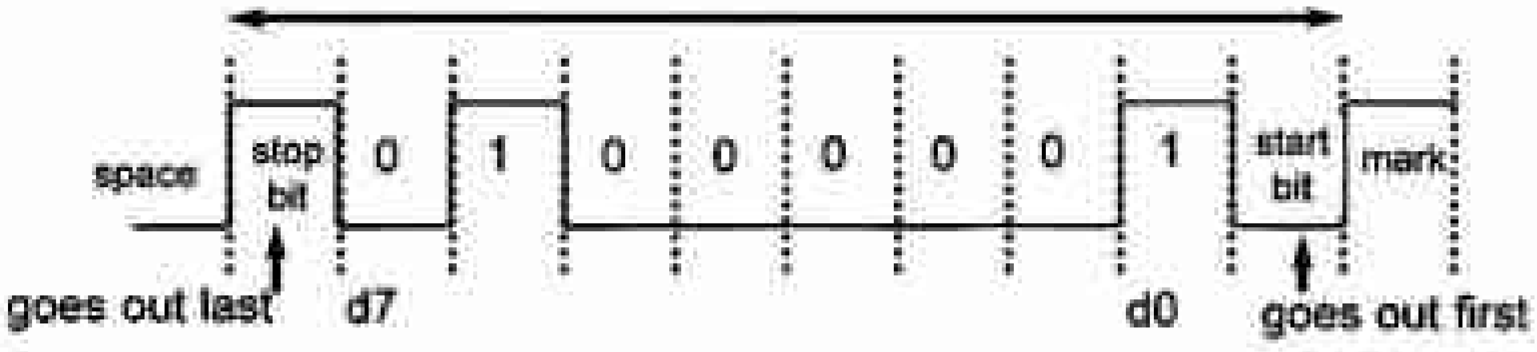
	SM0	SM1	Serial Mode	Baud Rate	Device
0	0	0	0 (Sync.) half duplex,	Oscillator/12 (fixed)	8-bit shift register
1	0	1	1(Async) full duplex	Set by Timer 1	8-bit UART
2	1	0	2(Sync) half duplex	Oscillator/64 (fixed)	9-bit UART
3	1	1	3(Async) full duplex	Set by Timer 1	9-bit UART

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

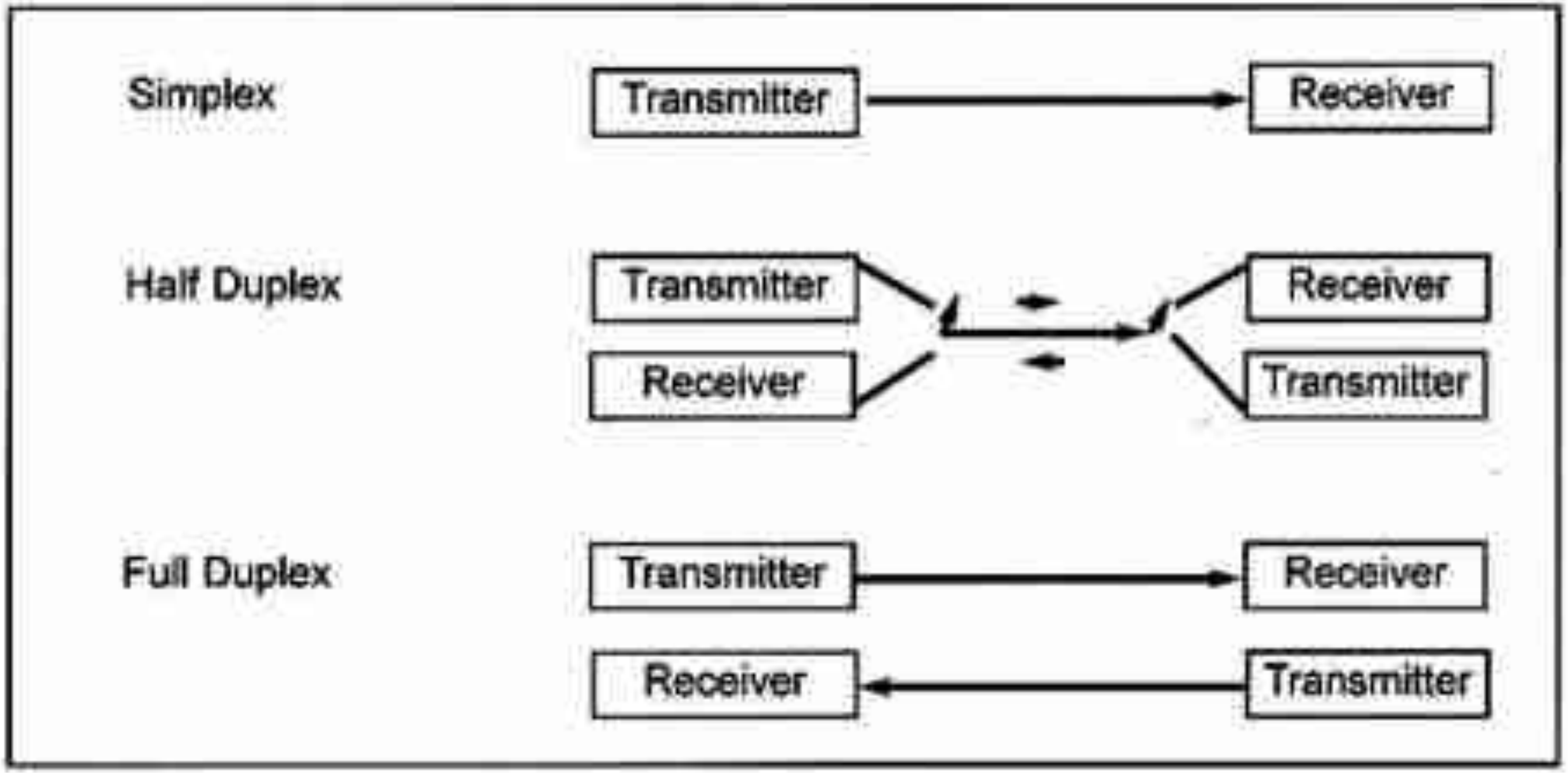


- Another job of the UART is to frame the byte of data that is serialized and transmitted. There is always one start bit (set to 0) and one stop bit (set to 1). Looking at it another way, for every byte of data, 10 bits are transmitted.

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.

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

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

Baud rate

Where:

- SMOD is the normal/double baud rate bit.
- Frequency Oscillator is the clock rate in hertz.
- Instruction 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.

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.

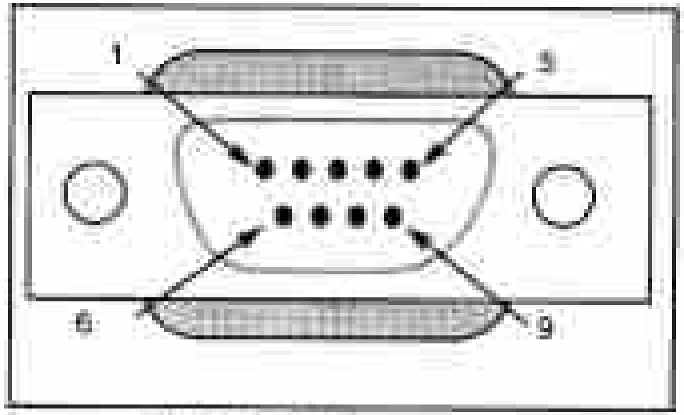
- The SBUF register (both transmit and receive) can only hold one byte. How do you know when the byte that you wrote to the port has been transmitted? Conversely, how do you know when a byte is available?
- There are ways to handle this using time delays and polling. If your application is simple enough, 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).

Handshaking

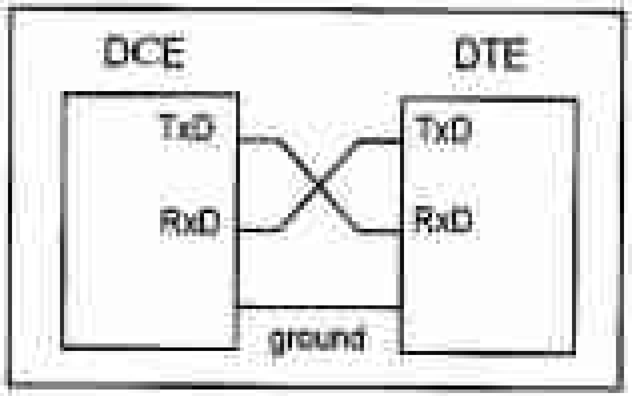
- The 8051 only has a one-byte buffer – SBUF. In contrast, a typical PC serial port with a UART with 16-byte buffer.
- If SBUF is not serviced “quickly” enough, an incoming byte may overwrite a byte that has not yet been read and processed. Using a control technique called handshaking, it is possible 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.

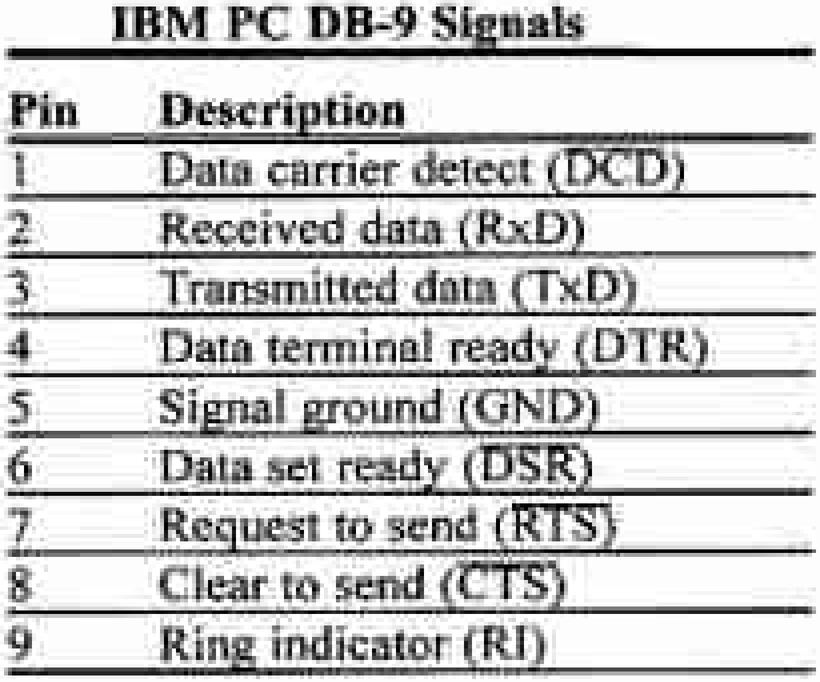
Data communication classification



DB-9 9-Pin Connector



Null Modem Connection



IBM PC DB-9 Signals

Pin	Description
1	Data carrier detect (DCD)
2	Received data (RxD)
3	Transmitted data (TxD)
4	Data terminal ready (DTR)
5	Signal ground (GND)
6	Data set ready (DSR)
7	Request to send (RTS)
8	Clear to send (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)

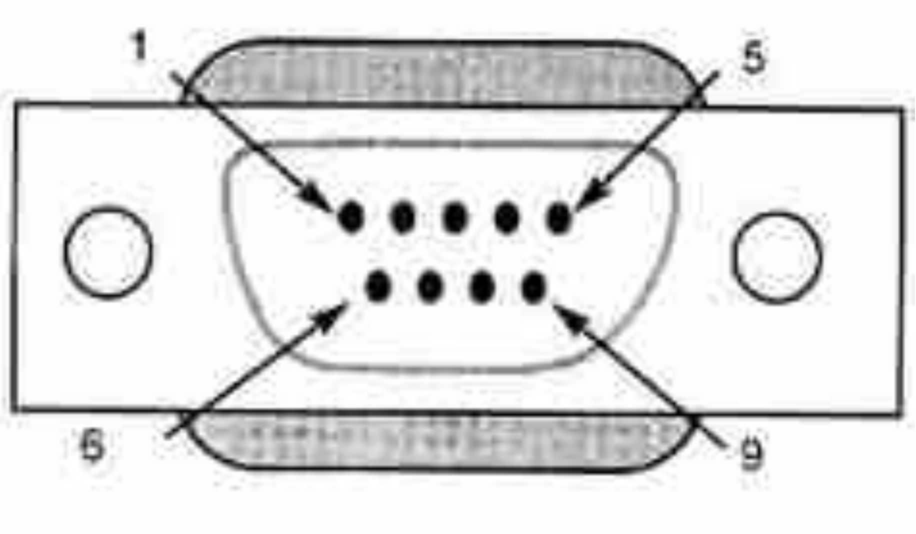
- RS232 is a point-to-point protocol meant to connect two devices together – terminals and modems. E.g., the PC is the DTE while the modem is the DCE.
- But what about other types of devices like barcode scanners and weigh scales that connect to a PC. With respect to the PC, they are all DCE devices.
- 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

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, programming using serial communcation transmitter and receiver 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.

