Overview of the 8051 Microcontroller family

Real Time Computing and Programming

Engr. A. N. Aniedu
Electronic and Computer Engineering
Nnamdi Azikiwe University, Awka

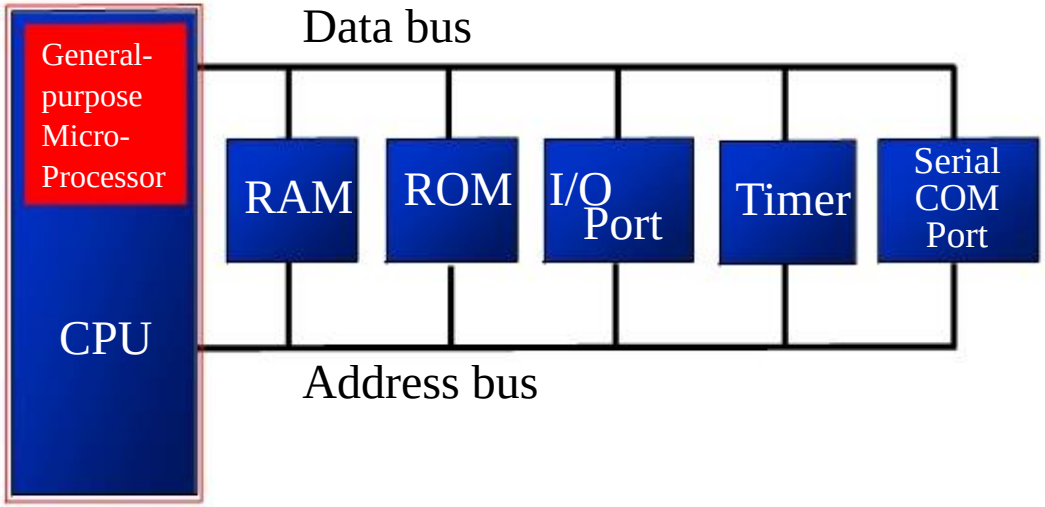
INTRODUCTION

Microcontroller vs General Purpose Microprocessor

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

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

Microcontroller vs General Purpose Microprocessor contd.



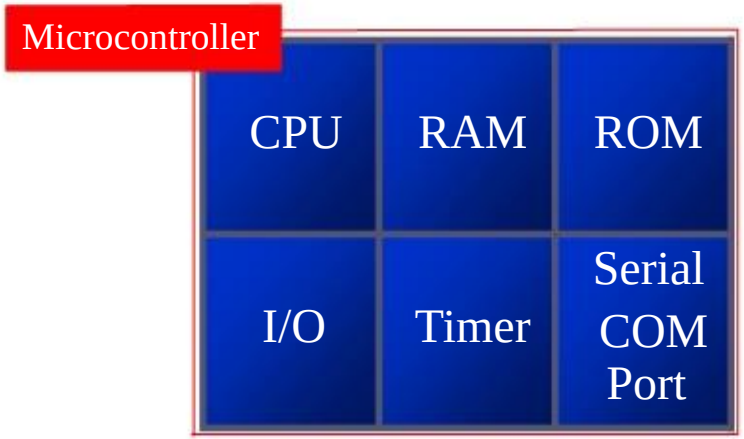


Fig1: Comparism between general purpose microprocessors and microcontrollers

Microcontroller vs General Purpose Microprocessor contd.

- General-purpose microprocessors
  - Must add RAM, ROM, I/O ports, and timers externally to make them functional
  - Make the system bulkier and much more expensive
  - Have the advantage of versatility on the amount of RAM, ROM, and I/O ports

- Microcontroller
  - The fixed amount of on-chip ROM, RAM, and number of I/O ports makes them ideal for many applications in which cost and space are critical
  - In many applications, the space it takes, the power it consumes, and the price per unit are much more critical considerations than the computing power

Microcontrollers for Embedded Systems

- An embedded product uses a microprocessor (or microcontroller) to do one task and one task only
  - There is only one application software that is typically burned into ROM
- A PC, in contrast with the embedded system, can be used for any number of applications
  - It has RAM memory and an operating system that loads a variety of applications into RAM and lets the CPU run them
  - A PC contains or is connected to various embedded products
  - Each one peripheral has a microcontroller inside it that performs only one task

Microcontrollers for Embedded Systems contd.

- Home
  - Appliances, intercom, telephones, security systems, garage door openers, answering machines, fax
  - machines, home computers, 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, computers, security systems, fax machines, microwave, copier, laser printer, color printer, paging
- Auto
  - Trip computer, engine control, air bag, ABS, instrumentation, security system, transmission control, entertainment, climate control, cellular phone, keyless entry

x86 PC Embedded Applications

- Many manufactures of general-purpose microprocessors have targeted their microprocessor for the high end of the embedded market
  - There are times that a microcontroller is inadequate for the task
- When a company targets a general- purpose microprocessor for the embedded market, it optimizes the processor used for embedded systems
- Very often the terms embedded processor and microcontroller are used interchangeably

x86 PC Embedded Applications contd.

- One of the most critical needs of an embedded system is to decrease power consumption and space
- In high-performance embedded processors, the trend is to integrate more functions on the CPU chip and let designer decide which features he/she wants to use
- In many cases using x86 PCs for the high-end embedded applications
  - Saves money and shortens development time
  - A vast library of software already written
  - Windows is a widely used and well understood platform

Choosing a Microcontroller

- 8-bit microcontrollers
  - Motorola's 6811
  - Intel's 8051
  - Atmel's AT89C51
  - Zilog's Z8
  - Microchip's PIC
- There are also 16-bit and 32-bit microcontrollers made by various chip makers

Criteria for choosing a Microcontroller

- Meeting the computing needs of the task at hand efficiently and cost effectively
  - Speed
  - Packaging
  - Power consumption
  - The amount of RAM and ROM on chip the number of I/O pins and the timer on chip
  - How easy to upgrade to higher-performance or lower power-consumption versions
  - Cost per unit

Criteria for choosing a Microcontroller contd.

- Availability of software development tools, such as compilers, assemblers, and debuggers
- Wide availability and reliable sources of the microcontroller
  - 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

OVERVIEW OF 8051 FAMILY

Overview of the 8051 Family

- 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
  - The 8051 had
    - 128 bytes of RAM
    - 4K bytes of on-chip ROM
    - Two timers
    - One serial port
    - Four I/O ports, each 8 bits wide
    - 6 interrupt sources
- The 8051 became widely popular after allowing other manufactures to make and market any flavor of the 8051, but remaining code-compatible

Overview of the 8051 Family contd.

