

## 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

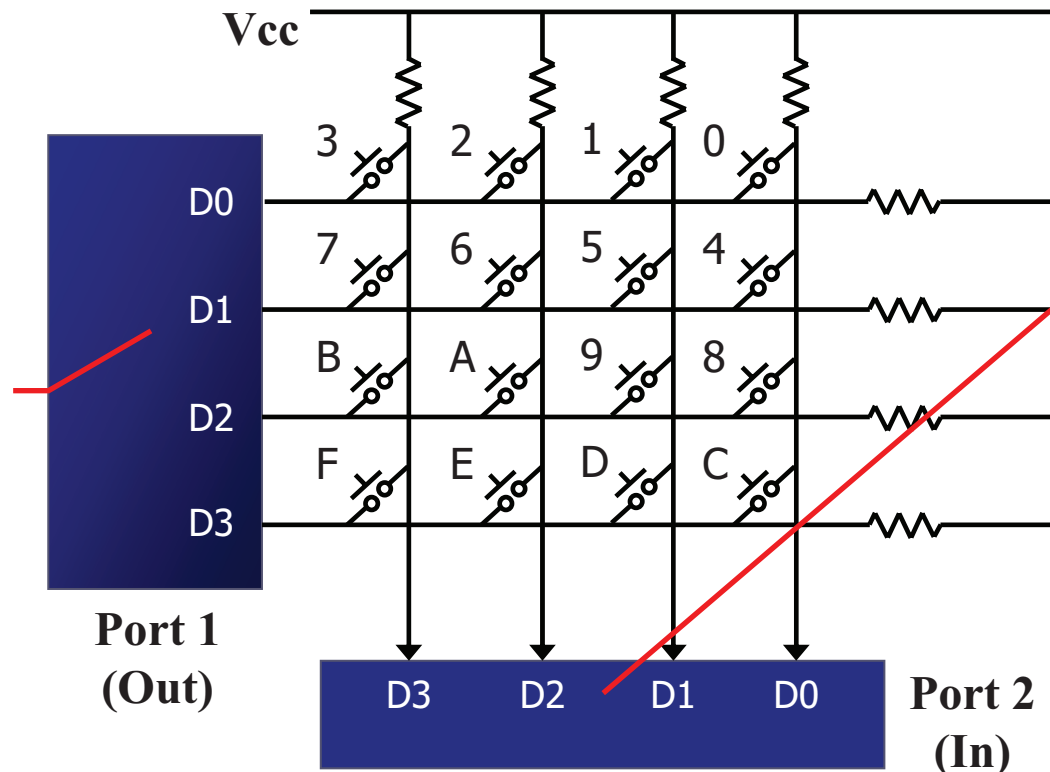
# KEYBOARD INTERFACING

## Scanning and Identifying the Key

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

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

**Matrix Keyboard Connection to ports**



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

## KEYBOARD INTERFACING

### Grounding Rows and Reading Columns

- ❑ 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

# KEYBOARD INTERFACING

## Grounding Rows and Reading Columns (cont')

- ❑ 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
  - Find out which column the pressed key belongs to

# KEYBOARD INTERFACING

## Grounding Rows and Reading Columns (cont')

### Example 12-3

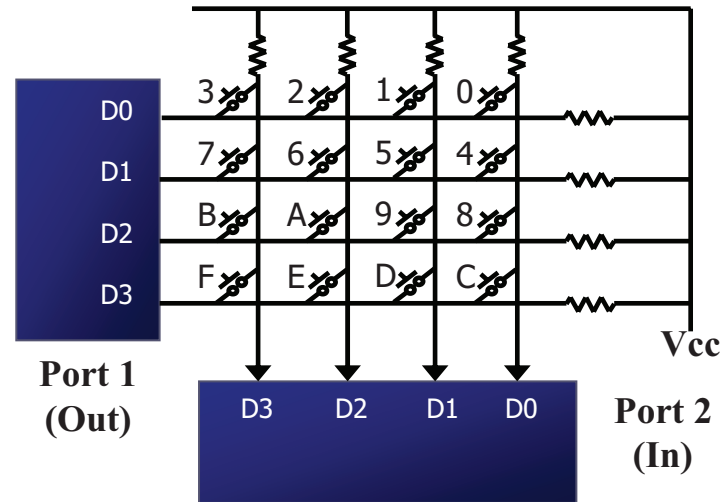
From Figure 12-6, identify the row and column of the pressed key for each of the following.