LCD AND KEYBOARD INTERFACING

LCD Operation

- LCD is finding widespread use replacing LEDs
  - The declining prices of LCD
  - The ability to display numbers, characters, and graphics
  - Incorporation of a refreshing controller into the LCD
    - Relieving the CPU of the task of refreshing the LCD
  - Ease of programming for characters and graphics

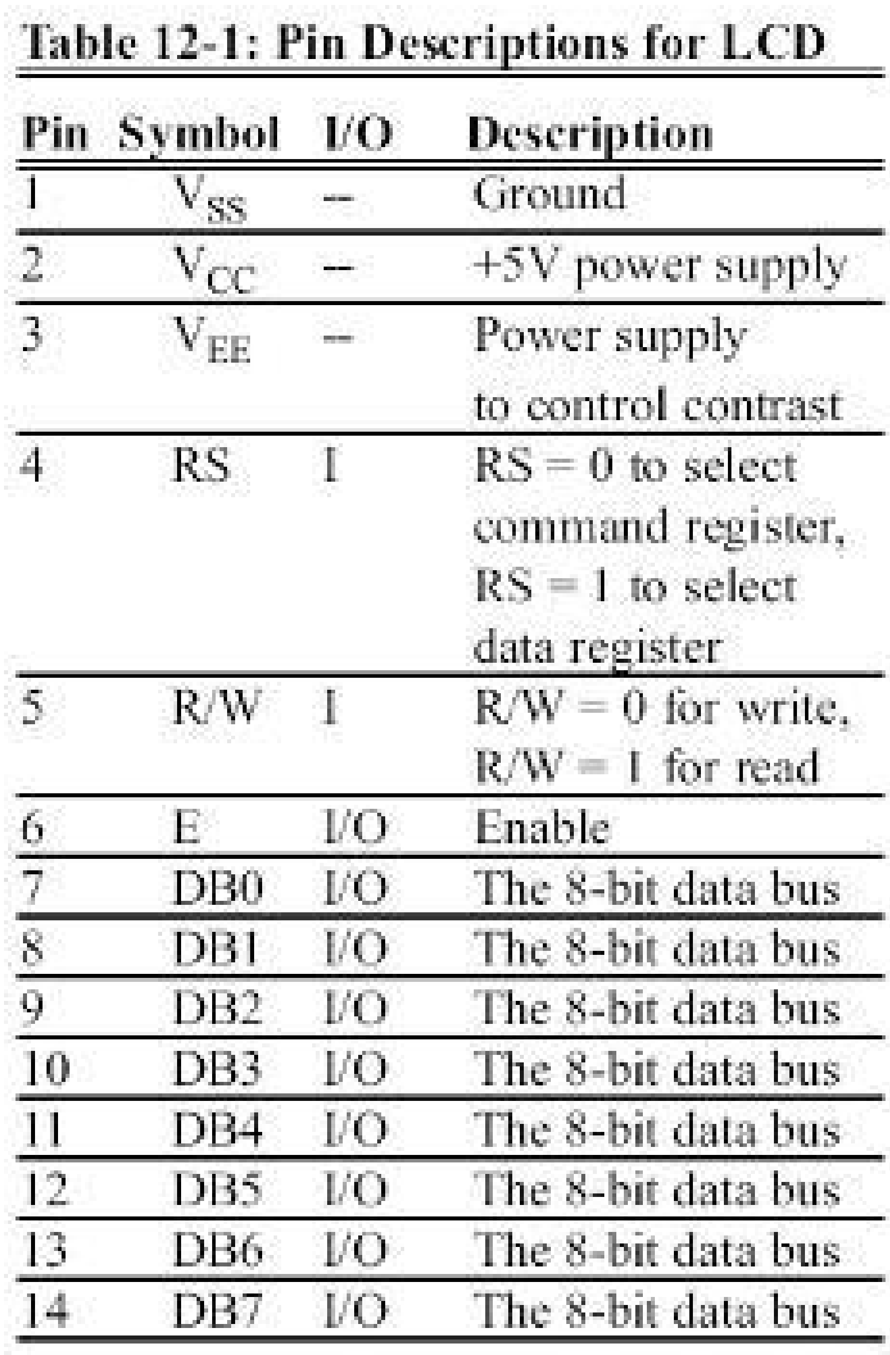


Table 12-1: Pin Descriptions for LCD

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

used by the LCD to latch information presented to its data bus

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

To send any of the commands to the LCD, make pin RS=0. For data, make RS=1. Then send a high-to-low pulse to the E pin to enable the internal latch of the LCD. This is shown in the code below.

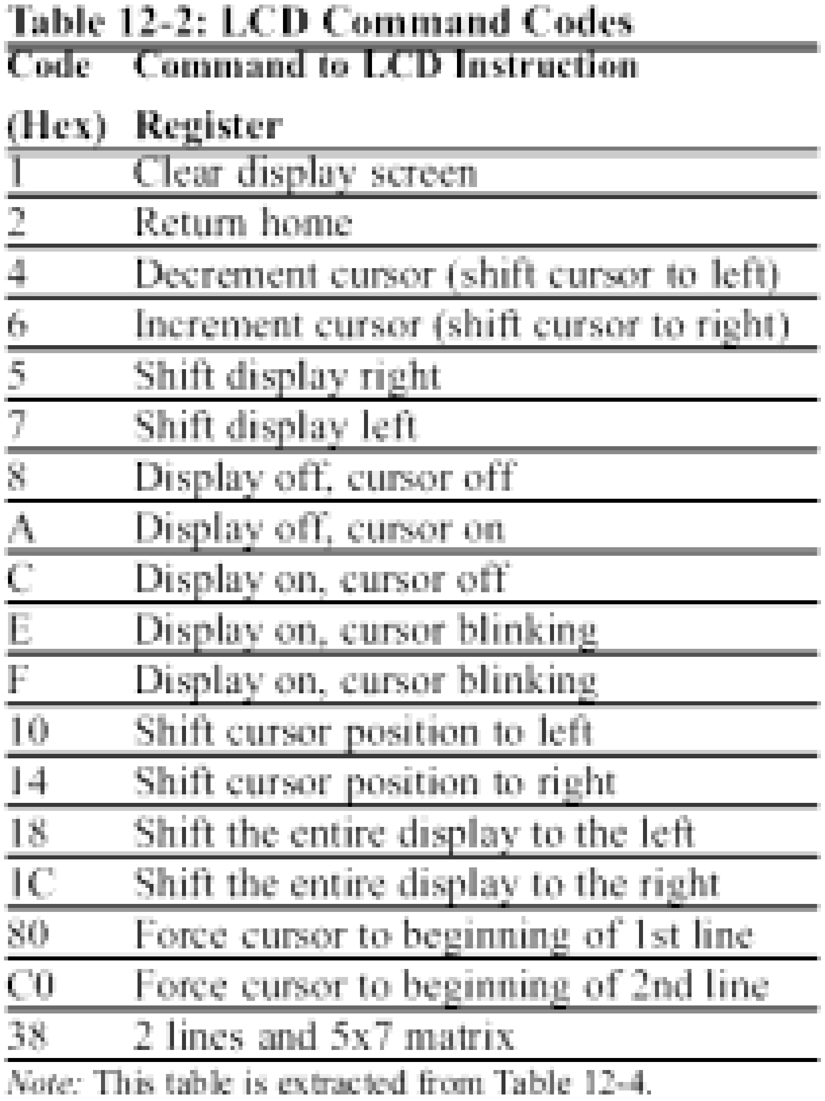
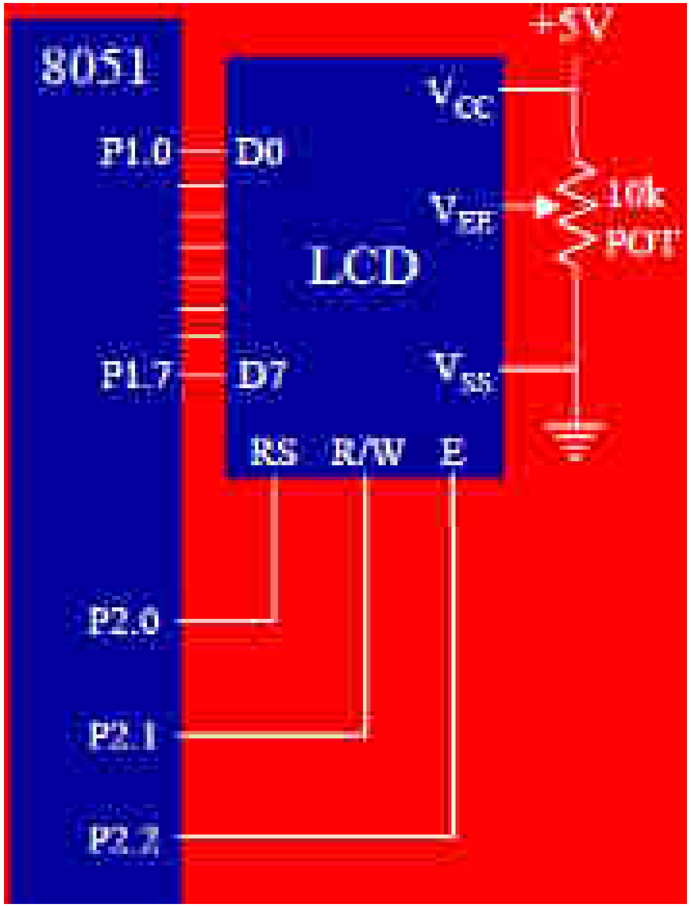
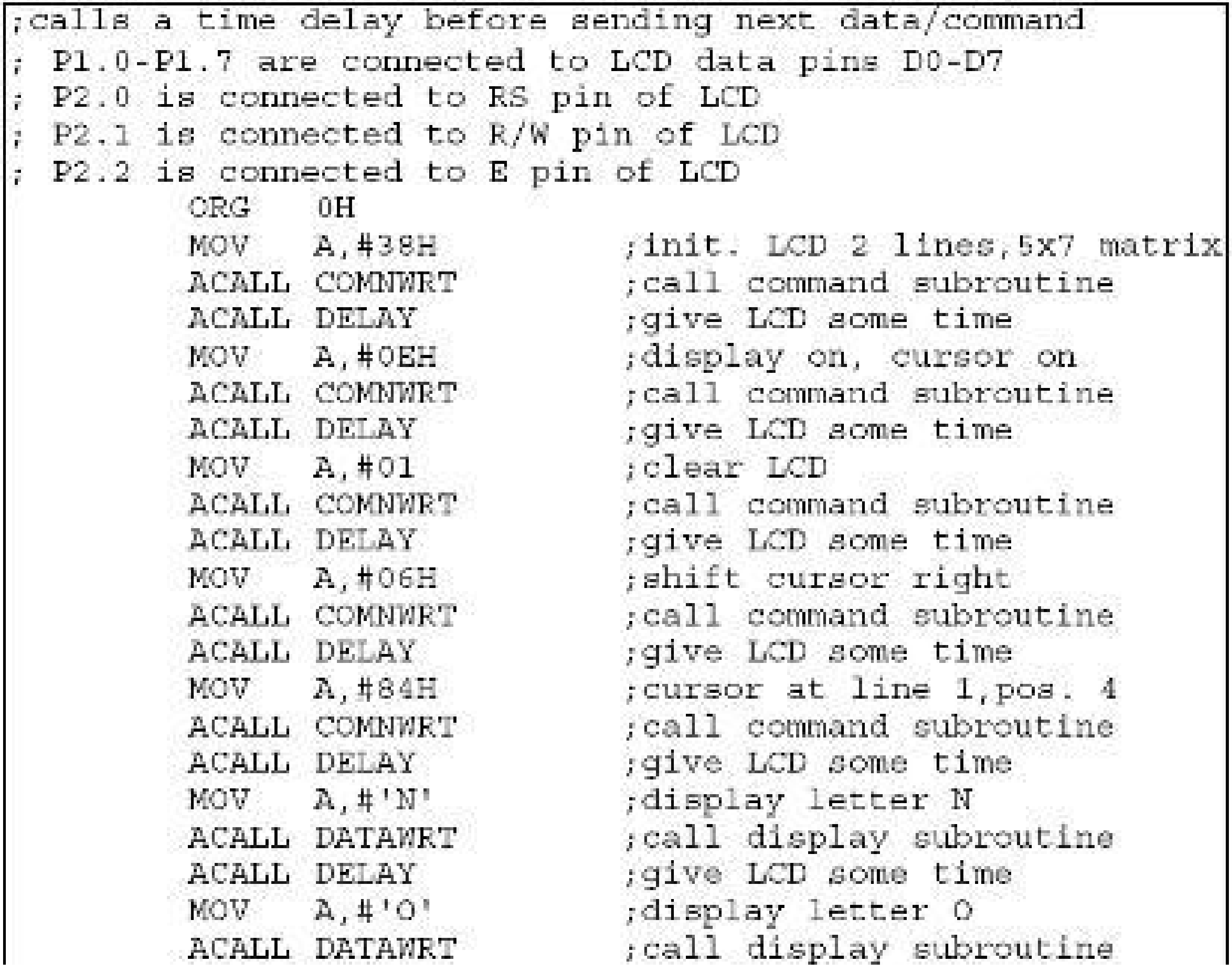