- The 8051 is a subset of the 8052
- The 8031 is a ROM-less 8051
- 8751 microcontroller
- AT89C51 from Atmel Corporation
- DS89C4x0 from Dallas Semiconductor, now part of Maxim Corp.
- DS5000 from Dallas Semiconductor
- OTP (one-time-programmable) version of 8051
- 8051 family from Philips

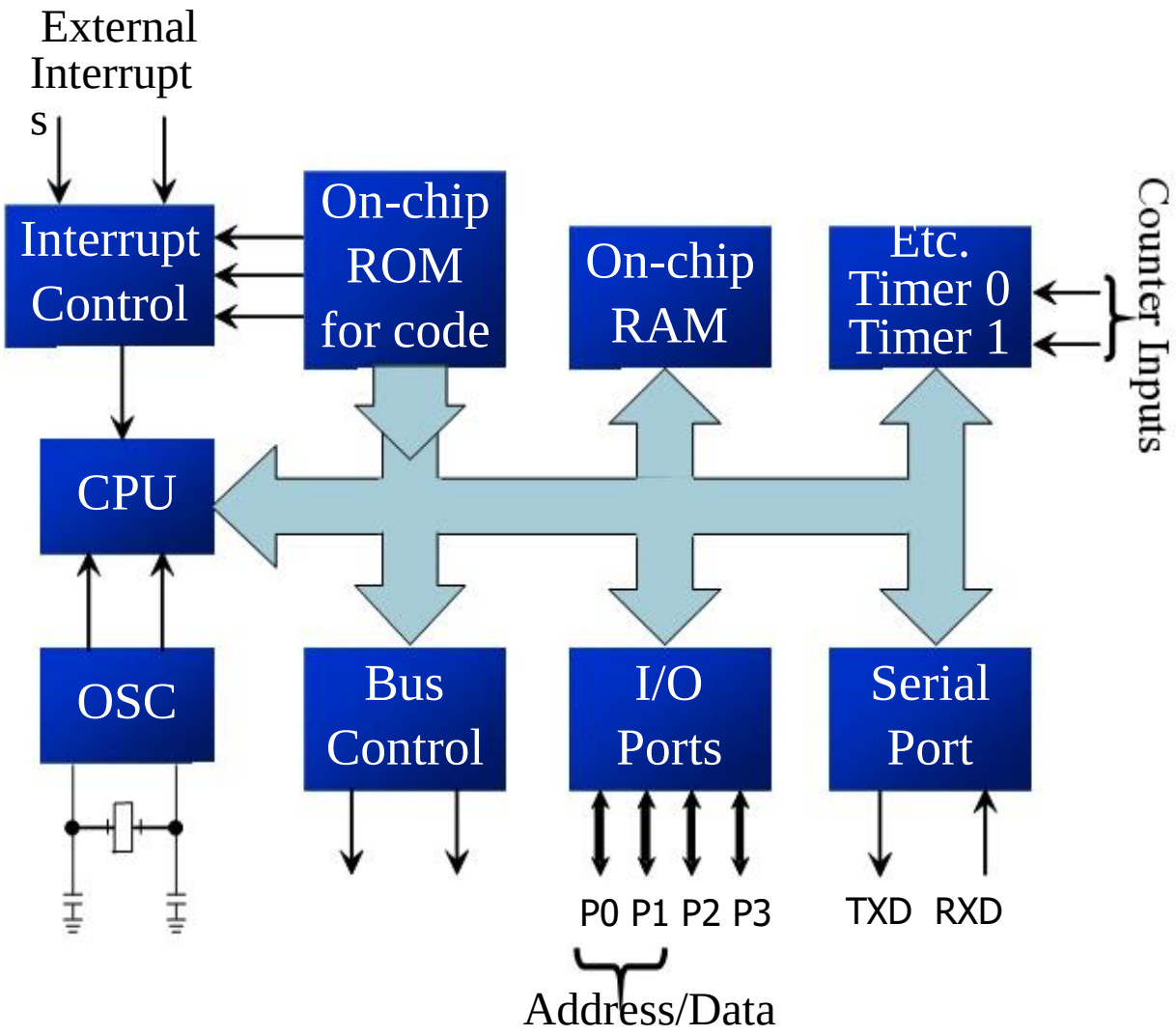


Fig 2: Overview of the 8051 microcontroller chip

The 8051 family of microcontrollers

Table 1: The 8051 family of controllers

Device number	Internal memory		Timers/ Event Counter	Interrupt Sources
	Program	Data		
8051	4K X 8 ROM	128 X 8 RAM	2 X 16-bit	5
8751H	4K X 8 EPROM	128 X 8 RAM	2 X 16-bit	5
8031	None	128 X 8 RAM	2 X 16-bit	5
8052H	8K X 8 ROM	256 X 8 RAM	3 X 16-bit	5
8752BH	8K X 8 EPROM	256 X 8 RAM	3 X 16-bit	5
8032AH	None	256 X 8 RAM	3 X 16-bit	5

8051 Architecture

8051 Architecture

- Hardware Features of the 8051
  - 0K (8031), ROM 4K (8051), EPROM 4K (8751) RAM 128 bytes (8XX1), 256 bytes (8XX2)
  - (where X= 0 or 7 & X=3 or 5)
  - Four 8-bit I/O Ports (P0 - P3)
  - Two 16-bit Timers/Counters (T0 & T1) Serial I/O Port
  - Boolean Processor (Operates on Single Bits) 210 bit-addressable locations
  - Oscillator & Clock Circuit

