Interfacing a matrix keypad

Interfacing a matrix keypad

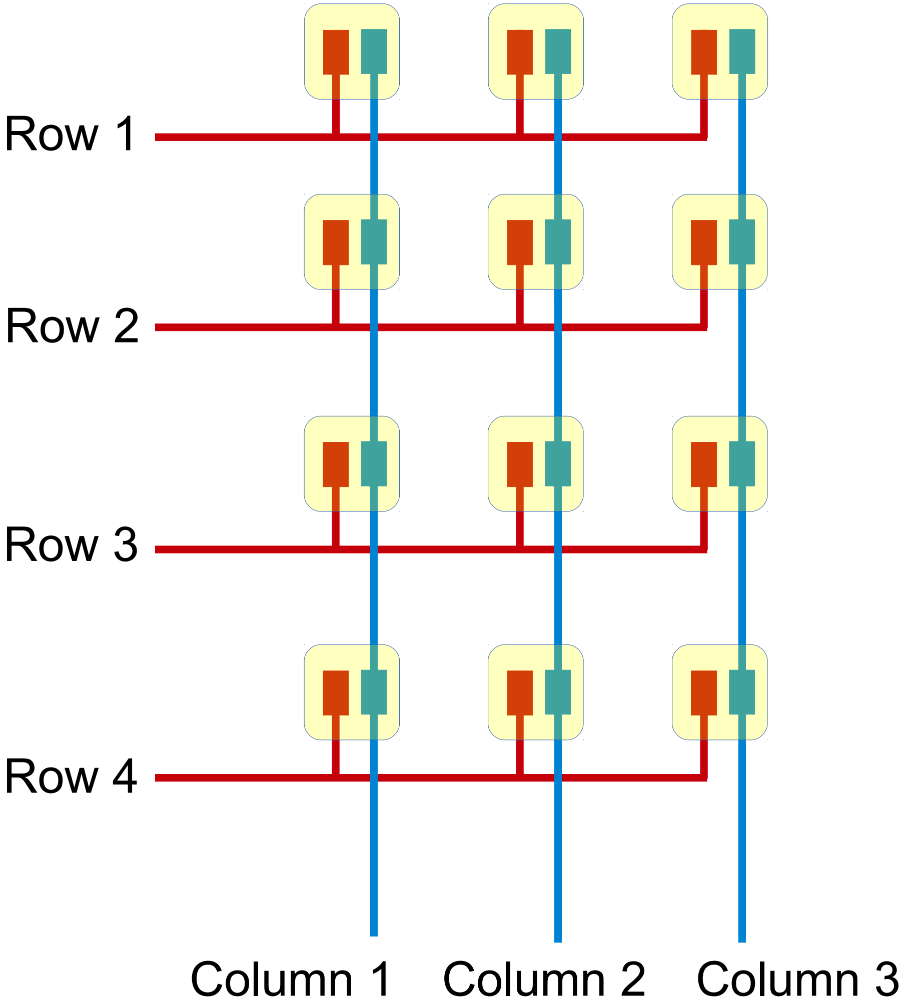
- Matrix keypads contain a set of buttons
- Each button has two terminals
- Without a matrix arrangement the keypad would require
  - 2xN wires where N = number of buttons
  - OR
  - N+1 wires if one terminal on each button share a common pin
- Matrix arrangement greatly reduces number of wires required

