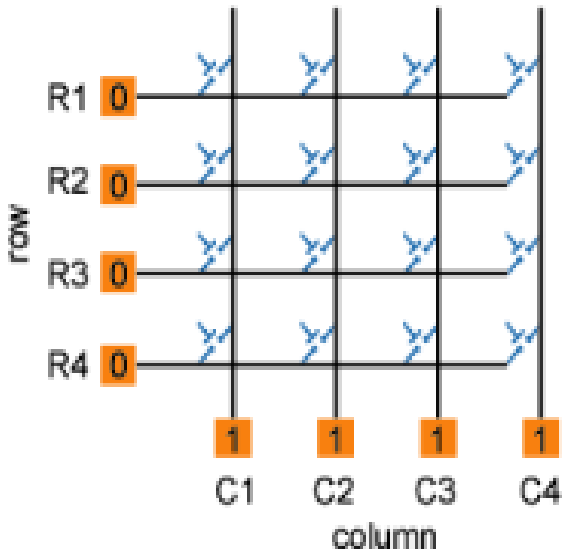
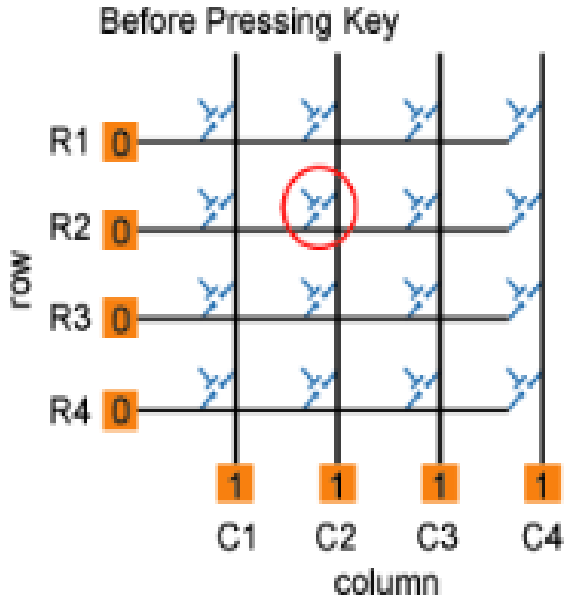
Keypad interfacing (port multiplexing)

- Keyboards are organized in a matrix of rows and columns
- When a key is pressed a row and a column make contact, otherwise there is no connection between them
- With 8-bit PORT 4x4 matrix of keys can be connected to a microcontroller.