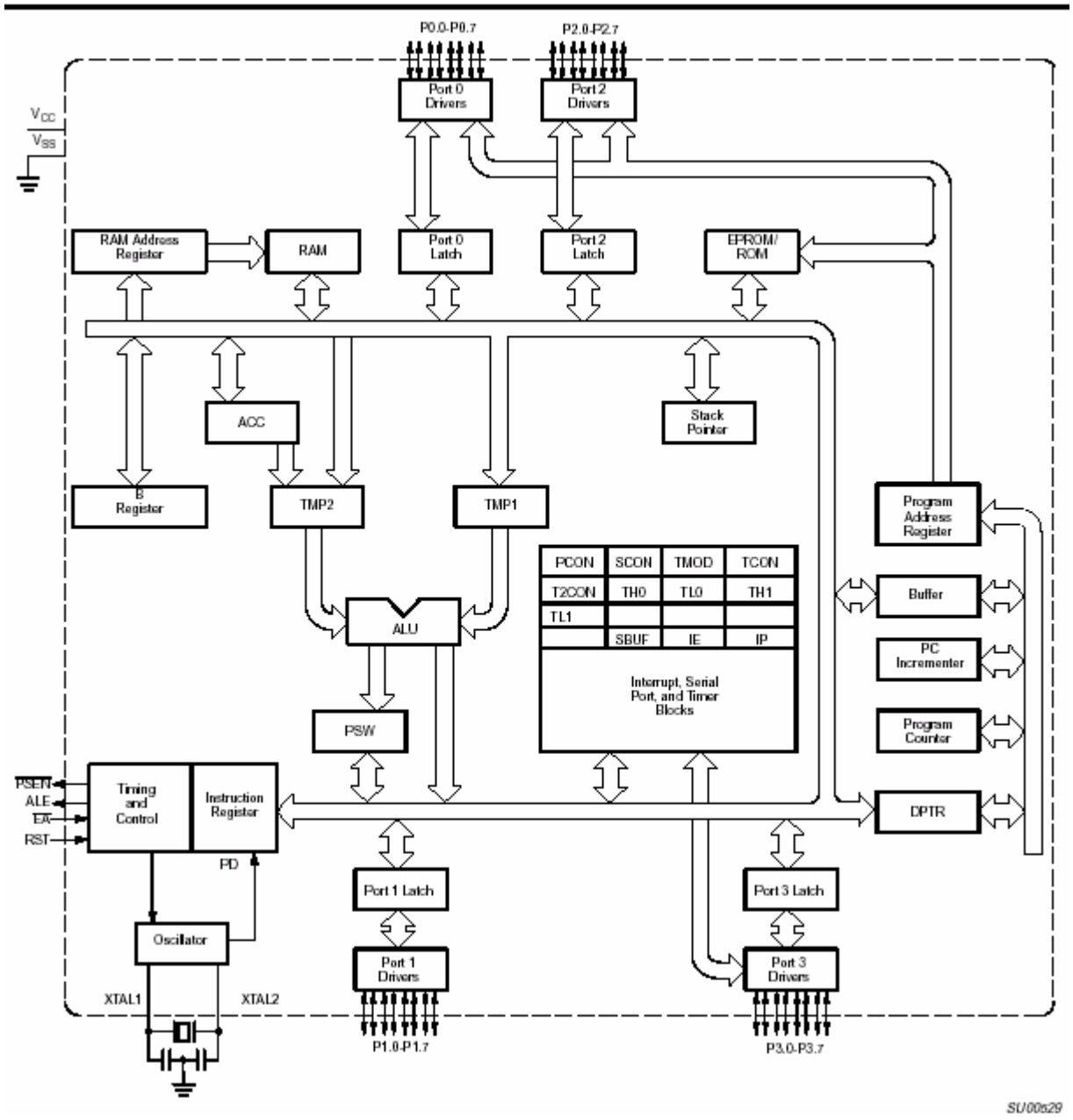


Fig 3: Block diagram of the 8051 microcontroller

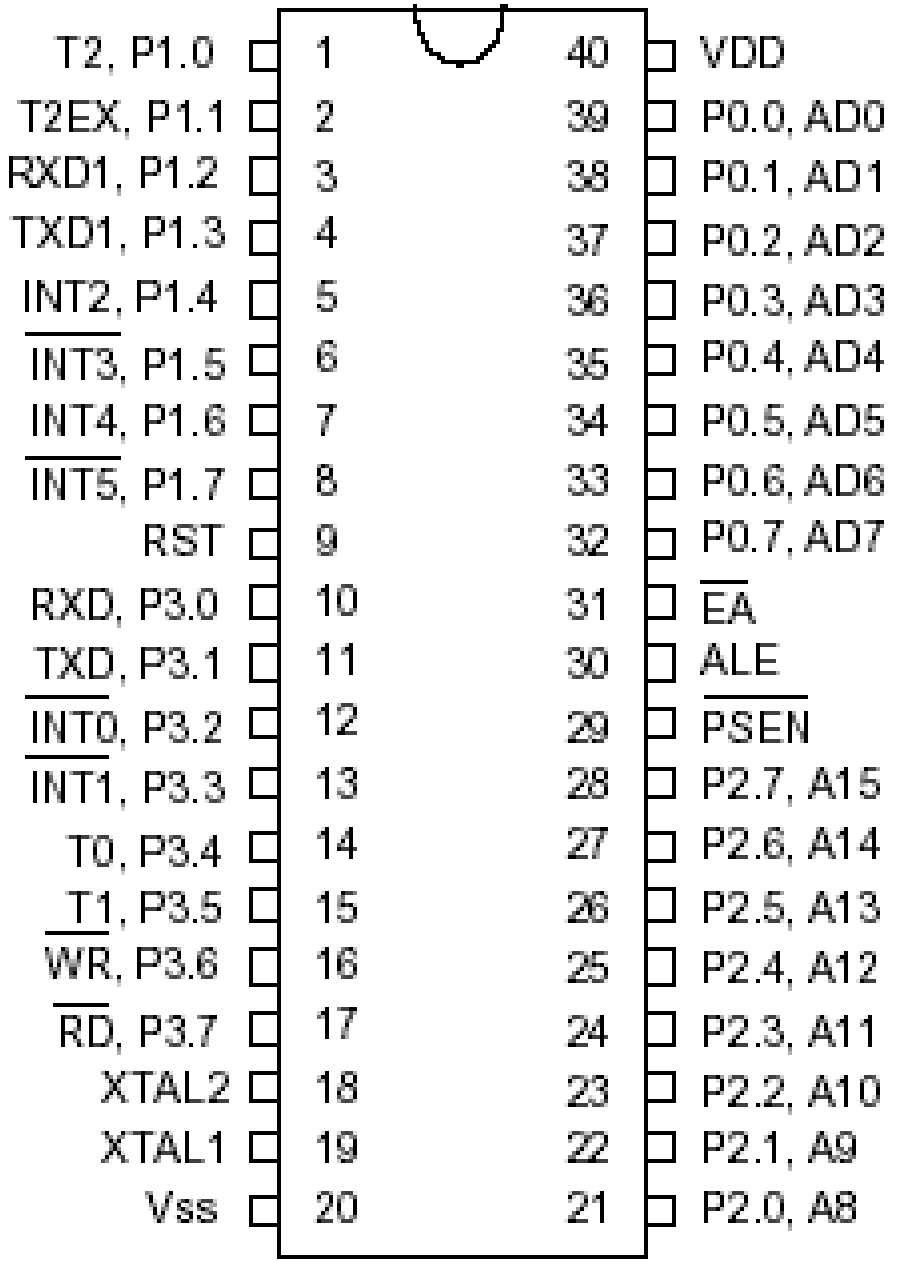


Fig 4: 8051 microcontroller Pin configuration

8051 Architecture contd.

- I/O Ports
  - Four 8-bit I/O ports.
    - Port 0, Port 1, Port 2, Port 3
  - Most have alternate functions.
  - Bi-directional.