Table 12-2: LCD Command Codes

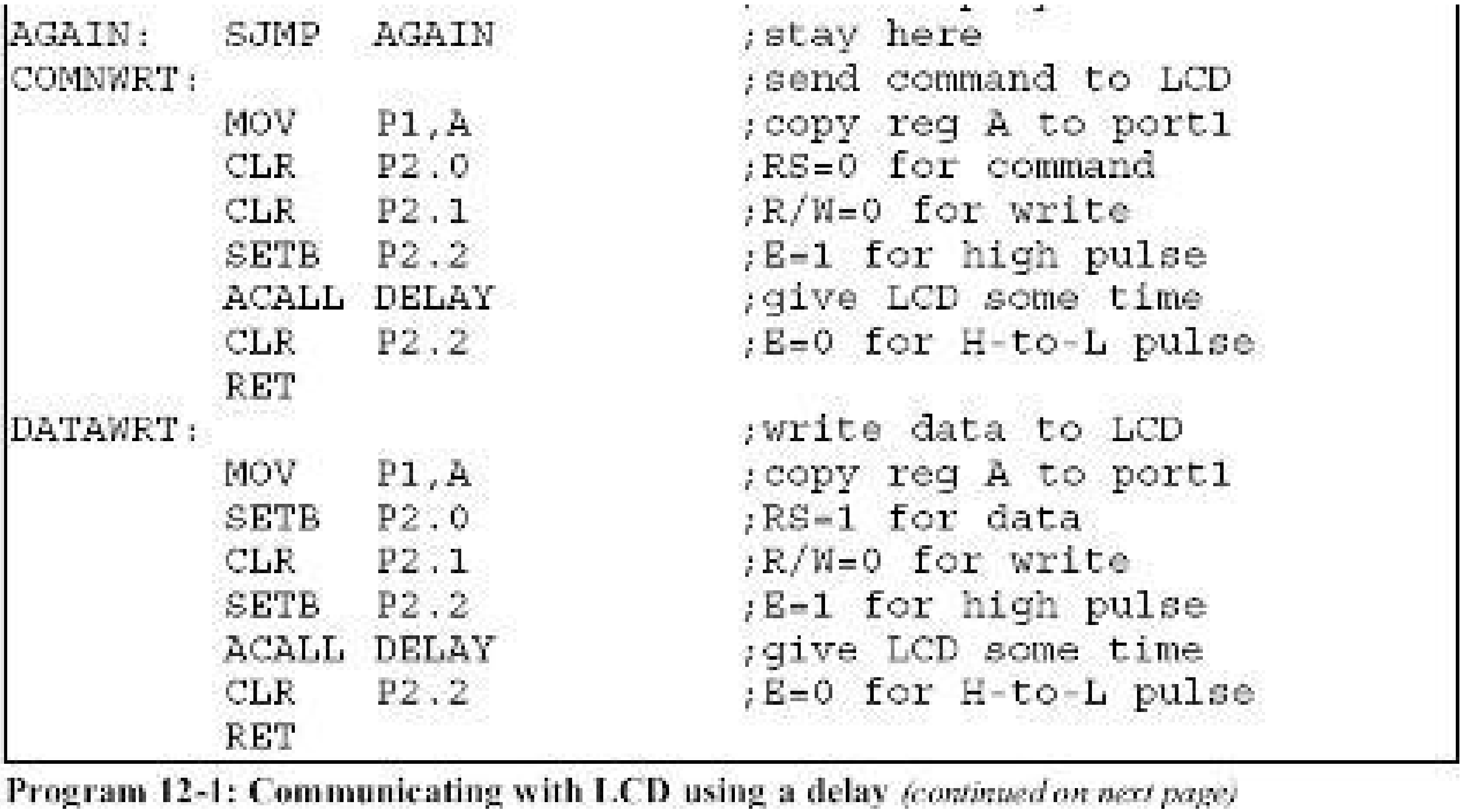
Code (Hex)	Command to LCD Instruction Register
1	Clear display screen
2	Return home
4	Decrement cursor (shift cursor to left)
6	Increment cursor (shift cursor to right)
5	Shift display right
7	Shift display left
8	Display off, cursor off
A	Display off, cursor on
C	Display on, cursor off
E	Display on,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

Note: This table is extracted from Table 12-4.



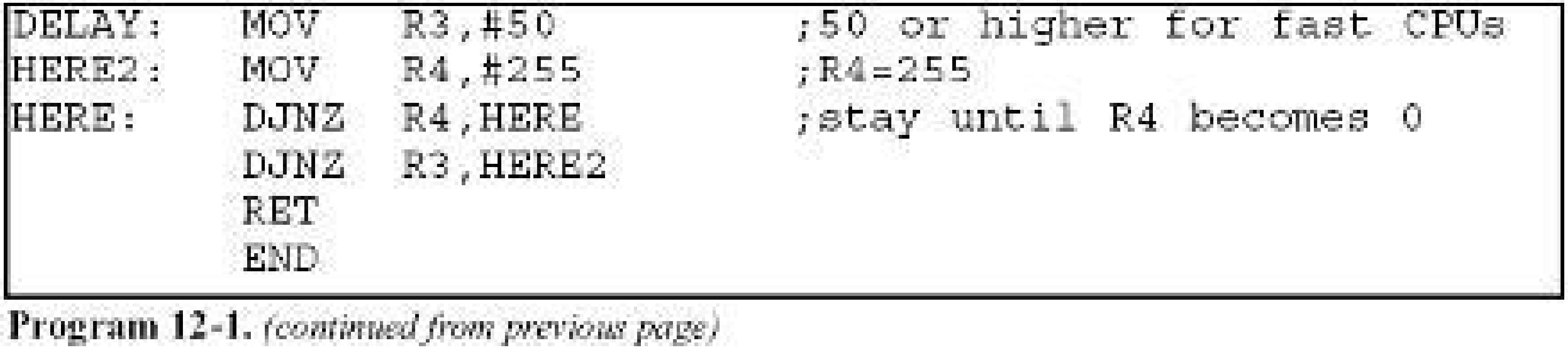


```
;calls a time delay before sending next data/command
; P1.0-P1.7 are connected to LCD data pins D0-D7
; P2.0 is connected to RS pin of LCD
; P2.1 is connected to R/W pin of LCD
; P2.2 is connected to E pin of LCD
         ORG    0H
         MOV    A,#38H          ;init. LCD 2 lines,5x7 matrix
         ACALL  COMNWRT         ;call command subroutine
         ACALL  DELAY           ;give LCD some time
         MOV    A,#0EH          ;display on, cursor on
         ACALL  COMNWRT         ;call command subroutine
         ACALL  DELAY           ;give LCD some time
         MOV    A,#01           ;clear LCD
         ACALL  COMNWRT         ;call command subroutine
         ACALL  DELAY           ;give LCD some time
         MOV    A,#06H          ;shift cursor right
         ACALL  COMNWRT         ;call command subroutine
         ACALL  DELAY           ;give LCD some time
         MOV    A,#84H          ;cursor at line 1,pos. 4
         ACALL  COMNWRT         ;call command subroutine
         ACALL  DELAY           ;give LCD some time
         MOV    A,#'N'          ;display letter N
         ACALL  DATAWRT         ;call display subroutine
         ACALL  DELAY           ;give LCD some time
         MOV    A,#'O'          ;display letter O
         ACALL  DATAWRT         ;call display subroutine
```



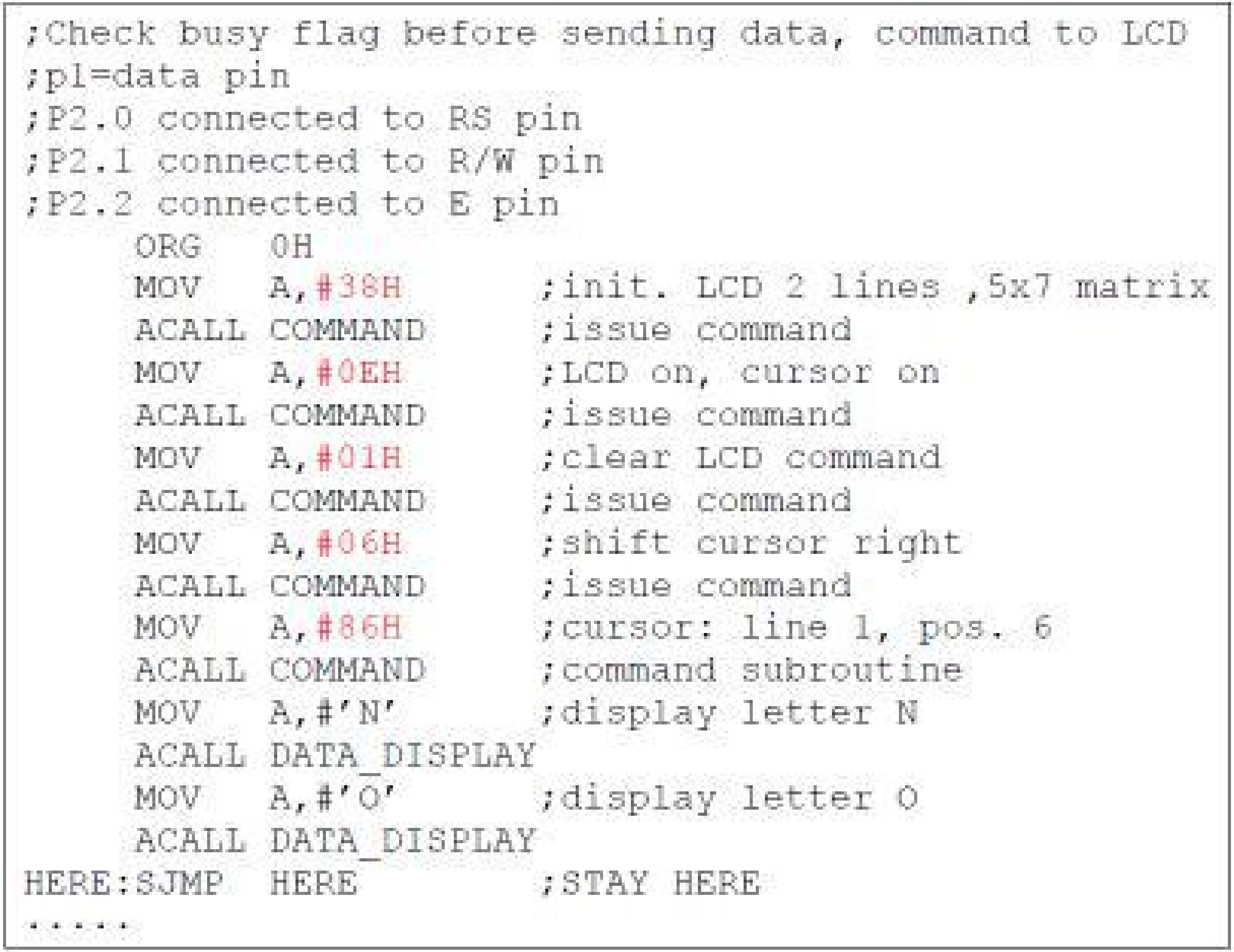
```
AGAIN:    SJMP   AGAIN               ;stay here
COMNWRT:                             ;send command to LCD
          MOV    P1,A                ;copy reg A to port1
          CLR    P2.0                ;RS=0 for command
          CLR    P2.1                ;R/W=0 for write
          SETB   P2.2                ;E=1 for high pulse
          ACALL  DELAY               ;give LCD some time
          CLR    P2.2                ;E=0 for H-to-L pulse
          RET
DATAWRT:                             ;write data to LCD
          MOV    P1,A                ;copy reg A to port1
          SETB   P2.0                ;RS=1 for data
          CLR    P2.1                ;R/W=0 for write
          SETB   P2.2                ;E=1 for high pulse
          ACALL  DELAY               ;give LCD some time
          CLR    P2.2                ;E=0 for H-to-L pulse
          RET
```

Program 12-1: Communicating with LCD using a delay *(continued on next page)*

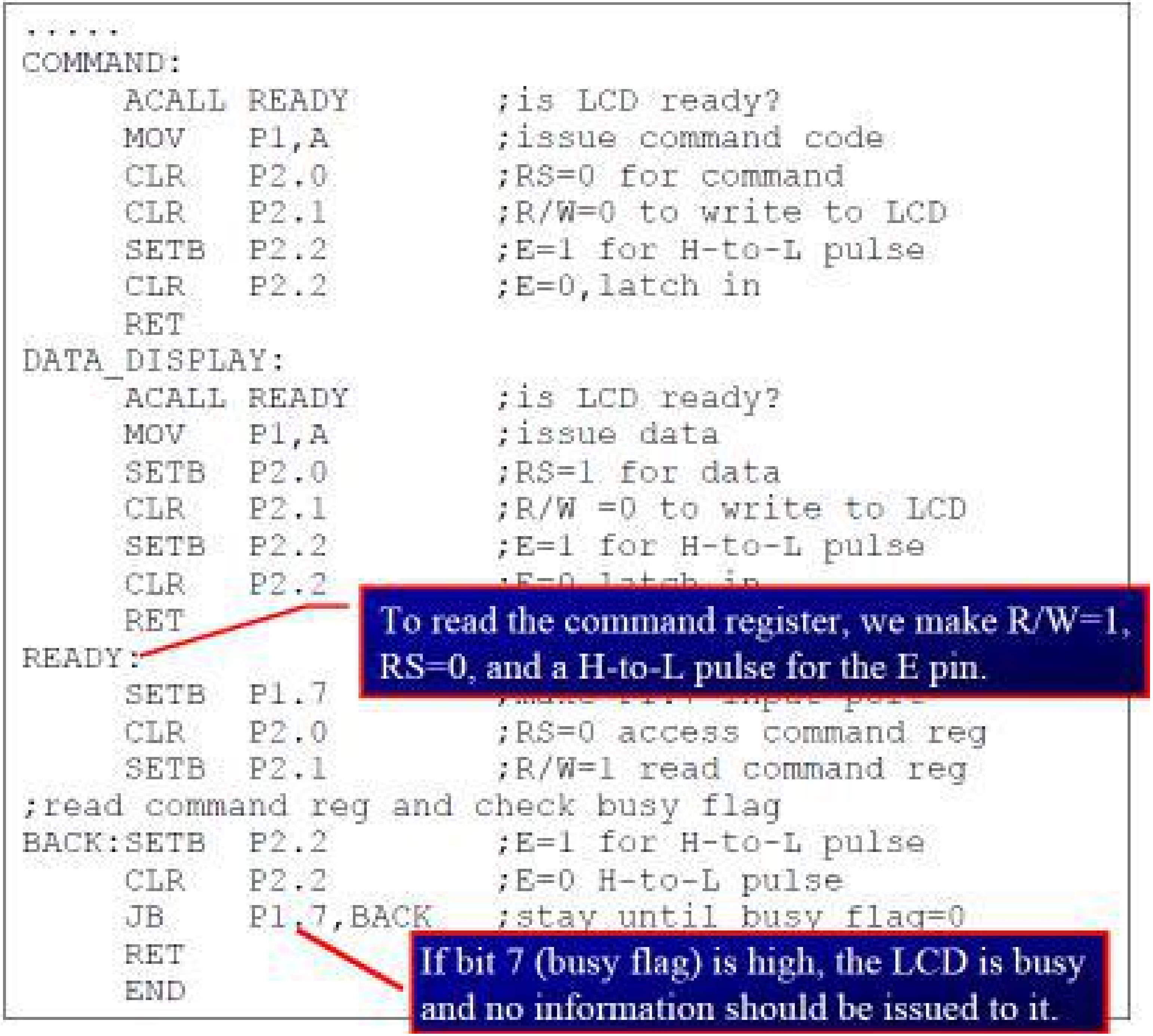


```
DELAY:    MOV    R3,#50              ;50 or higher for fast CPUs
HERE2:    MOV    R4,#255             ;R4=255
HERE:     DJNZ   R4,HERE             ;stay until R4 becomes 0
          DJNZ   R3,HERE2
          RET
          END
```