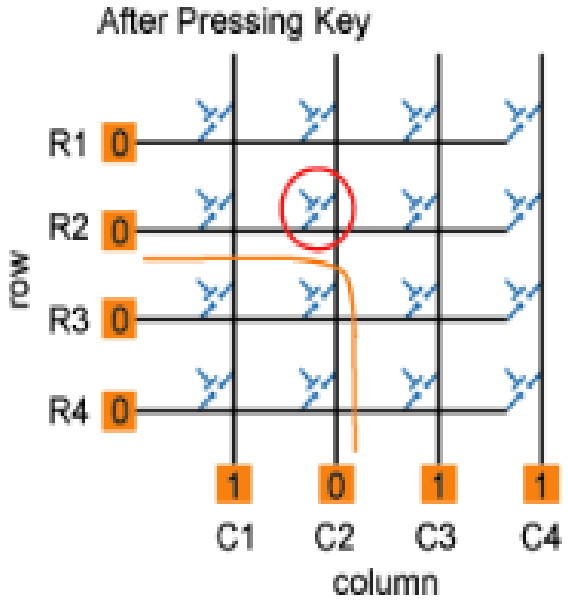
Keypad Matrix Basics





R1 R2 R3 R4 = 0 0 0 0

C1 C2 C3 C4 = 1 1 1 1



R1 R2 R3 R4 = 0 0 0 0

C1 C2 C3 C4 = 1 0 1 1

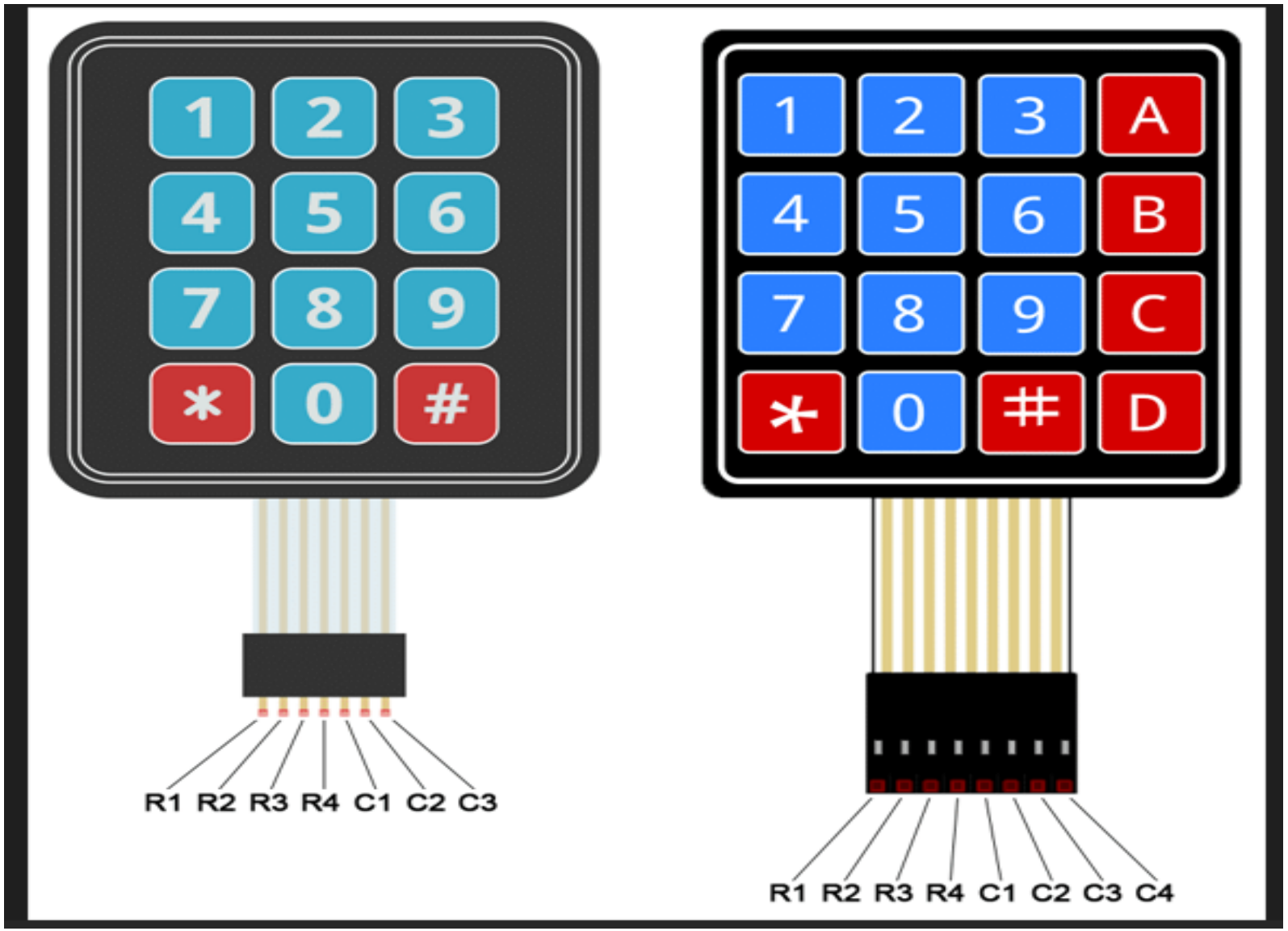
How it works ???