8051 Architecture contd.

- Port 0 (pin 32 ~ 39)
  - Dual purpose I/O port.
  - In minimum component design, it is used as a general purpose I/O port.
  - In larger designs with external memory, it becomes a multiplexed data bus:
    - Low byte of address bus, strobed by ALE.
    - 8-bit instruction bus, strobed by PSEN.
    - 8-bit data bus, strobed by WR and RD.

8051 Architecture contd.

- Port 1 (pin 1 ~ 8)
  - As an I/O port:
    - Standard bi-directional port for interfacing to external devices as required for I/O.
  - Alternate functions:
    - Only on some derivatives.

8051 Architecture contd.

- Port 2 (pin 21 ~ 28)
  - Dual purpose I/O port.
  - As an I/O port:
    - Standard bi-directional general purpose I/O port.
  - Alternate functions:
    - High byte of address bus for external program and data memory accesses.

8051 Architecture contd.

- Port 3 (pin 10 ~ 17)
  - Dual purpose I/O port.
  - As an I/O port:
    - Standard bi-directional general purpose I/O port.
  - Alternate functions:

Port pin	Alternative function
P3.0	RXD (serial input port)
P3.1	TXD (serial input port)
P3.2	INT0 (external interrupt 0)
P3.3	INT1 (external interrupt 1)
P3.4	T0 (Timer 0 external input)
P3.5	T1 (Timer 1 external input)
P3.6	WR (external data memory write strobe)
P3.7	RD (external data memory read strobe)

8051 Architecture contd.

- Bus control signals
  - **PSEN** (pin 29): (Program Store Enable) enables external program (code) memory. Usually connected to EPROM's output enable (OE). It pulses low during fetch stage of an instruction. It remains high while executing a program
  - **ALE/PROG** (pin 30): (Address Latch Enable) used for demultiplexing the address and data bus when port 0 is used as the data bus and low-byte of address bus.
  - **EA/VPP** (pin 31): (External Access) high to execute programs from internal ROM and low to execute from external
  - **RST** (pin 9): (RESET) master reset of 8051.

8051 Architecture contd.

- Oscillator and power pins
  - Pins 18 and 19 are the oscillator pins to connect the crystal of nominal frequency 12 MHz.
  - Pin 40 is for +5V and pin 20 is for GND.

8051 Architecture contd.

- Addressing space
  - 64K x 8 ROM - External Program Memory.
    - (Enabled via PSEN)
  - 64K x 8 RAM - External Data Memory.
    - (Enabled via RD and WR)
  - 256 x 8 RAM - Internal Data Memory.
  - 128 x 8 Special Function Registers (SFRs).
  - Bit addressing of 16 RAM locations and 16 SFRs

8051 Architecture contd.

- Internal data memory
  - Four register banks (Register Bank 0-3): 00 to 1F hexadecimal.
  - Bit addressable RAM (128 bits): 20 to 2F hexadecimal.
  - General purpose RAM (directly addressable range): 30 to 7F hexadecimal.
  - Special function registers (indirectly addressable range): 80 to FF hexadecimal.

8051 Architecture contd.

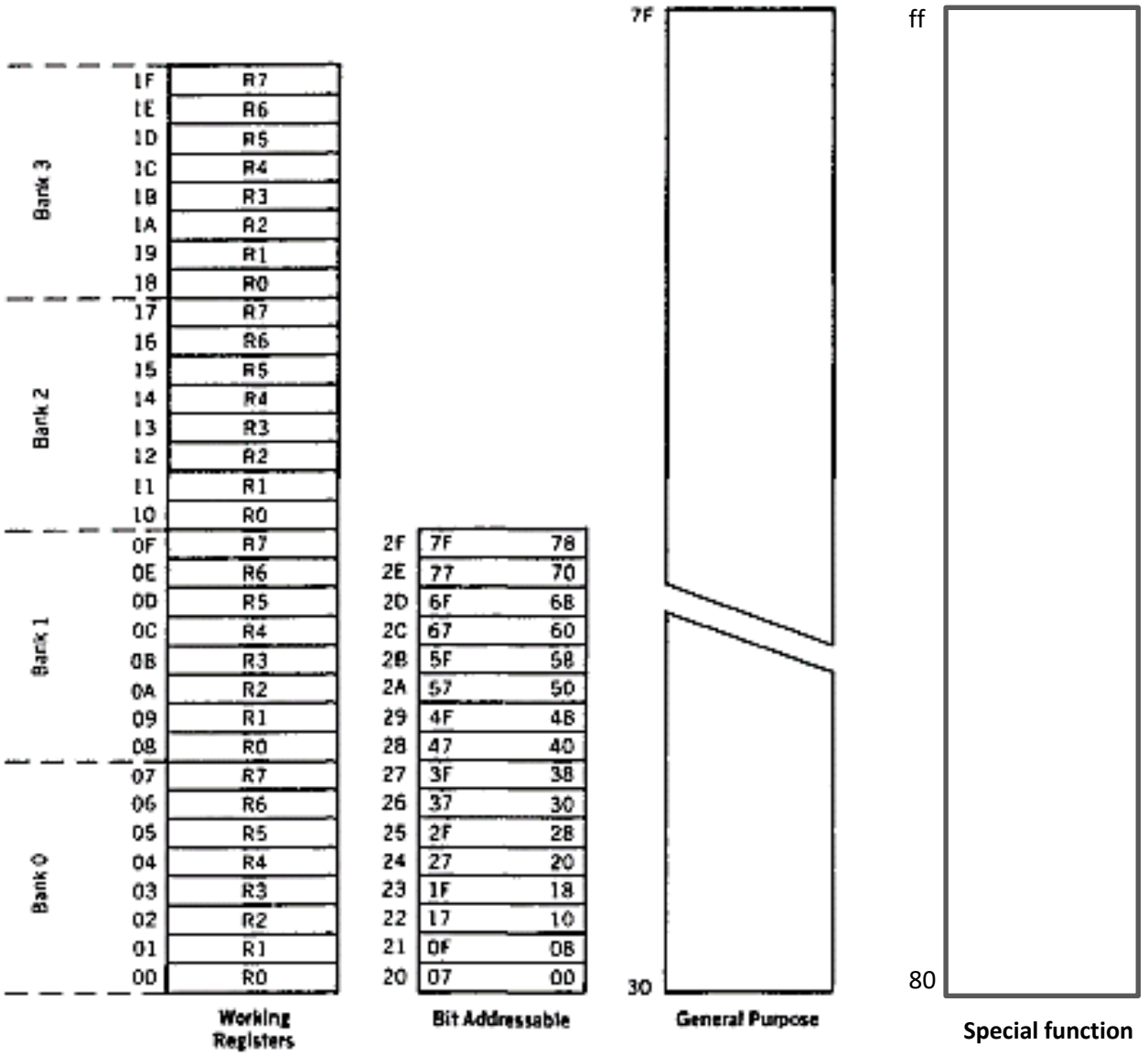


Fig 5: Internal RAM Organization

8051 Architecture contd.

- External data memory
  - 64K byte address space.
  - The only access to this memory is with the MOVX instruction, using either 16- bit data pointer DPTR, R0, or R1 as the address register.