Program 12-1. *(continued from previous page)*



```
;Check busy flag before sending data, command to LCD
;p1=data pin
;P2.0 connected to RS pin
;P2.1 connected to R/W pin
;P2.2 connected to E pin
     ORG    0H
     MOV    A,#38H          ;init. LCD 2 lines ,5x7 matrix
     ACALL COMMAND          ;issue command
     MOV    A,#0EH          ;LCD on, cursor on
     ACALL COMMAND          ;issue command
     MOV    A,#01H          ;clear LCD command
     ACALL COMMAND          ;issue command
     MOV    A,#06H          ;shift cursor right
     ACALL COMMAND          ;issue command
     MOV    A,#86H          ;cursor: line 1, pos. 6
     ACALL COMMAND          ;command subroutine
     MOV    A,#'N'          ;display letter N
     ACALL DATA_DISPLAY
     MOV    A,#'O'          ;display letter O
     ACALL DATA_DISPLAY
HERE:SJMP   HERE            ;STAY HERE
.....
```

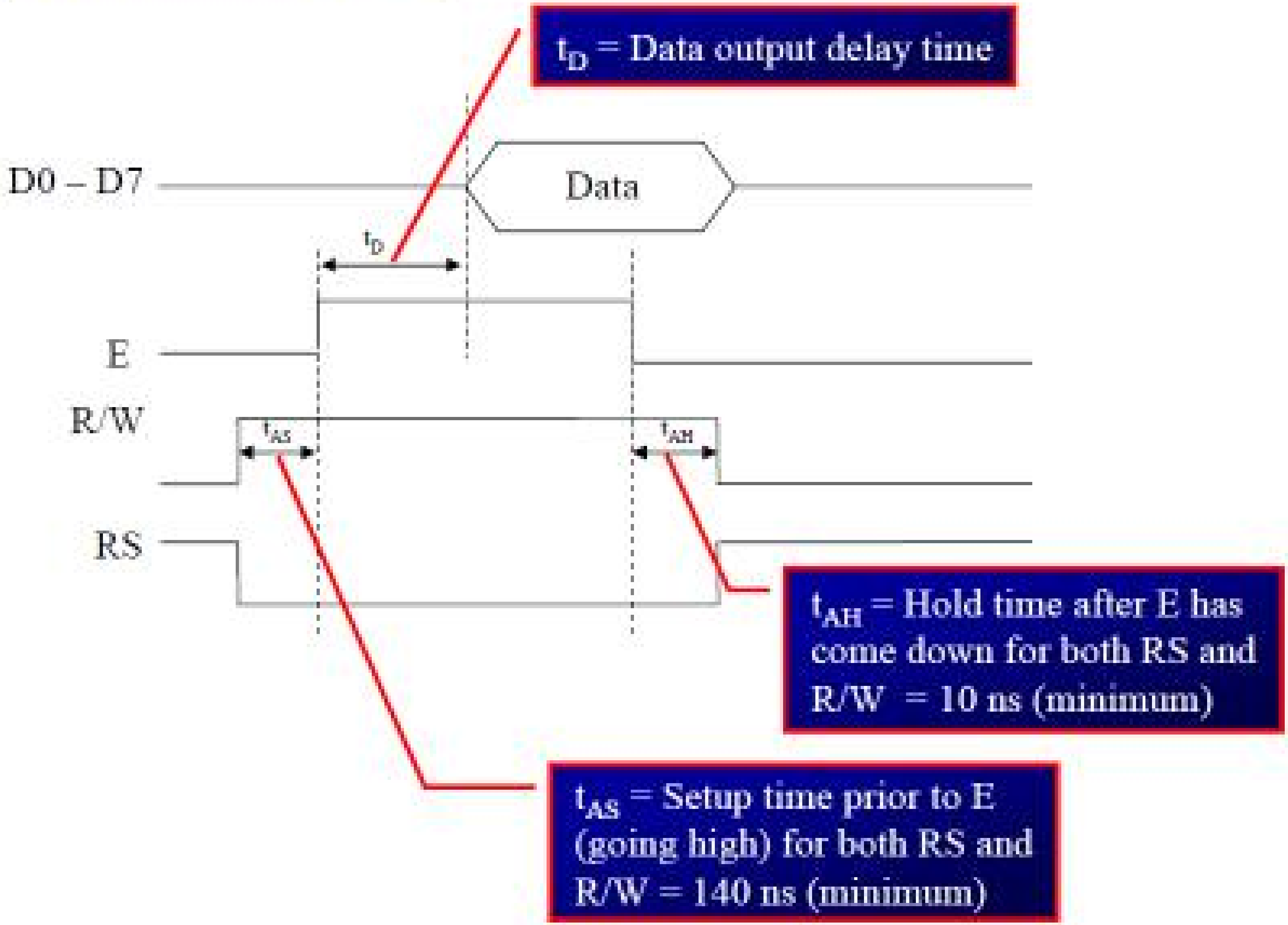


```
......
COMMAND:
     ACALL READY         ;is LCD ready?
     MOV   P1,A          ;issue command code
     CLR   P2.0          ;RS=0 for command
     CLR   P2.1          ;R/W=0 to write to LCD
     SETB  P2.2          ;E=1 for H-to-L pulse
     CLR   P2.2          ;E=0,latch in
     RET
DATA_DISPLAY:
     ACALL READY         ;is LCD ready?
     MOV   P1,A          ;issue data
     SETB  P2.0          ;RS=1 for data
     CLR   P2.1          ;R/W =0 to write to LCD
     SETB  P2.2          ;E=1 for H-to-L pulse
     CLR   P2.2          ;E=0,latch in
     RET
READY:
     SETB  P1.7          ;make P1.7 input port
     CLR   P2.0          ;RS=0 access command reg
     SETB  P2.1          ;R/W=1 read command reg
;read command reg and check busy flag
BACK:SETB  P2.2          ;E=1 for H-to-L pulse
     CLR   P2.2          ;E=0 H-to-L pulse
     JB    P1.7,BACK     ;stay until busy flag=0
     RET
     END
```

To read the command register, we make R/W=1, RS=0, and a H-to-L pulse for the E pin.

If bit 7 (busy flag) is high, the LCD is busy and no information should be issued to it.

LCD Timing for Read



t_D = Data output delay time

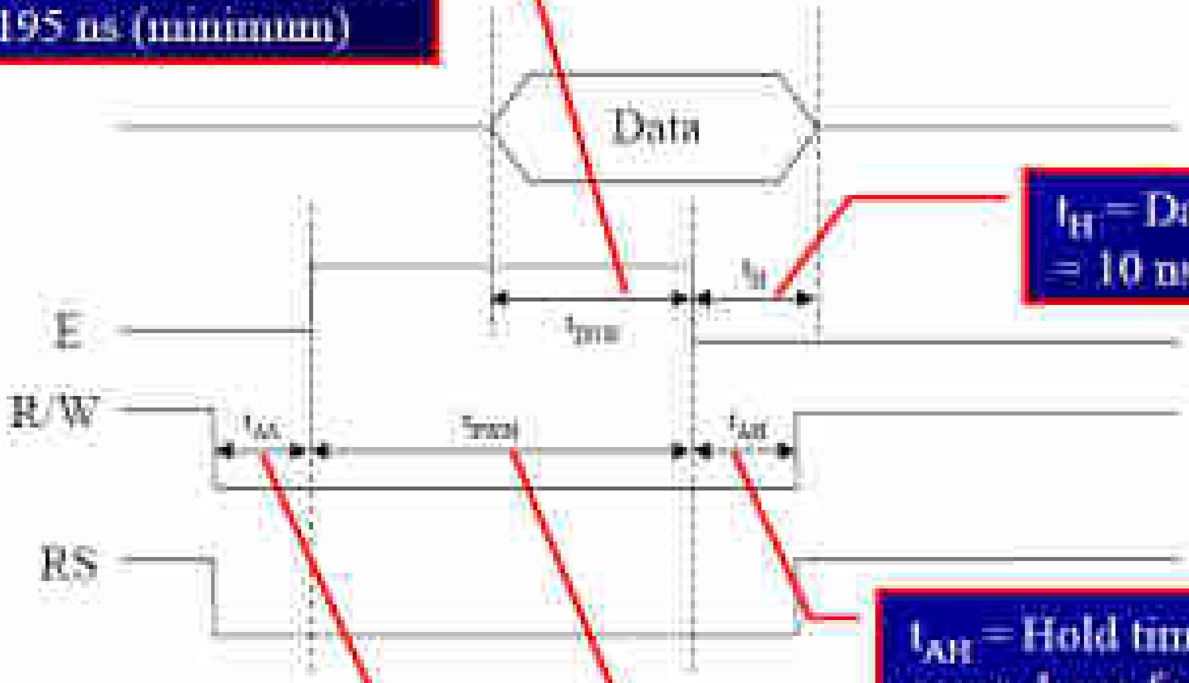
t_{AH} = Hold time after E has come down for both RS and R/W = 10 ns (minimum)

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

Note : Read requires an L-to-H pulse for the E pin

LCD Timing for Write

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



t_H = Data hold time = 10 ns (minimum)

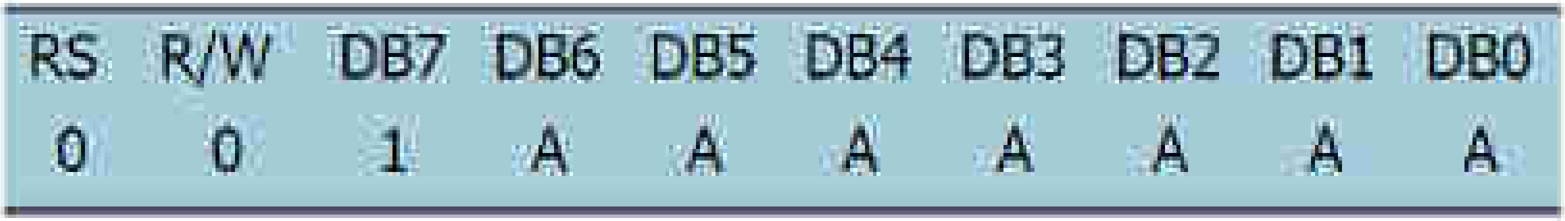
t_{AH} = Hold time after E has come down for both RS and R/W = 10 ns (minimum)

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

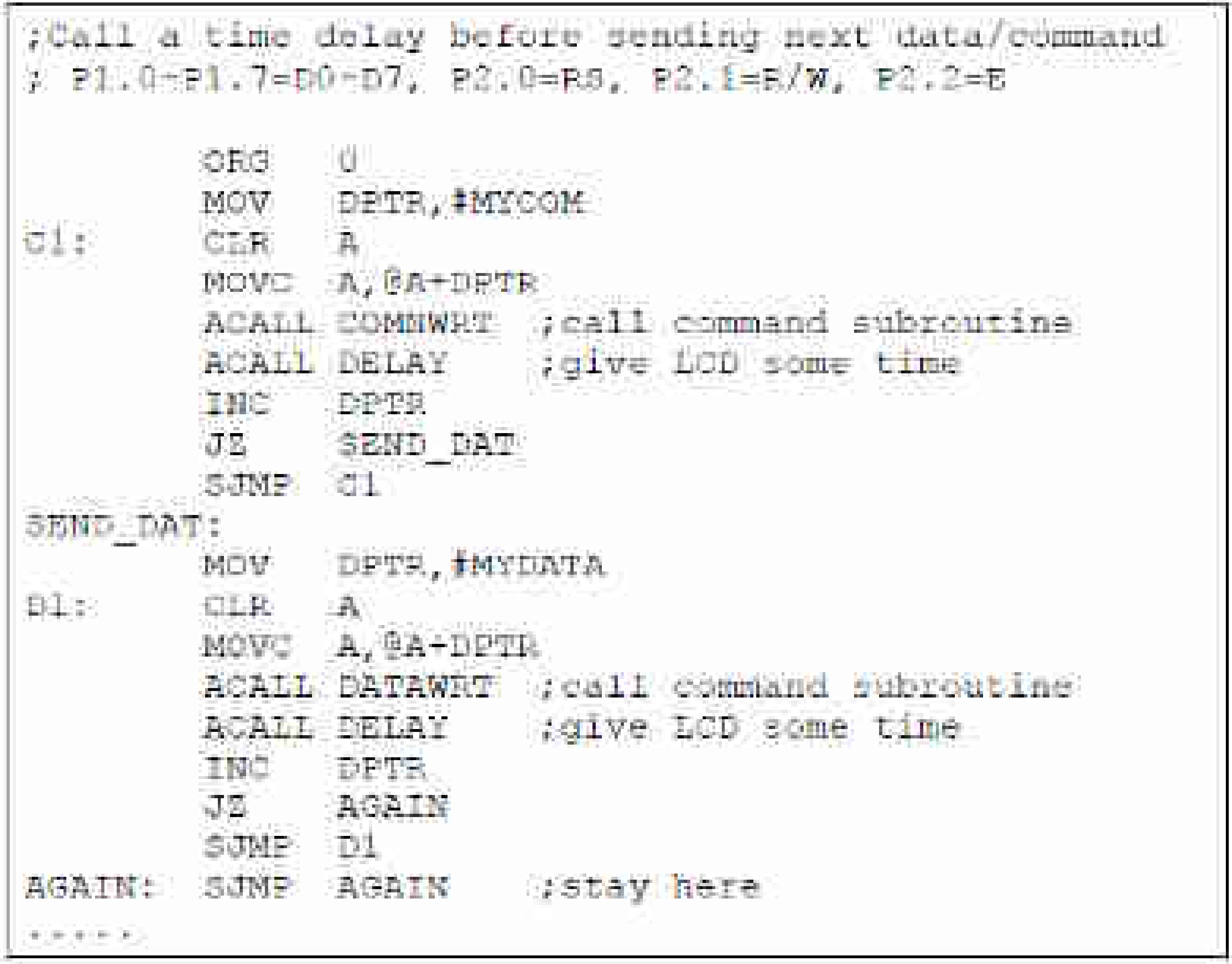
t_{AS} = Setup time prior to E (going high) for both RS and R/W = 140 ns (minimum)