- (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 13-5 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.



## KEYBOARD INTERFACING

### Grounding Rows and Reading Columns (cont')

- ❑ Program 12-4 for detection and identification of key activation goes through the following stages:
  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

# KEYBOARD INTERFACING

## Grounding Rows and Reading Columns (cont')

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

# KEYBOARD INTERFACING

## Grounding Rows and Reading Columns (cont')

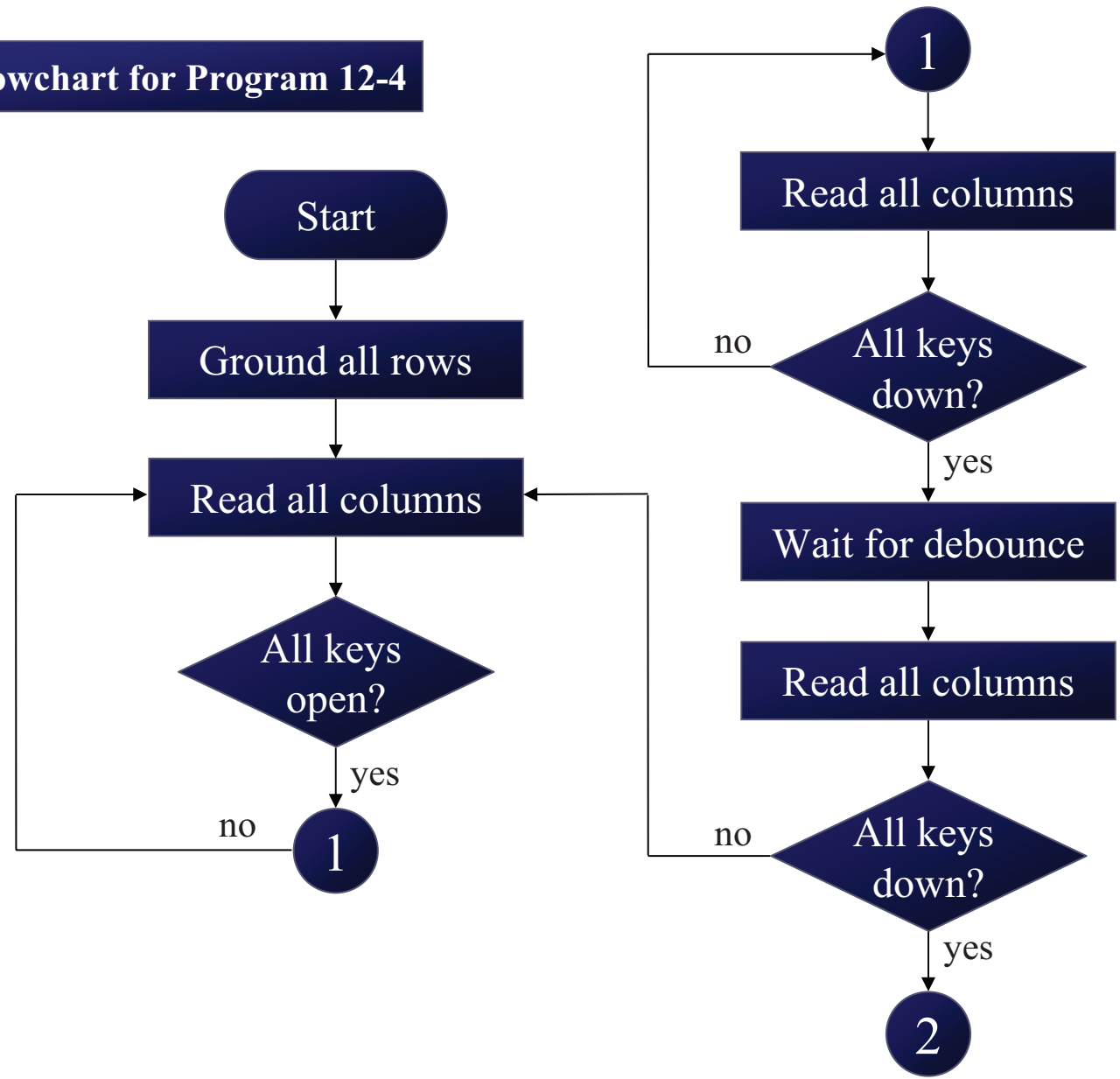
3. To detect which row key press belongs to, it grounds one row at a time, reading the columns each time
  - If it finds that all columns are high, this means that the key press cannot belong to that row
    - Therefore, it grounds the next row and continues until it finds the row the key press belongs 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