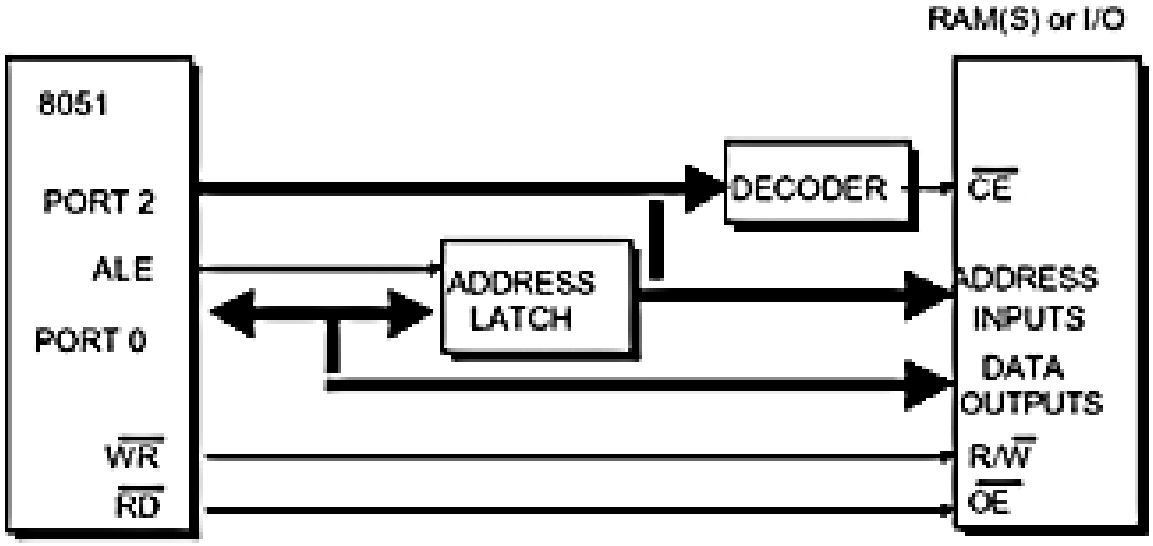


Fig 6: External data memory addressing

8051 Architecture contd.

- External program memory
  - 64K byte address space.
  - Enabled by PSEN signal.