LCD Data Sheet

- One can put data at any location in the LCD
  - The following shows address locations and how they are accessed
    - AAAAAAA=000_0000 to 010_0111 for line1
    - AAAAAAA=100_0000 to 110_0111 for line2
      - The upper address range can go as high as 0100111 for the 40-character-wide LCD
      - Corresponds to locations 0 to 39

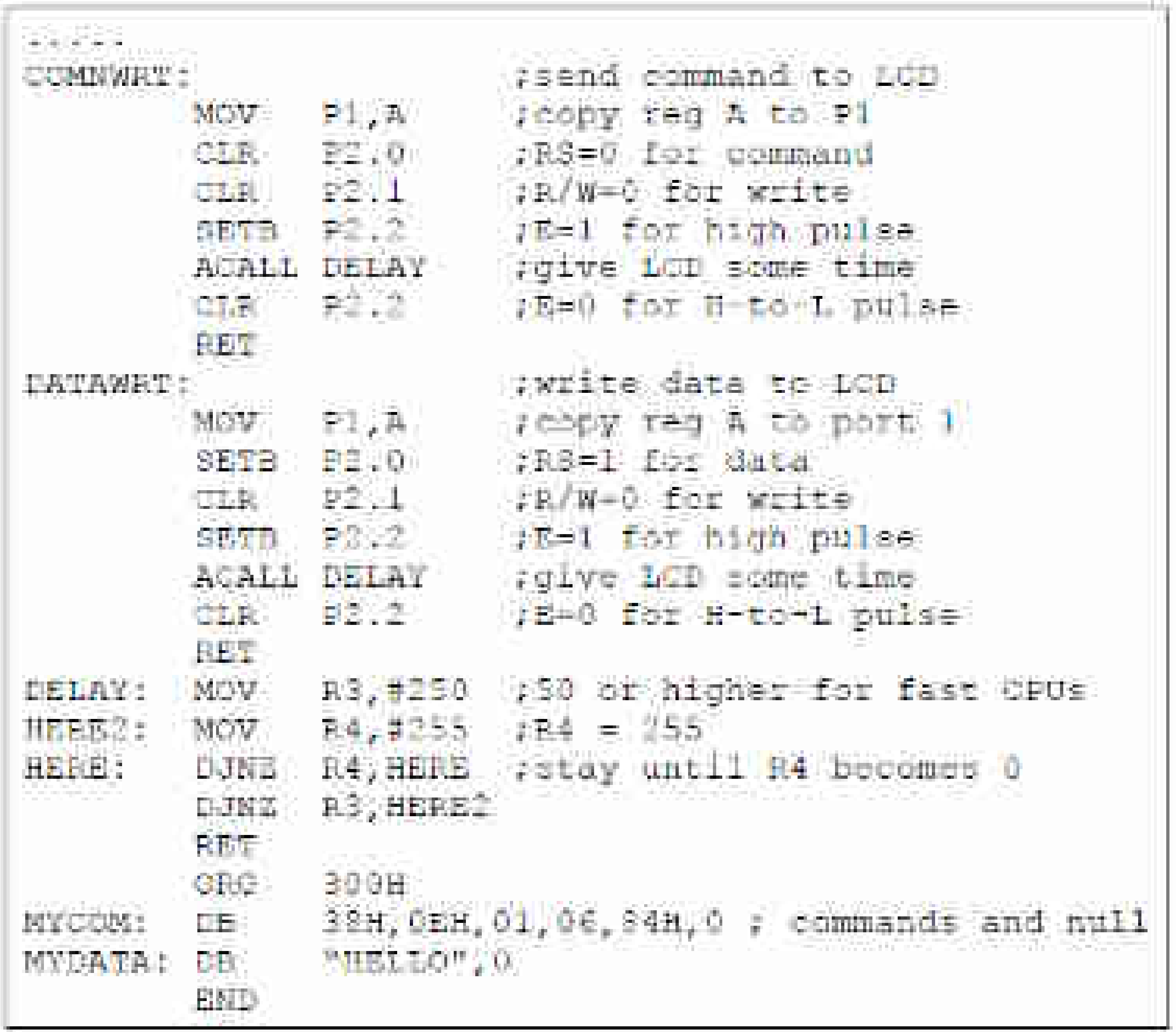


RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	1	A	A	A	A	A	A	A



```
;Call a time delay before sending next data/command
; P1.0-P1.7=D0-D7, P2.0=RS, P2.1=R/W, P2.2=E

          ORG    0
          MOV    DPTR,#MYCOM
C1:       CLR    A
          MOVC   A,@A+DPTR
          ACALL  COMNWRT   ;call command subroutine
          ACALL  DELAY     ;give LCD some time
          INC    DPTR
          JZ     SEND_DAT
          SJMP   C1
SEND_DAT:
          MOV    DPTR,#MYDATA
D1:       CLR    A
          MOVC   A,@A+DPTR
          ACALL  DATAWRT   ;call command subroutine
          ACALL  DELAY     ;give LCD some time
          INC    DPTR
          JZ     AGAIN
          SJMP   D1
AGAIN:    SJMP   AGAIN     ;stay here
.....
```



```
-----
COMNWRT:                          ;send command to LCD
         MOV    P1,A              ;copy reg A to P1
         CLR    P2.0              ;RS=0 for command
         CLR    P2.1              ;R/W=0 for write
         SETB   P2.2              ;E=1 for high pulse
         ACALL  DELAY             ;give LCD some time
         CLR    P2.2              ;E=0 for H-to-L pulse
         RET
DATAWRT:                          ;write data to LCD
         MOV    P1,A              ;copy reg A to port 1
         SETB   P2.0              ;RS=1 for data
         CLR    P2.1              ;R/W=0 for write
         SETB   P2.2              ;E=1 for high pulse
         ACALL  DELAY             ;give LCD some time
         CLR    P2.2              ;E=0 for H-to-L pulse
         RET
DELAY:   MOV    R3,#250   ;50 or higher for fast CPUs
HERE2:   MOV    R4,#255   ;R4 = 255
HERE:    DJNZ   R4,HERE   ;stay until R4 becomes 0
         DJNZ   R3,HERE2
         RET
         ORG    300H
MYCOM:   DB     38H,0EH,01,06,84H,0 ; commands and null
MYDATA:  DB     "HELLO",0
         END
```

Keyboard Interfacing

- Keyboards are organized in a matrix of rows and columns
  - The CPU accesses both rows and columns through ports
    - With two 8-bit ports, an 8 x 8 matrix of keys can be connected to a microprocessor
    - When a key is pressed, a row and a column make a contact
      - Otherwise, there is no connection between rows and columns
  - In IBM PC keyboards, a microcontroller takes care of hardware and software interfacing

Keyboard Interfacing (cont.)

- A 4x4 matrix connected to two ports
  - The rows are connected to an output port
  - The columns are connected to an input port
    - 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})
    - 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
  - It is the function of the microcontroller to scan the keyboard continuously to detect and identify the key pressed