# KEYBOARD INTERFACING

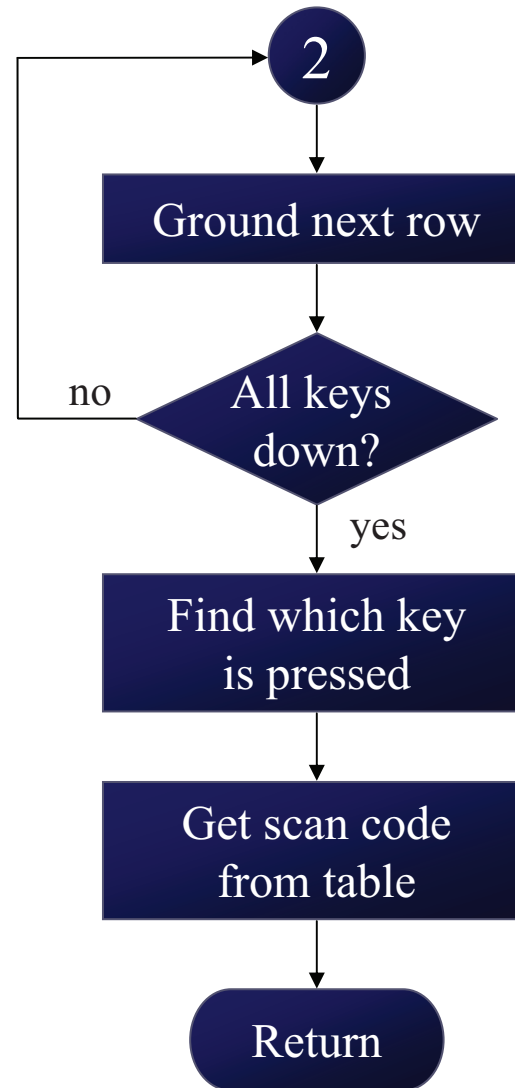
## Grounding Rows and Reading Columns (cont')

Flowchart for Program 12-4



# KEYBOARD INTERFACING

## Grounding Rows and Reading Columns (cont')



# KEYBOARD INTERFACING

## Grounding Rows and Reading Columns (cont')

### Program 12-4: Keyboard Program

;keyboard subroutine. This program sends the ASCII  
;code for pressed key to P0.1  
;P1.0-P1.3 connected to rows, P2.0-P2.3 to column

```

                MOV    P2,#0FFH    ;make P2 an input port
K1:             MOV    P1,#0        ;ground all rows at once
                MOV    A,P2        ;read all col
                                ;(ensure keys open)
                ANL    A,00001111B    ;masked unused bits
                CJNE   A,#00001111B,K1 ;till all keys release
K2:             ACALL  DELAY        ;call 20 msec delay
                MOV    A,P2        ;see if any key is pressed
                ANL    A,00001111B    ;mask unused bits
                CJNE   A,#00001111B,OVER;key pressed, find row
                SJMP   K2            ;check till key pressed
OVER:           ACALL  DELAY        ;wait 20 msec 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
. . . .
```

# KEYBOARD INTERFACING

## Grounding Rows and Reading Columns (cont')

```
.....  
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 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 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 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 col.  
        LJMP K2                ;if none, false input,  
                                ;repeat  
.....
```

# KEYBOARD INTERFACING

## Grounding Rows and Reading Columns (cont')

```
.....
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
        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 low
        JNC  MATCH             ;if zero, get ASCII code
        INC  DPTR               ;point to next col. addr
        SJMP FIND              ;keep searching
MATCH:  CLR  A                  ;set A=0 (match is found)
        MOVC A,@A+DPTR         ;get ASCII 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
```