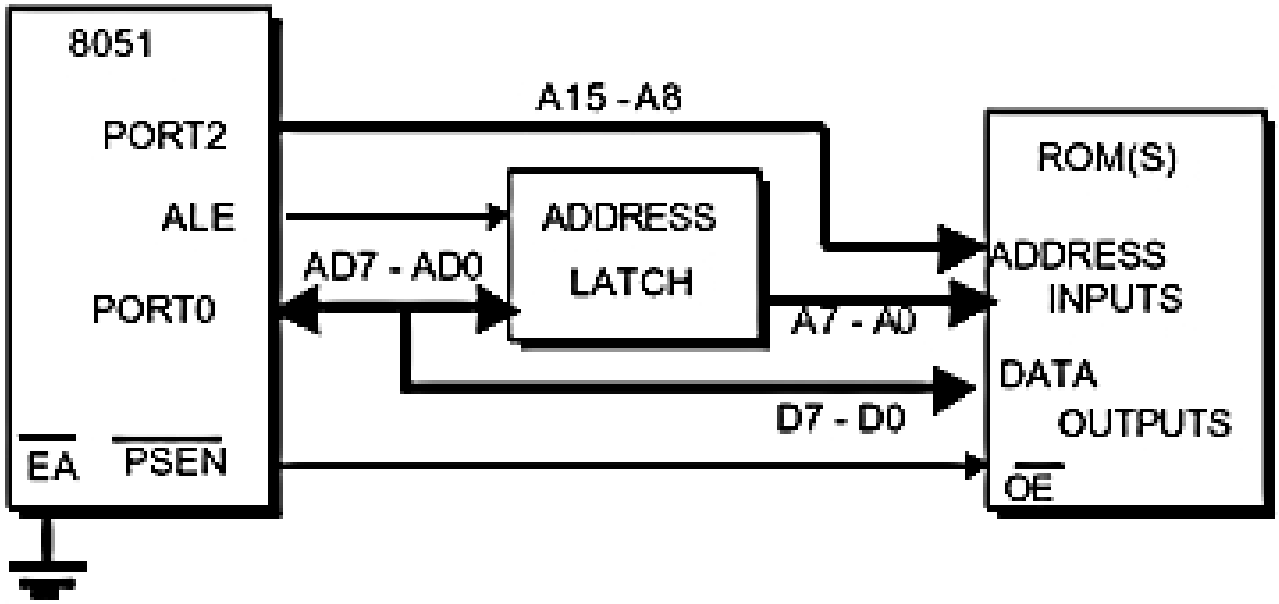


Fig 7: External program memory addressing

8051 Architecture contd.

- External bus expansion

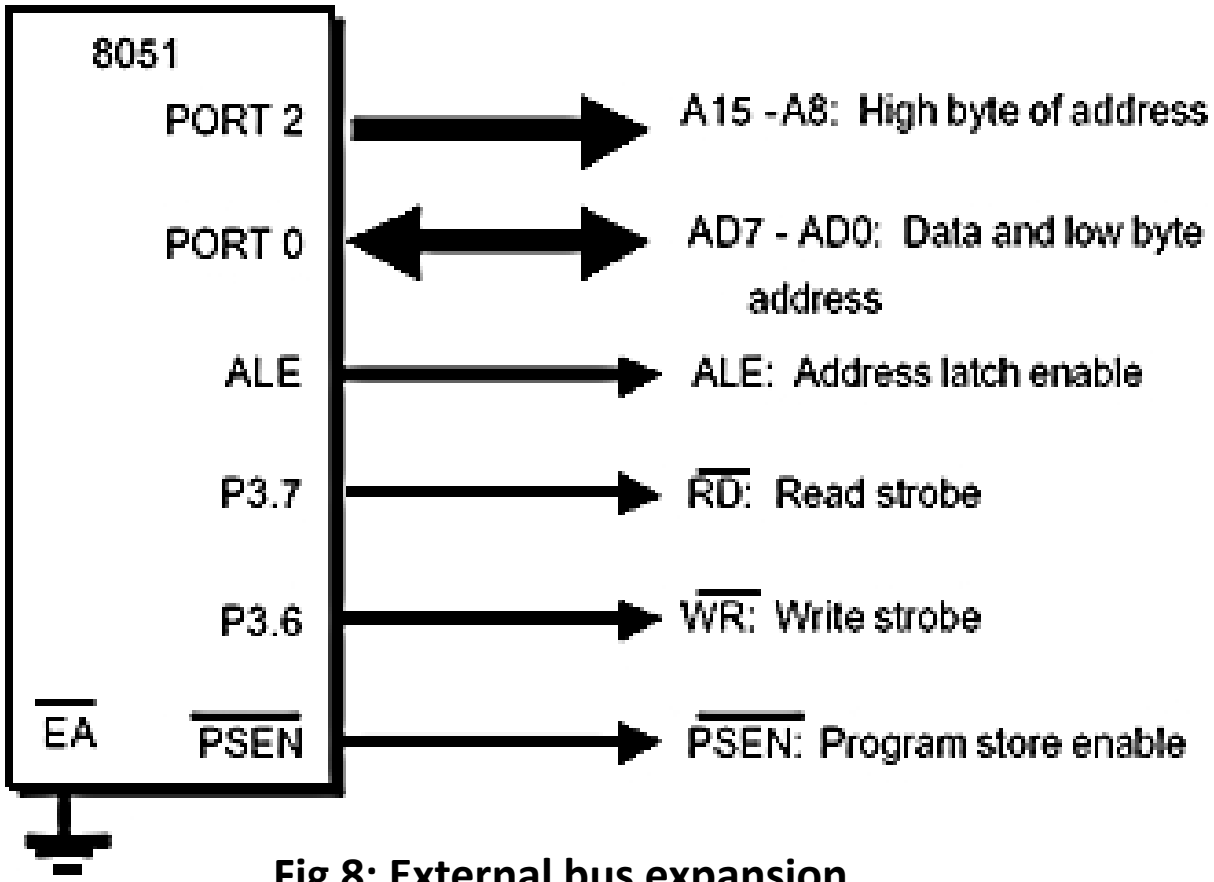


Fig 8: External bus expansion

8051 Architecture contd.

- Special function register space
  - 128 byte address space, directly addressable as 80H to FFH
  - 16 addresses are bit addressable: Set, Clear, AND, OR, MOV (those ending with 0 or 8).
  - Special function space contains:
    - Special purpose CPU registers.
    - I/O control registers.
    - I/O ports.

- Special function register map

The Special Function Registers (SFRs) [Internal Data Memory (80H to FFH)]

Register	*Address*	*Function*
P0	80H*	Port 0
SP	81H	Stack pointer
DPL	82H	Data pointer (Low)
DPH	83H	Data pointer (High)
TCON	88H*	Timer register
TMOD	89H	Timer mode register
TL0	8AH	Timer 0 low byte
TL1	8BH	Timer 1 low byte
TH0	8CH	Timer 0 high byte
TH1	8DH	Timer 1 high byte
P1	90H*	Port 1
SCON	98H*	Serial port control register
SBUF	99H	Serial port data buffer
P2	A0H*	Port 2
IE	A8H*	Interrupt enable register
P3	B0H*	Port 3
IP	B8H*	Interrupt priority register
PSW	D0H*	Program status word
ACC	E0H*	Accumulator (direct address)
B	F0H*	B register

*Bit-addressable register.

8051 Architecture contd.

- Program status word (PSW)
  - Flags are 1-bit registers provided to store the results of certain program instructions. In order to conveniently address the
    flags, they are grouped inside the PSW register.

The PSW Bit Addresses

Symbol	*Address*	*Function*
CY	D0H.7	Carry flag
AC	D0H.6	Auxiliary carry flag
F0	D0H.5	General-purpose flag
RS1	D0H.4	Register bank select (MSB)
RS0	D0H.3	Register bank select (LSB)
OV	D0H.2	Overflow flag
—	D0H.1	User-definable flag
P	D0H.0	Parity flag

8051 Architecture contd.

- Program status word (PSW)

- CY: (Carry Flag) is dual purpose:
  - (1) As traditional CY for arithmetic operations,
  - (2) As Boolean accumulator.
- AC: (Auxiliary Carry Flag) used in addition of BCD numbers, is set if a carry was generated out of bit 3 into bit 4. If the values are added are BCD, then the add instruction must be followed by DAA (decimal adjust accumulator) to bring results greater than 9 back into range.
- F0: (Flag 0) is a general-purpose flag bit available for user applications.

8051 Architecture contd.

- Program status word (PSW)

- OV: (Overflow Flag) is set after an addition or subtraction operation if there was an arithmetic overflow. Results greater than +127 or less than -128 will set OV bit.
- P: (Parity Bit) automatically set or cleared each machine cycle to establish even parity with the accumulator. Parity bit is most commonly used in conjunction with serial port routines to include a parity bit before or after the transmission.
- RS1 and RS0 are used to select different register banks.

MICROCONTROLLER LCD AND KEYBOARD INTERFACING

LCD Interfacing

- LCD is finding widespread use and replacing LEDs due to:
  - 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