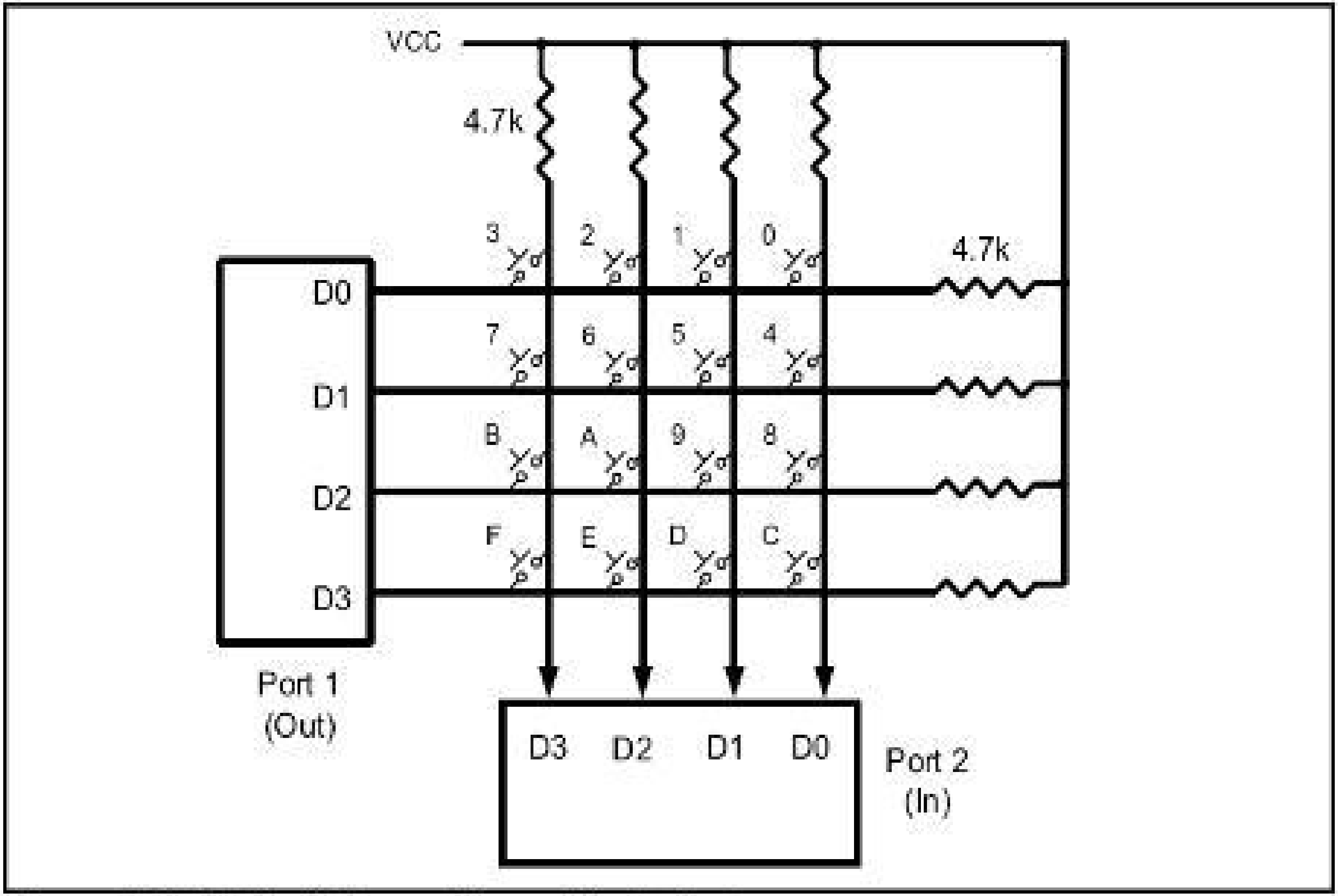


Figure 12-6. Matrix Keyboard Connection to Ports

Grounding Rows and Reading Columns

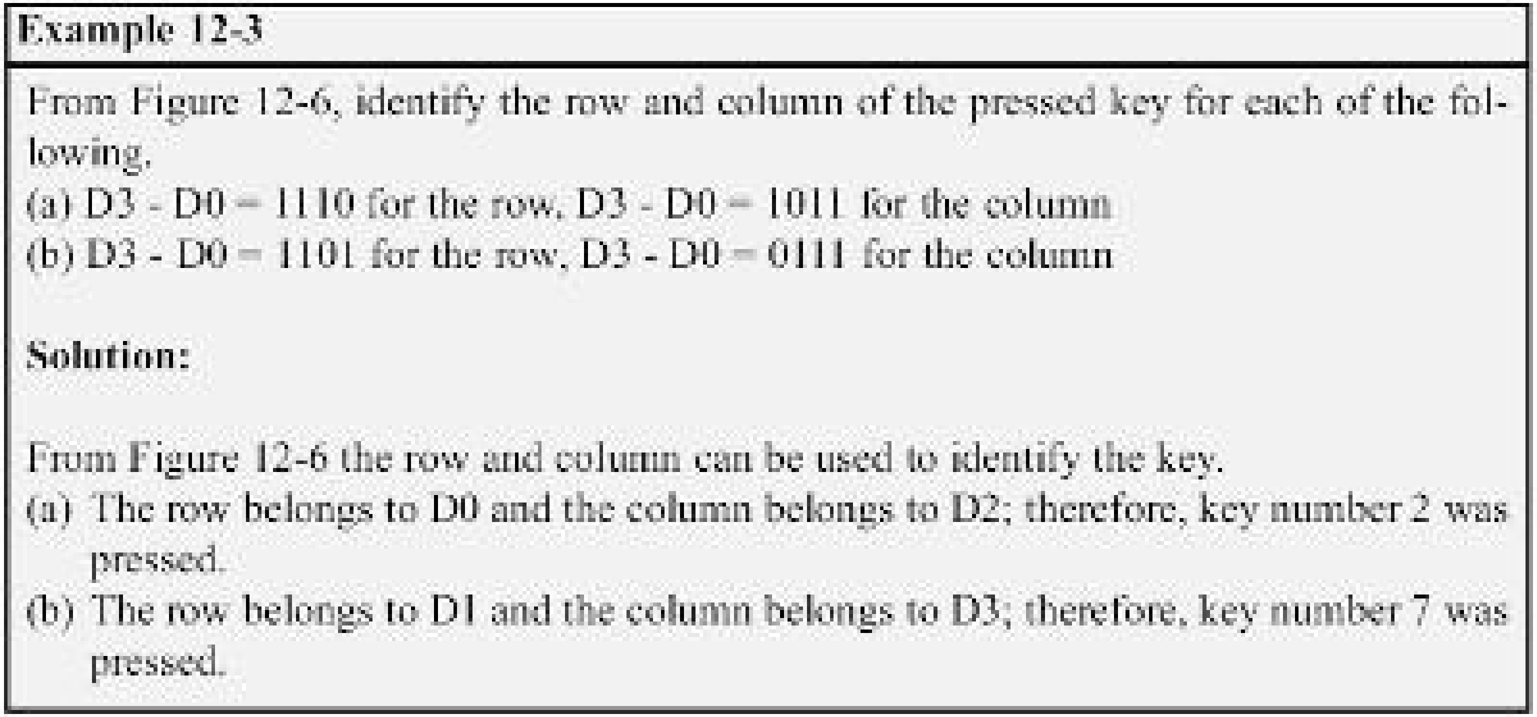
- To detect a pressed key
  - The microcontroller grounds all rows by providing 0 to the output latch
  - Then it reads the columns
    - If the data read from columns is D3 – D0 = 1111, no key has been pressed
      - The process continues till key press is detected
    - If one of the column bits has a zero, this means that a key press has occurred
    - For example, if D3 – D0 = 1101, this means that a key in the D1 column has been pressed

Grounding Rows and Reading Columns (cont.)

- After detecting a key press, the microcontroller will go through the process of identifying the key
  - Starting with the top row, the microcontroller grounds it by providing a low to row D0 only
    - It reads the columns, if the data read is all 1s, no key in that row is activated
      - The process is moved to the next row
  - It grounds the next row, reads the columns, and checks for any zero

Grounding Rows and Reading Columns (cont.)

  - This process continues until the row is identified
- After identification of the row in which the key has been pressed
  - Find out which column the pressed key belongs to



Example 12-3

From Figure 12-6, identify the row and column of the pressed key for each of the following.

(a) D3 - D0 = 1110 for the row, D3 - D0 = 1011 for the column

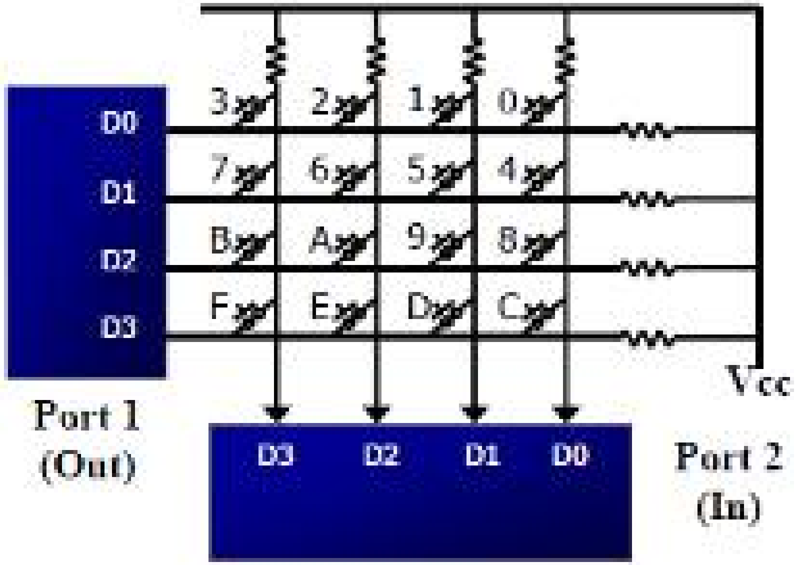
(b) D3 - D0 = 1101 for the row, D3 - D0 = 0111 for the column

Solution:

From Figure 12-6 the row and column can be used to identify the key.

(a) The row belongs to D0 and the column belongs to D2; therefore, key number 2 was pressed.

(b) The row belongs to D1 and the column belongs to D3; therefore, key number 7 was pressed.



Grounding Rows and Reading Columns (cont.)

- Detection and identification of key activation goes through the following:
  - To make sure that the preceding key has been released, 0s are output to all rows at once, and the columns are read and checked repeatedly until all the columns are high
    - When all columns are found to be high, the program waits for a short amount of time before it goes to the next stage of waiting for a key to be pressed

Grounding Rows and Reading Columns (cont.)

- To see if any key is pressed, the columns are scanned over and over in an infinite loop until one of them has a 0 on it
  - Remember that the output latches connected to rows still have their initial zeros (provided in stage 1), making them grounded
  - After the key press detection, it waits 20 ms for the bounce and then scans the columns again
    - It ensures that the first key press detection was not an erroneous one due a spike noise
    - If after the 20-ms delay the key is still pressed, it goes back into the loop to detect a real key press

Grounding Rows and Reading Columns (cont.)

- To detect which row key press belongs to, it grounds one row at a time, reading the columns each time
  - If it finds that all columns are high, this means that the key press cannot belong to that row
    - It grounds the next row and continues until it finds the row the key press belongs to
  - Upon finding the row that the key press belongs to, it sets up the starting address for the look-up table holding the scan codes (or ASCII) for that row

Grounding Rows and Reading Columns (cont.)

- To identify the key press, it rotates the column bits, one bit at a time, into the carry flag and checks to see if it is low
  - Upon finding the zero, it pulls out the ASCII code for that key from the look-up table
  - Otherwise, it increments the pointer to point to the next element of the look-up table

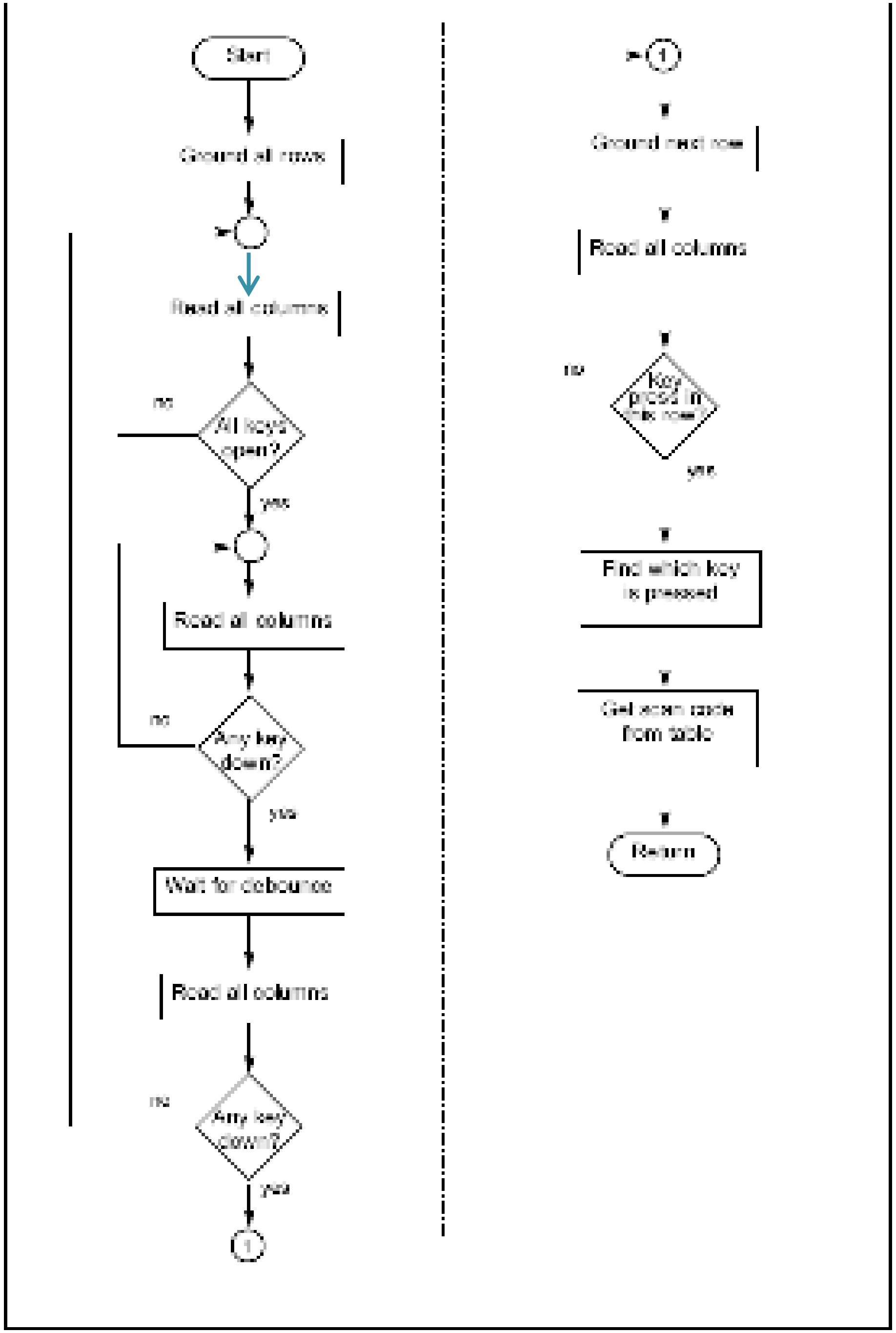
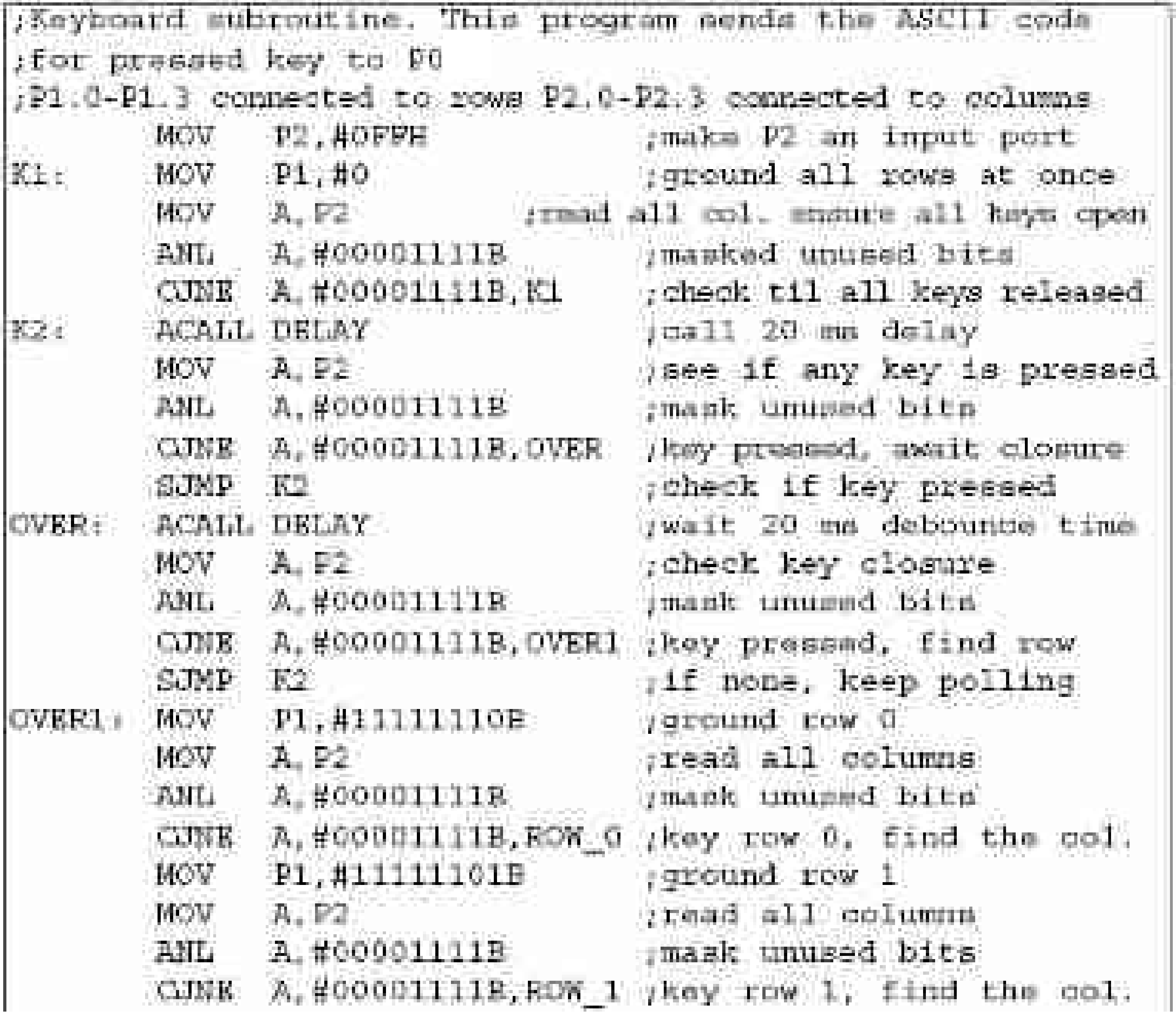
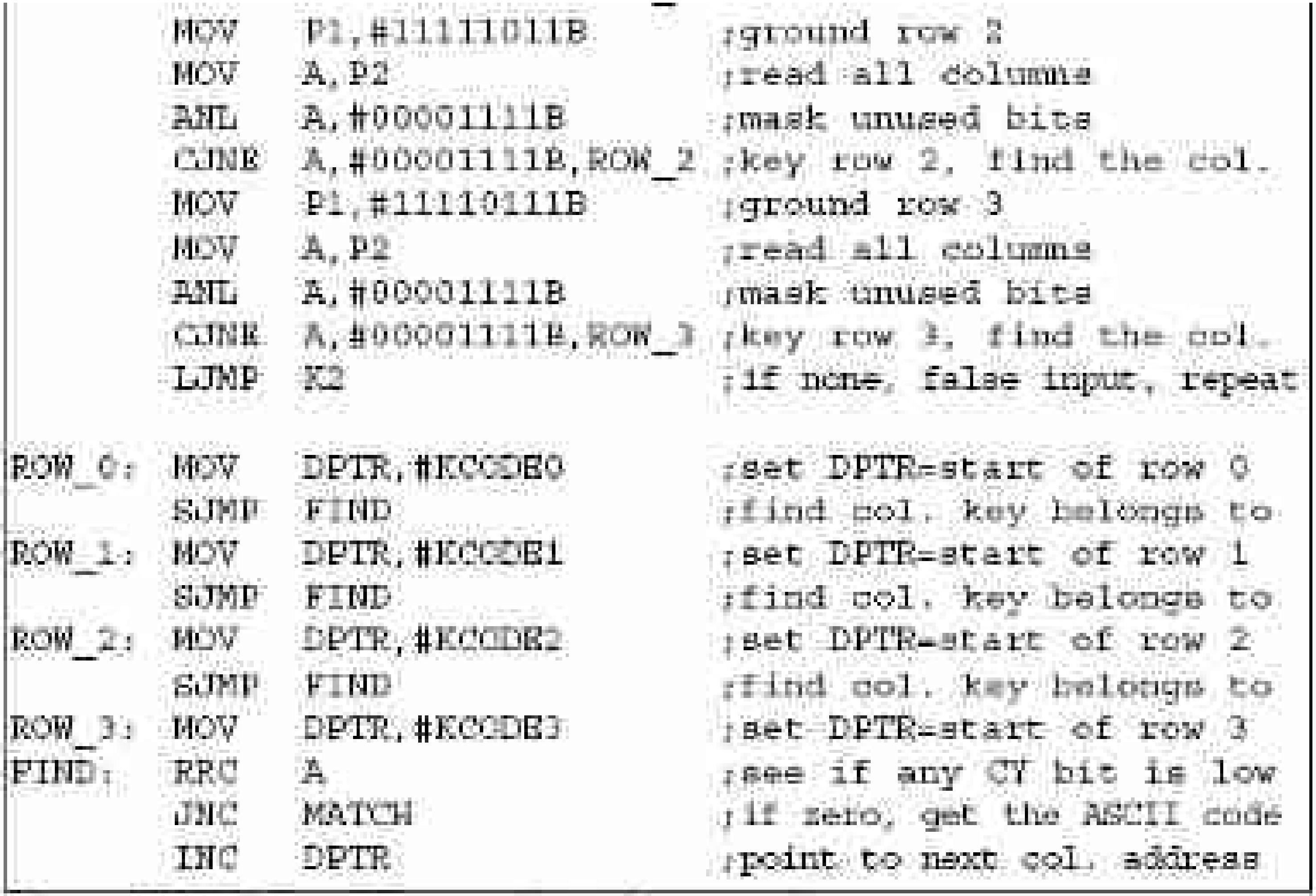