- The hex keypad has 8 communication lines namely
  - R1, R2, R3, and R4
  - C1, C2, C3, and C4
- R1 to R4 represents the four rows
- C1 to C4 represents the four columns
- When a particular key is pressed the corresponding row and column to which the terminals of the key are connected gets shorted.
- For example if key 1 is pressed row R1 and column C1 gets shorted and so on

Key Scanning:

- To detect a pressed key, the microcontroller grounds all rows by providing 0 to the output latch, and then it reads the columns
- If the data read from columns is = 1111, no key has been pressed
- After pressing key, it makes contact of row with column
- If one of the column bits has a zero, this means that a key press has occurred.
  For example, if C1:C4 = 1011, this means that a key in the C2 column has been pressed.
- After detecting a key press, microcontroller will go through the process of identifying the key.

- Starting from the top row, the microcontroller will ground it by providing a low to row R1 only.
- Now read the columns, if the data read is all 1s, no key in that row is pressed and the process continues for the next row.
- So, now ground the next row, R2. Read the columns, check for any zero and this process continues until the row is identified.
  E.g. In above case we will get row 2 in which column is not equal to 1111. So, after identification of the row in which the key has been pressed we can easily find out the key by row and column value.


1 2 3
4 5 6
7 8 9
* 0 #
R1 R2 R3 R4 C1 C2 C3
1 2 3 A
4 5 6 B
7 8 9 C
* 0 # D
R1 R2 R3 R4 C1 C2 C3 C4

```c
// Keypad row and column pins
#define ROWS (BIT4 | BIT5 | BIT6 | BIT7)                    // Rows: P1.4 to P1.7
#define COLS (BIT0 | BIT1 | BIT2 | BIT3)                    // Columns: P5.0 to P5.3

void setupLCD(void);
void setupKeypad(void);
unsigned char scanKeypad(void);
void displayNumber(unsigned char number);

int main(void)
{
    WDTCTL = WDTPW | WDTHOLD;                               // Stop watchdog timer

    // Disable GPIO power-on high-impedance mode
    PM5CTL0 &= ~LOCKLPM5;

    setupLCD();
    setupKeypad();

    unsigned char key;
```

```
    while (1)
    {
        key = scanKeypad();                         // Scan the keypad for input
        if (key != 0xFF)                            // If a key is pressed
        {
            displayNumber(key);                     // Display the corresponding numbe
        }
    }
}
```

```
void setupKeypad(void)
{
    // Configure rows as outputs and columns as inputs
    P1DIR |= ROWS;                                          // Rows as outputs
    P1OUT &= ~ROWS;                                         // Initialize rows to 0

    P5DIR &= ~COLS;                                         // Columns as inputs
    P5REN |= COLS;
    // Enable pull-up/pull-down resistors
    P5OUT |= COLS;                                          // Set pull-up resistors
}
```

```c
unsigned char scanKeypad(void)
{
    unsigned char row, col;
    unsigned char keyMap[4][4] = {
        {1, 2, 3, 'A'},
        {4, 5, 6, 'B'},
        {7, 8, 9, 'C'},
        // 0xFF indicates no valid key
        {0xFF, 0, 0xFF, 0xFF}
    };

    for (row = 0; row < 4; row++)
    {
        P1OUT = ~(BIT4 << row);
        // Drive one row low at a time
        __delay_cycles(1000);
        // Small delay for signal stabilization

        for (col = 0; col < 4; col++)
        {
            if (!(P5IN & (BIT0 << col)))
                // Check if a column is low
            {
                return keyMap[row][col];
                // Return the corresponding key
            }
        }
    }

    return 0xFF;                                                    // No key pressed
}
```

```
void displayNumber(unsigned char number)
{
    if (number < 10)                                    // Ensure the number is valid
    {
        LCDMEM[pos1] = digit[number];                   // Display the number on the LCD
    }
}
```

TASKS:

- 1) Run the program given in the lecture
- 2) Display Number from to 1 to 9 on the onboard LCD (available on the Launchpad) of the MSP430Fr4133 MCU.

(Home Task)

- 2) Display Numbers from 1 to 9 on the seven segment display and ON the corresponding LED's attached with any PORT of the MCU

Note: Attach the seven segment with P3 or any other PORT and keypad with P1 or any other Port (you can use proteus).