- Fig below shows a typical interfacing of and LCD to an 8051 microcontroller

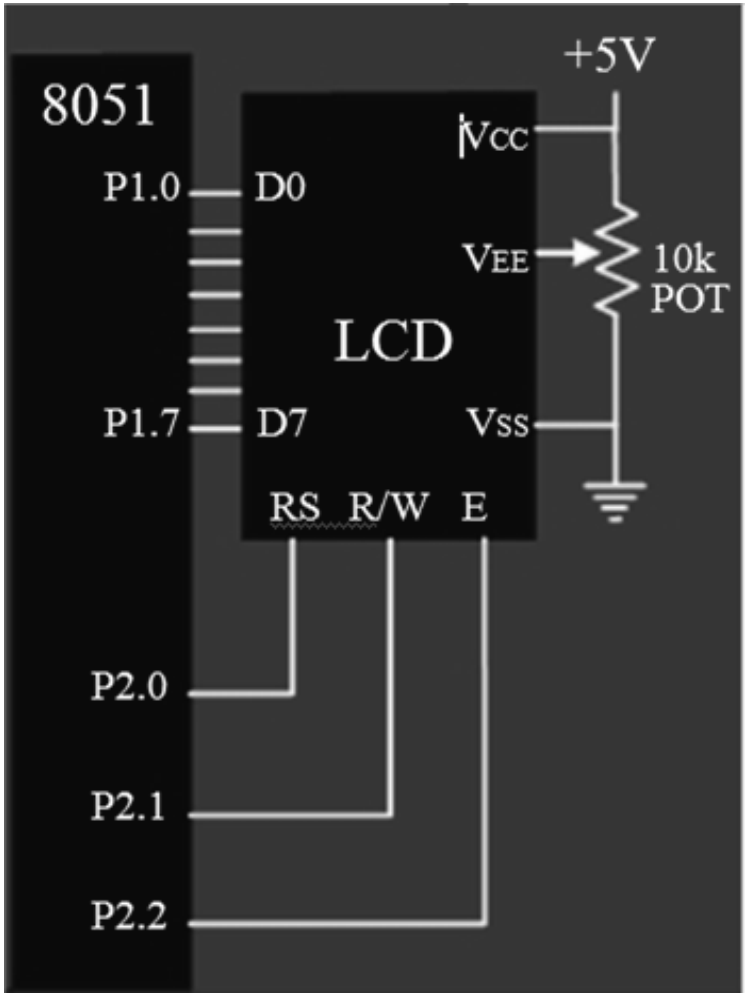


Fig 9: Connecting the LCD to the microcontroller

Pin Descriptions for LCD

Pin	Symbol	I/O	Descriptions
1	VSS	--	Ground
2	VCC	--	+5V power supply
3	VEE	--	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 -used by the LCD to latch information presented to its data bus
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

- Pins 7-14 are used to send displayed information or instruction command codes to the LCD and read the contents of the LCD’s internal registers

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 to 1st line
C0	Force cursor to beginning to 2nd line
38	2 lines and 5x7 matrix

- 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.

```
;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 port 1
            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,#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
```

Keyboard Interfacing

- **Keyboards** are organized in a matrix of rows and columns
  - The CPU accesses both rows and columns through ports
  - Therefore, 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 single microcontroller takes care of hardware and software interfacing

- A 4x4 matrix connected to two ports
  - The rows are connected to an output port and the columns are connected to an input port

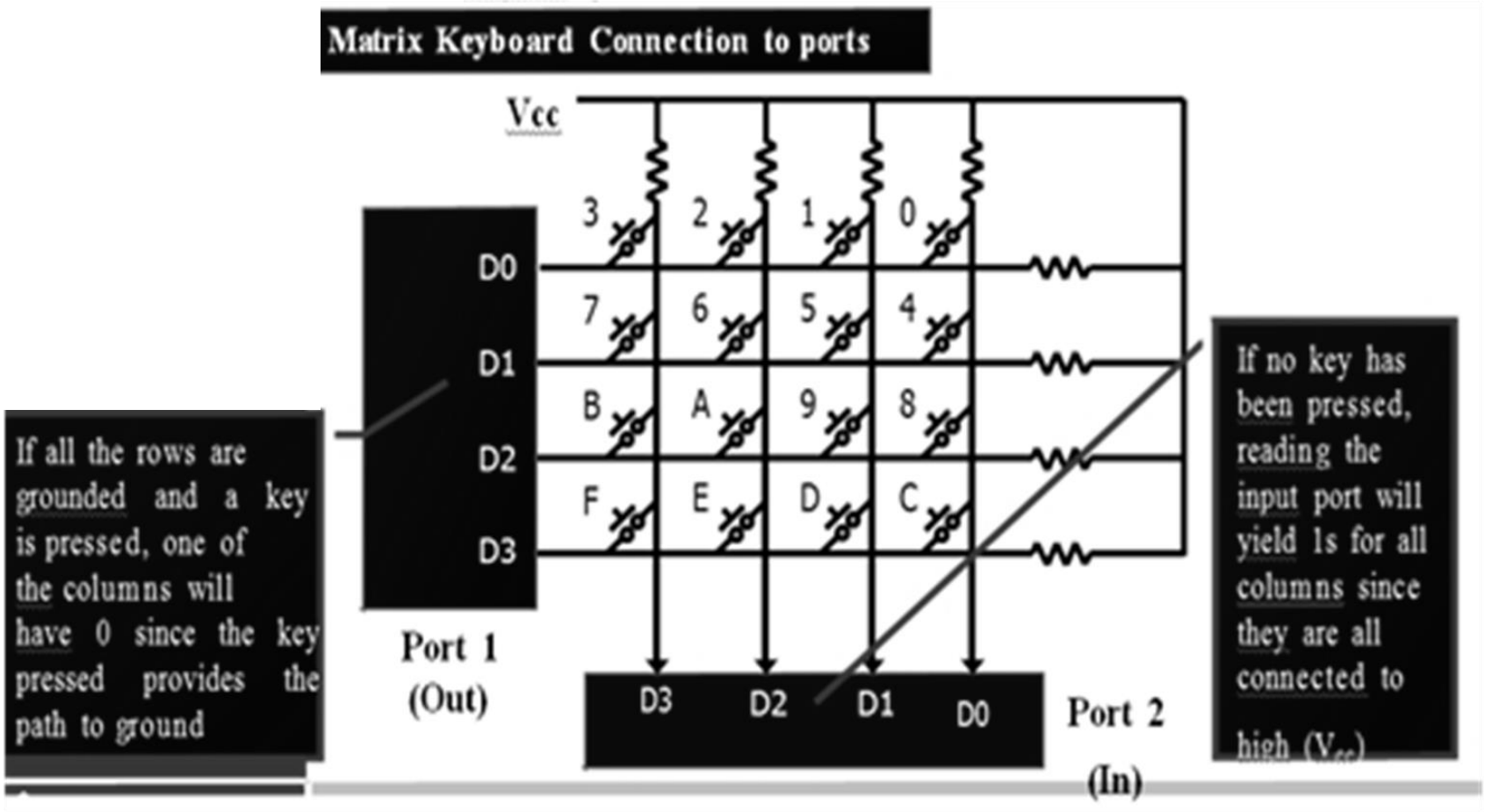


Fig 10: matrix keyboard connection to microcontroller ports

- It is the function of the microcontroller to scan the keyboard continuously to detect and identify the key pressed
- 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 and 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