Figure 12-7. Flowchart for Program 12-4



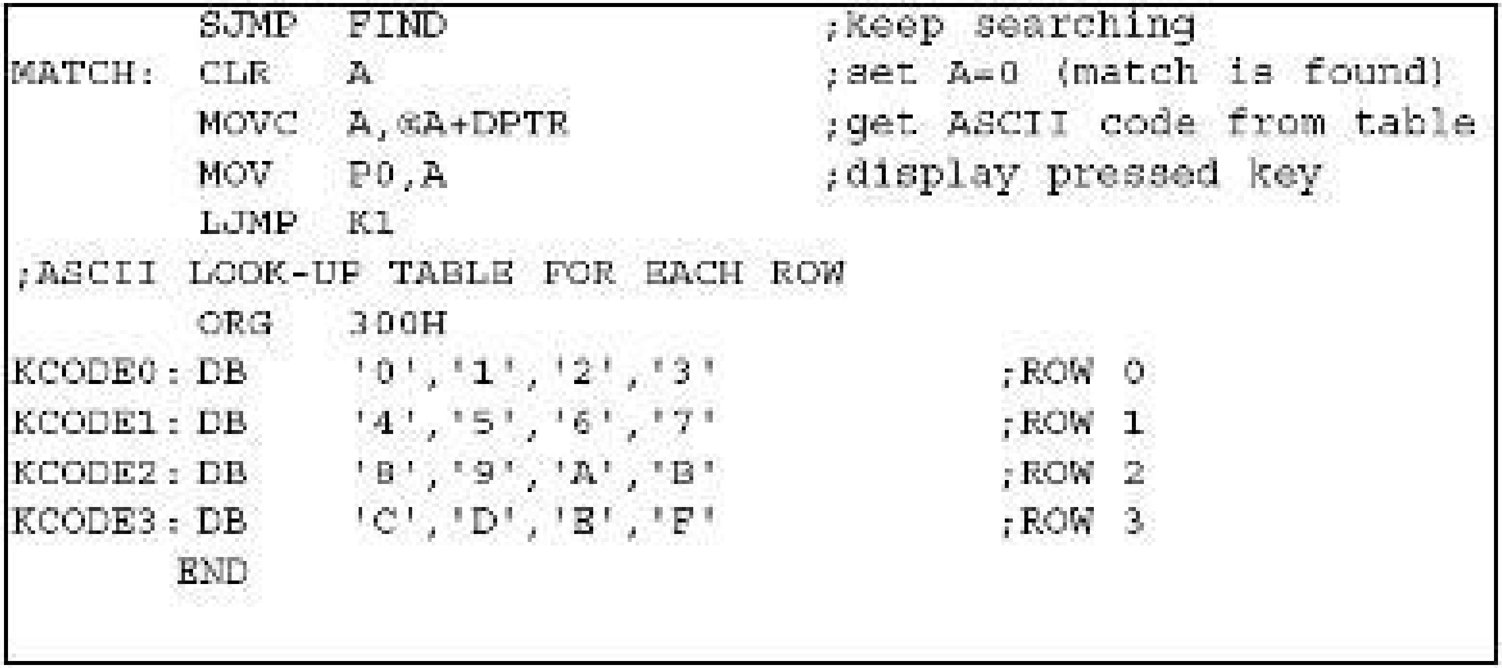
```
;Keyboard subroutine. This program sends the ASCII code
;for pressed key to P0
;P1.0-P1.3 connected to rows P2.0-P2.3 connected to columns
        MOV    P2,#0FFH                ;make P2 an input port
K1:     MOV    P1,#0                   ;ground all rows at once
        MOV    A,P2             ;read all col. ensure all keys open
        ANL    A,#00001111B            ;masked unused bits
        CJNE   A,#00001111B,K1         ;check til all keys released
K2:     ACALL  DELAY                   ;call 20 ms delay
        MOV    A,P2                    ;see if any key is pressed
        ANL    A,#00001111B            ;mask unused bits
        CJNE   A,#00001111B,OVER       ;key pressed, await closure
        SJMP   K2                      ;check if key pressed
OVER:   ACALL  DELAY                   ;wait 20 ms debounce time
        MOV    A,P2                    ;check key closure
        ANL    A,#00001111B            ;mask unused bits
        CJNE   A,#00001111B,OVER1      ;key pressed, find row
        SJMP   K2                      ;if none, keep polling
OVER1:  MOV    P1,#11111110B           ;ground row 0
        MOV    A,P2                    ;read all columns
        ANL    A,#00001111B            ;mask unused bits
        CJNE   A,#00001111B,ROW_0      ;key row 0, find the col.
        MOV    P1,#11111101B           ;ground row 1
        MOV    A,P2                    ;read all columns
        ANL    A,#00001111B            ;mask unused bits
        CJNE   A,#00001111B,ROW_1      ;key row 1, find the col.
```



```
        MOV   P1,#11111011B          ;ground row 2
        MOV   A,P2                   ;read all columns
        ANL   A,#00001111B           ;mask unused bits
        CJNE  A,#00001111B,ROW_2     ;key row 2, find the col.
        MOV   P1,#11110111B          ;ground row 3
        MOV   A,P2                   ;read all columns
        ANL   A,#00001111B           ;mask unused bits
        CJNE  A,#00001111B,ROW_3     ;key row 3, find the col.
        LJMP  K2                     ;if none, false input, repeat

ROW_0:  MOV   DPTR,#KCODE0           ;set DPTR=start of row 0
        SJMP  FIND                   ;find col. key belongs to
ROW_1:  MOV   DPTR,#KCODE1           ;set DPTR=start of row 1
        SJMP  FIND                   ;find col. key belongs to
ROW_2:  MOV   DPTR,#KCODE2           ;set DPTR=start of row 2
        SJMP  FIND                   ;find col. key belongs to
ROW_3:  MOV   DPTR,#KCODE3           ;set DPTR=start of row 3
FIND:   RRC   A                      ;see if any CY bit is low
        JNC   MATCH                  ;if zero, get the ASCII code
        INC   DPTR                   ;point to next col. address
```

Program 12-4: Keyboard Program *(continued on next page)*



```
        SJMP  FIND                    ;keep searching
MATCH:  CLR   A                       ;set A=0 (match is found)
        MOVC  A,@A+DPTR               ;get ASCII code from table
        MOV   P0,A                    ;display pressed key
        LJMP  K1
;ASCII LOOK-UP TABLE FOR EACH ROW
        ORG   300H
KCODE0: DB    '0','1','2','3'                 ;ROW 0
KCODE1: DB    '4','5','6','7'                 ;ROW 1
KCODE2: DB    '8','9','A','B'                 ;ROW 2
KCODE3: DB    'C','D','E','F'                 ;ROW 3
       END
```

Program 12-4. *(continued from previous page)*

Question Bank(Microcontroller (BELE0-F99)

1. The special function registers are maintained in the next 128 locations after the general-purpose data storage and stack.

A.True
B.False

Answer: A

2. Which data memory control and handle the operation of several peripherals by assigning them in the category of special function registers?

a. Internal on-chip RAM
b. External off-chip RAM
c. Both a & b
d. None of the above

ANSWER: (a) Internal on-chip RAM

3. Why is the speed accessibility of external data memory slower than internal on-chip RAM?

a. Due to multiplexing of lower order byte of address-data bus
b. Due to multiplexing of higher order byte of address-data bus
c. Due to demultiplexing of lower order byte of address-data bus
d. Due to demultiplexing of higher order byte of address-data bus

ANSWER: (a) Due to multiplexing of lower order byte of address-data bus

4. Which operations are performed by the bit manipulating instructions of boolean processor?

a. Complement bit
b. Set bit
c. Clear bit
d. All of the above

ANSWER: (d) All of the above

5. Which control signal/s is/are generated by timing and control unit of 8051 microcontroller in order to access the off-chip devices apart from the internal timings?

a. ALE
b. PSEN

c. RD & WR
d. All of the above

ANSWER: (d) All of the above

6. Which register usually store the output generated by ALU in several arithmetic and logical operations?

a. Accumulator
b. Special Function Register
c. Timer Register
d. Stack Pointer

ANSWER: (a) Accumulator

7) Which condition approve to prefer the EPROM/ROM versions for mass production in order to prevent the external memory connections?

a. size of code < size of on-chip program memory
b. size of code > size of on-chip program memory
c. size of code = size of on-chip program memory
d. None of the above

ANSWER: (a) size of code < size of on-chip program memory

8) Which among the below mentioned devices of MCS-51 family does not possess two 16 - bit timers/counters?

a. 8031
b. 8052
c. 8751
d. All of the above

ANSWER: (b) 8052

9) Which characteristic/s of accumulator is /are of greater significance in terms of its functionality?

a. Ability to store one of the operands before the execution of an instruction
b. Ability to store the result after the execution of an instruction
c. Both a & b
d. None of the above

ANSWER: (c) Both a & b

10) Which general purpose register holds eight bit divisor and store the remainder especially after the execution of division operation?

a. A-Register
b. B-Register
c. Registers R0 through R7
d. All of the above

ANSWER: (b) B-Register

11) How many registers can be utilized to write the programs by an effective selection of register bank in program status word (PSW)?

a. 8
b. 16
c. 32
d. 64

ANSWER: (c) 32

12) Which operations are performed by stack pointer during its incremental phase?

a. Push
b. Pop
c. Return
d. All of the above

ANSWER: (a) Push

13) Which is the only register without internal on-chip RAM address in MCS-51?

a. Stack Pointer
b. Program Counter
c. Data Pointer
d. Timer Register

ANSWER: (b) Program Counter

14) What kind of instructions usually affect the program counter?

a. Call & Jump
b. Call & Return
c. Push & Pop
d. Return & Jump

ANSWER: (a) Call & Jump

15) What is the default value of stack once after the system undergoes the reset condition?

a. 07H
b. 08H
c. 09H
d. 00H

ANSWER:(a) 07H

16) Which bit/s play/s a significant role in the selection of a bank register of Program Status Word (PSW)?

a. RS1
b. RS0
c. Both a & b
d. None of the above

ANSWER: (c) Both a & b

17) Which flags represent the least significant bit (LSB) and most significant bit (MSB) of Program Status Word (PSW) respectively?

a. Parity Flag & Carry Flag
b. Parity Flag & Auxiliary Carry Flag
c. Carry Flag & Overflow Flag
d. Carry Flag & Auxiliary Carry Flag

ANSWER: (a) Parity Flag & Carry Flag

18) Which register bank is supposed to get selected if the values of register bank select bits RS1 & Rs0 are detected to be '1' & '0' respectively?

a. Bank 0
b. Bank 1
c. Bank 2
d. Bank 3

ANSWER: (c) Bank 2

19) It is possible to set the auxiliary carry flag while performing addition or subtraction operations only when the carry exceeds _______

a. 1st bit
b. 2nd bit
c. 3rd bit
d. 4th bit

ANSWER: (c) 3rd bit

20) Which locations of 128 bytes on-chip additional RAM are generally reserved for special functions?

a. 80H to 0FFH
b. 70H to 0FFH
c. 90H to 0FFH
d. 60H to 0FFH

ANSWER: (a) 80H to 0FFH

21) Which commands are used for addressing the off-chip data and associated codes respectively by data pointer?

a. MOVX & MOVC
b. MOVY & MOVB
c. MOVZ & MOVA
d. MOVC & MOVY

ANSWER: (a) MOVX & MOVC

22) Which instruction find its utility in loading the data pointer with 16 bits immediate data?

a. MOV
b. INC
c. DEC
d. ADDC

ANSWER: (a) MOV

23) What is the maximum capability of addressing the off-chip data memory & off-chip program memory in a data pointer?

a. 8K
b. 16K
c. 32K
d. 64K

ANSWER: (d) 64K

24) Which among the below stated registers does not belong to the category of special function registers?

a. TCON & TMOD
b. TH0 & TL0
c. P0 & P1
d. SP & PC

ANSWER: (d) SP & PC

25) Which timer is attributed to the register pair of RCAP2H & RCAP2L for capture mode operation?

a. Timer 0
b. Timer 1
c. Timer 2
d. Timer 3

ANSWER:(c) Timer 2

26) Which registers are supposed to get copied into RCAP2H & RCAP2L respectively due to the transition at 8052 T2EX pin in the capture mode operation?

a. TH0 & TH1
b. TH1 & TH1
c. TH2 & TH2
d. All of the above

ANSWER: (c) TH2 & TH2

27) Which mode of timer 2 allow to hold the reload values with an assistance of RCAP2H & RCAP2L register pair?

a. 8 bit auto-reload mode
b. 16 bit auto reload mode
c. 8 bit capture mode
d. 16 bit capture mode

ANSWER: (b) 16 bit auto reload mode

28) Where should the pin 19 (XTAL1), acting as an input of inverting amplifier as well as part of an oscillator circuit, be connected under the application of external clock?

a. to XTAL2
b. to Vcc
c. to GND
d. to ALE

ANSWER: (c) to GND

29) Which port does not represent quasi-bidirectional nature of I/O ports in accordance to the pin configuration of 8051 microcontroller?

a. Port 0 (Pins 32-39)
b. Port 1 (Pins 1-8)
c. Port 2 (Pins 21-28)
d. Port 3 (Pins 10-17)

ANSWER: (a) Port 0 (Pins 32-39)

30) What is the required baud rate for an efficient operation of serial port devices in 8051 microcontroller?

a. 1200
b. 2400
c. 4800
d. 9600

ANSWER: (d) 9600

31) Which among the below mentioned functions does not belong to the category of alternate functions usually performed by Port 3 (Pins 10-17)?

a. External Interrupts
b. Internal Interrupts
c. Serial Ports
d. Read / Write Control signals

ANSWER: (b) Internal Interrupts

32) What is the constant activation rate of ALE that is optimized periodically in terms of an oscillator frequency?

a. 1 / 8
b. 1 / 6
c. 1 / 4
d. 1 / 2

ANSWER:(b) 1 / 6

33) Which output control signal is activated after every six oscillator periods while fetching the external program memory and almost remains high during internal program execution?

a. ALE
b. PSEN

c. EA
d. All of the above

ANSWER: (b) PSEN

34) Which memory allow the execution of instructions till the address limit of 0FFFH especially when the External Access (EA) pin is held high?

a. Internal Program Memory
b. External Program Memory
c. Both a & b
d. None of the above

ANSWER: (a) Internal Program Memory

35) Which value of disc capacitors is preferred or recommended especially when the quartz crystal is connected externally in an oscillator circuit of 8051?

a. 10 pF
b. 20 pF
c. 30 pF
d. 40 pF

ANSWER: (c) 30 pF

36) Why are the resonators not preferred for an oscillator circuit of 8051?

a. Because they do not avail for 12 MHz higher order frequencies
b. Because they are unstable as compared to quartz crystals
c. Because cost reduction due to its utility is almost negligible in comparison to total cost of microcontroller board
d. All of the above

ANSWER: (d) All of the above

37) Which version of MCS-51 requires the necessary connection of external clock source to XTAL2 in addition to the XTAL1 connectivity to ground level?

a. HMOS
b. CHMOS
c. CMOS
d. All of the abov

ANSWER: (a) HMOS

38) Which signal from CPU has an ability to respond the clocking value of D- flipflop (bit latch) from the internal bus?

a. Write-to-Read Signal
b. Write-to-Latch Signal
c. Read-to-Write Signal
d. Read-to-Latch Signal

ANSWER: (b) Write-to-Latch Signal

39) Which among the below mentioned statements are precisely related to quasi-bidirectional port?

a. Fixed high pull-up resistors are internally connected
b. Configuration in the form of input pulls the port at higher position whereas they get pulled lower when configured as a source current
c. It is possible to drive the pin as output at any duration when FET gets turned OFF for an input function
d. Upper pull-up FET is always OFF with the provision of 'open-drain' output pin for normal operation of port

a. A, B, C, D
b. A, B & C
c. A & B
d. C & D

ANSWER: (b) A, B & C

40) What happens when the pins of port 0 & port 2 are switched to internal ADDR and ADDR / DATA bus respectively while accessing an external memory?

a. Ports cannot be used as general-purpose Inputs/Outputs
b. Ports start sinking more current than sourcing
c. Ports cannot be further used as high impedance input
d. All of the above

ANSWER: (a) Ports cannot be used as general-purpose Inputs/Outputs

41) The upper 128 bytes of an internal data memory from 80H through FFH usually represent _______.

a. general-purpose registers
b. special function registers
c. stack pointers
d. program counters

ANSWER: (b) special function registers

42) What is the bit addressing range of addressable individual bits over the on-chip RAM?

a. 00H to FFH
b. 01H to 7FH
c. 00H to 7FH
d. 80H to FFH

ANSWER: (c) 00H to 7FH

43) What is the divisional range of program memory for internal and external memory portions respectively when enable access pin is held high (unity)?

a. 0000H – 0FFFH & 1000H – FFFFH
b. 0000H – 1000H & 0FFFH – FFFFH
c. 0001H – 0FFFH & 01FFH – FFFFH
d. None of the above

ANSWER: (a) 0000H – 0FFFH & 1000H – FFFFH

44) Consider the following statements. Which of them is/are correct in case of program execution related to program memory?

a. External Program memory execution takes place from 1000H through 0FFFFH only when the status of EA pin is high (1)
b. External Program memory execution takes place from 0000H through 0FFFH only when the status of EA pin is low (0)
c. Internal Program execution occurs from 0000H through 0FFFH only when the status of EA pin is held low (0)
d. Internal program memory execution occurs from 0000H through 0FFFH only when EA pin is held high (1)

a. A & C
b. B & D
c. A & B
d. Only A

ANSWER: (b) B & D

45) How does the processor respond to an occurrence of the interrupt?

a. By Interrupt Service Subroutine
b. By Interrupt Status Subroutine

c. By Interrupt Structure Subroutine
d. By Interrupt System Subroutine

ANSWER: (a) By Interrupt Service Subroutine

46) Which address/location in the program memory is supposed to get occupied when CPU jump and execute instantaneously during the occurrence of an interrupt?

a. Scalar
b. Vector
c. Register
d. All of the above

ANSWER: (b) Vector

47) Which location specify the storage/loading of vector address during the interrupt generation?

a. Stack Pointer
b. Program Counter
c. Data Pointer
d. All of the above

ANSWER: (b) Program Counter

48) Match the following :

a. ISS ———————————— 1. Monitors the status of interrupt pin
b. IER ———————————— 2. Allows the termination of ISS
c. RETI ———————————— 3. MCS-51 Interrupts Initialization
d. INTO ———————————— 4. Occurrence of high to low transition level

a. A-1, B-2, C-3, D-4
b. A-3, B-2, C-4, D-1
c. A-1, B-3, C-2, D-4
d. A-4, B-3, C-2, D-1

ANSWER:(c) A-1, B-3, C-2, D-4

49) What kind of triggering configuration of external interrupt intimate the signal to stay low until the generation of subsequent interrupt?

a. Edge-Triggering
b. Level Triggering
c. Both a & b
d. None of the above

ANSWER: (b) Level Triggering

50) Which among the below mentioned reasons is/are responsible for the generation of Serial Port Interrupt?

a. Overflow of timer/counter 1
b. High to low transition on pin INT1
c. High to low transition on pin INT0
d. Setting of either TI or RI flag

a. A & B
b. Only B
c. C & D
d. Only D

ANSWER: (d) Only D

51) What is the counting rate of a machine cycle in correlation to the oscillator frequency for timers?

a. 1 / 10
b. 1 / 12
c. 1 / 15
d. 1 / 20

ANSWER: (b) 1 / 12

52) Which special function register play a vital role in the timer/counter mode selection process by allocating the bits in it?

a. TMOD
b. TCON
c. SCON
d. PCON

ANSWER:(a) TMOD

53) How many machine cycle/s is/are executed by the counters in 8051 in order to detect '1' to '0' transition at the external pin?

a. One
b. Two
c. Four
d. Eight

ANSWER: (b) Two

54) Which bit must be set in TCON register in order to start the 'Timer 0' while operating in 'Mode 0'?

a. TR0
b. TF0
c. IT0
d. IE0

ANSWER: (a) TR0

55. Which among the following control/s the timer1 especially when it is configured as a timer in mode'0′, where gate and TR1 bits are attributed to be '1" in TMOD register?

a. TR1
b. External input at (INT1)
c. TF1
d. All of the above

ANSWER: (b) External input at (INT1)

56) Which timer mode exhibit the necessity to generate the interrupt by setting EA bit in IE enhancing the program counter to jump to another vector location?

a. Mode 0
b. Mode 1
c. Mode 2
d. Mode 3

ANSWER: (b) Mode 1

57. Which among the below mentioned program segments represent the correct code?

a. MOV SP, # 54 H
MOV TCON ,# 0010 0000 C
SETC ET1
SETC TR0
SJMP $
b. MOV SP, # 54H
MOV TMOD ,# 0010 0000 C
SETC ET0
SETC TR0
SJMP $
c. MOV SP, # 54 H
MOV TMOD ,# 0010 0000 C
SETC ET1
SETC TR1

SETC EA
SJMP $
d. MOV SP, # 54 H
MOV TMOD ,# 0010 0000 C
SETC ET0
SETC TR1
SETC EA
SJMP $

ANSWER: (c)

MOV SP, # 54 H
MOV TMOD ,# 0010 0000 C
SETC ET1
SETC TR1
SETC EA
SJMP $

58) What is the maximum delay generated by the 12 MHz clock frequency in accordance to an auto-reload mode (Mode 2) operation of the timer?

a. 125 µs
b. 250 µs
c. 256 µs
d. 1200 µs

ANSWER: (c) 256 µs

59) Which among the below mentioned sequence of program instructions represent the correct chronological order for the generation of 2kHz square wave frequency?

1. MOV TMOD, 0000 0010 B

2. MOV TL0, # 06H

3. MOV TH0, # 06H

4. SETB TR0

5. CPL p1.0

6. ORG 0000H

a. 6, 5, 2, 4, 1, 3
b. 6, 1, 3, 2, 4, 5

c. 6, 5, 4, 3, 2, 1
d. 6, 2, 4, 5, 1, 3

ANSWER: (b) 6, 1, 3, 2, 4, 5

60) Why is it not necessary to specify the baud rate to be equal to the number of bits per second?

a. Because each bit is preceded by a start bit & followed by one stop bit
b. Because each byte is preceded by a start byte & followed by one stop byte
c. Because each byte is preceded by a start bit & followed by one stop bit
d. Because each bit is preceded by a start byte &followed by one stop byte

ANSWER: (c) Because each byte is preceded by a start bit & followed by one stop bit

61. Why is CHMOS technology preferred over HMOS technology for designing the devices of MCS-51 family?

a. Due to higher noise immunity
b. Due to lower power consumption
c. Due to higher speed
d. All of the above

ANSWER: (d) All of the above

62. MOV A, @ R1 will:
A.copy R1 to the accumulator
B.copy the accumulator to R1
C.copy the contents of memory whose address is in R1 to the accumulator
D.copy the accumulator to the contents of memory whose address is in R1

Answer: Option C