- After detecting a key press, 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 and the process is moved to the next row
    - It grounds the next row, reads the columns, and checks for any zero. This process continues until the row is identified
- After identification of the row in which the key has been pressed, it finds out which column the pressed key belongs to

Example

From Figure, 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 :

(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

- Program for detection and identification of key activation goes through the following stages (fig 11):

1. 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

2. 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
     a) it ensures that the first key press detection was not an erroneous one due a spike noise
     b) the key press. If after the 20-ms delay the key is still pressed, it goes back into the loop to detect a real key press

3. 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. Therefore, 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
4. 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

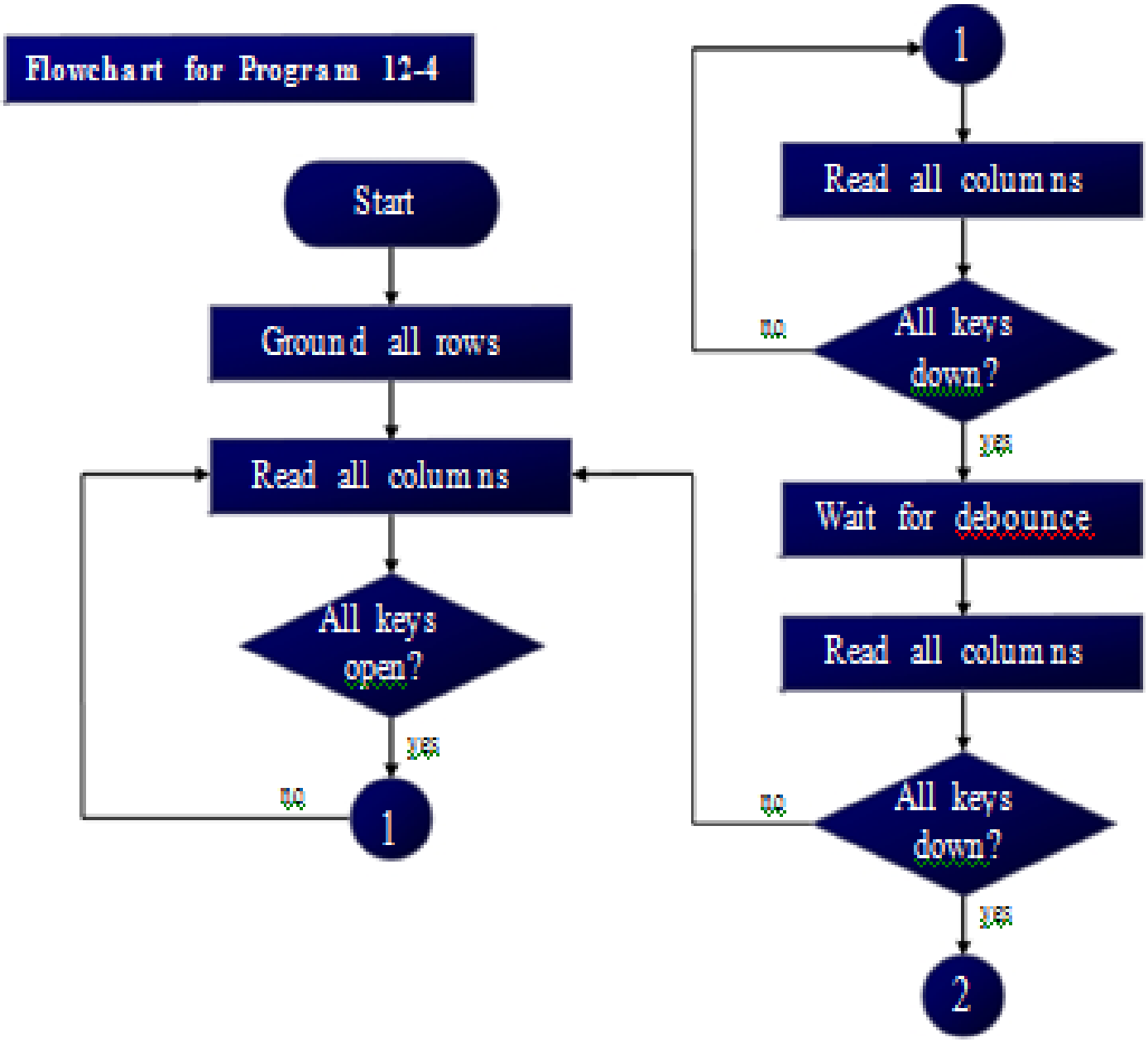


Fig 11a: Flowchart for key activation identification

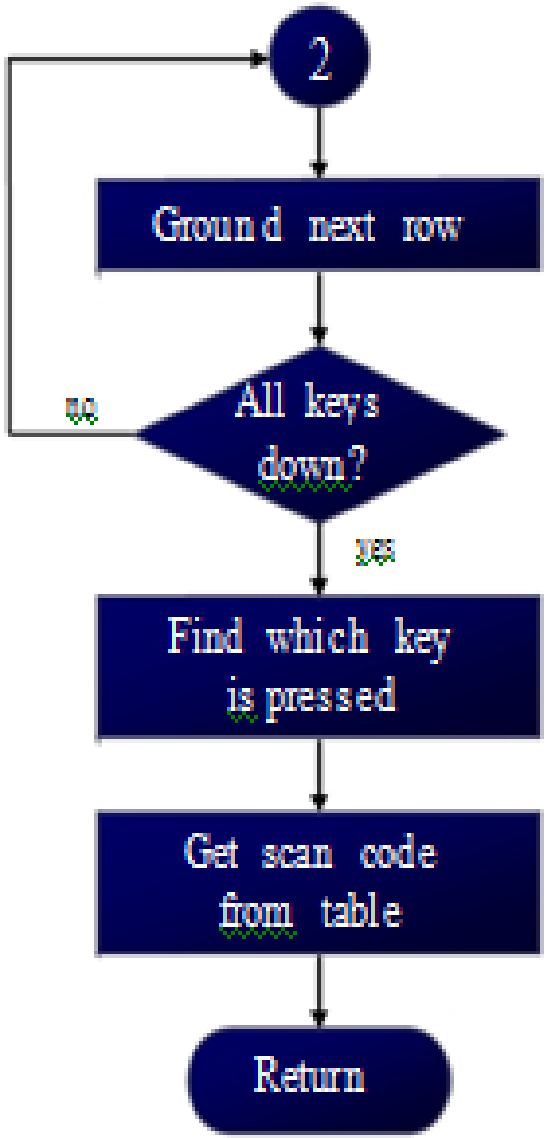


Fig 11b: Flowchart for key activation identification

THANK YOU