Interfacing a matrix keypad

Matrix keypad

Membrane keypad

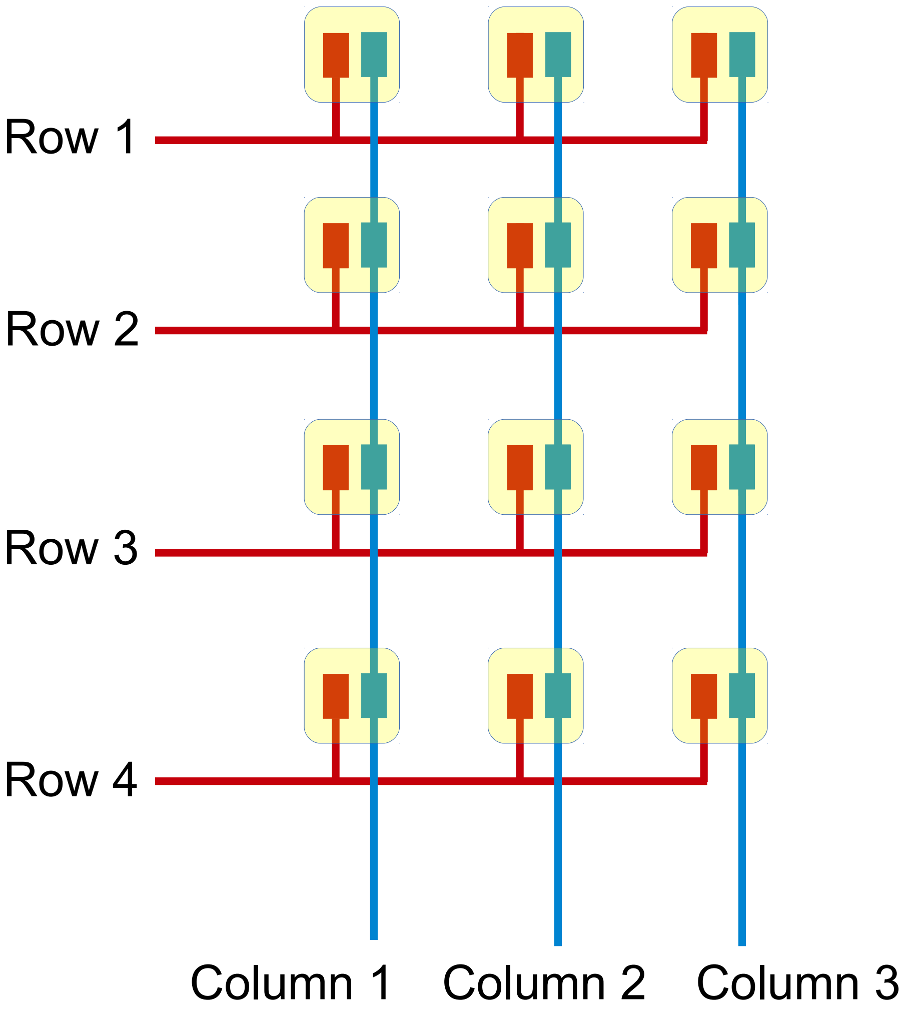
Interfacing a matrix keypad



Operation:

Conductive buttons create connections between rows and columns when pressed

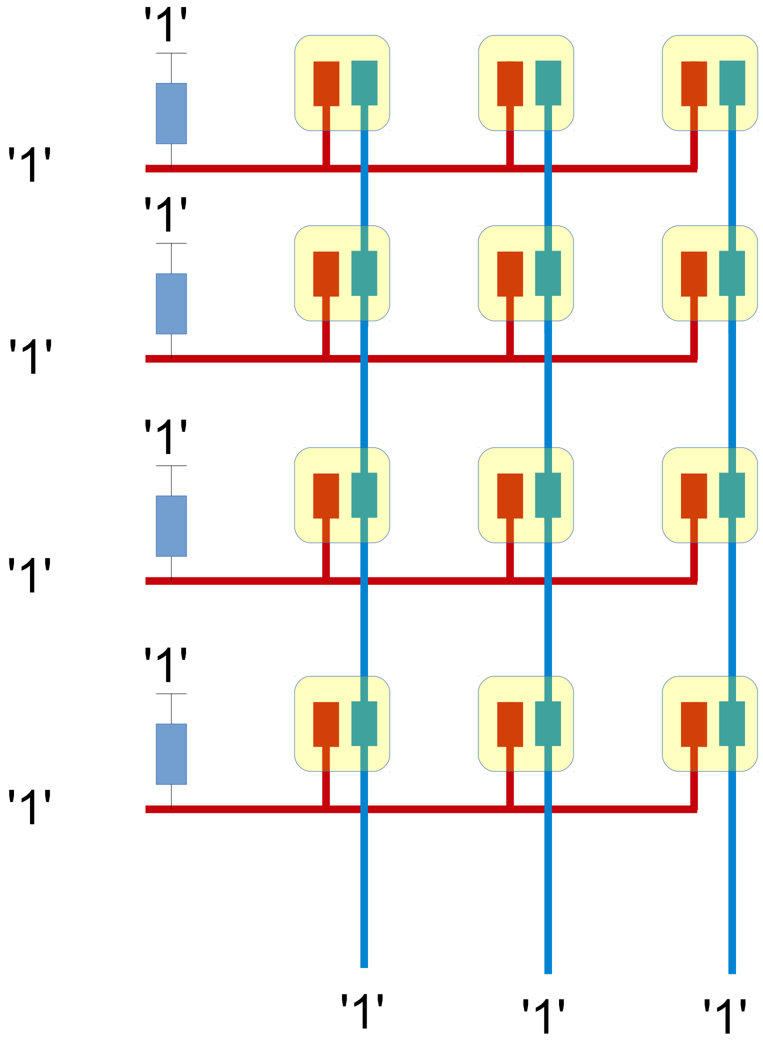
Interfacing a matrix keypad



Operation:

Microcontroller has to scan each row (or column) to check if a button has been pressed

Interfacing a matrix keypad



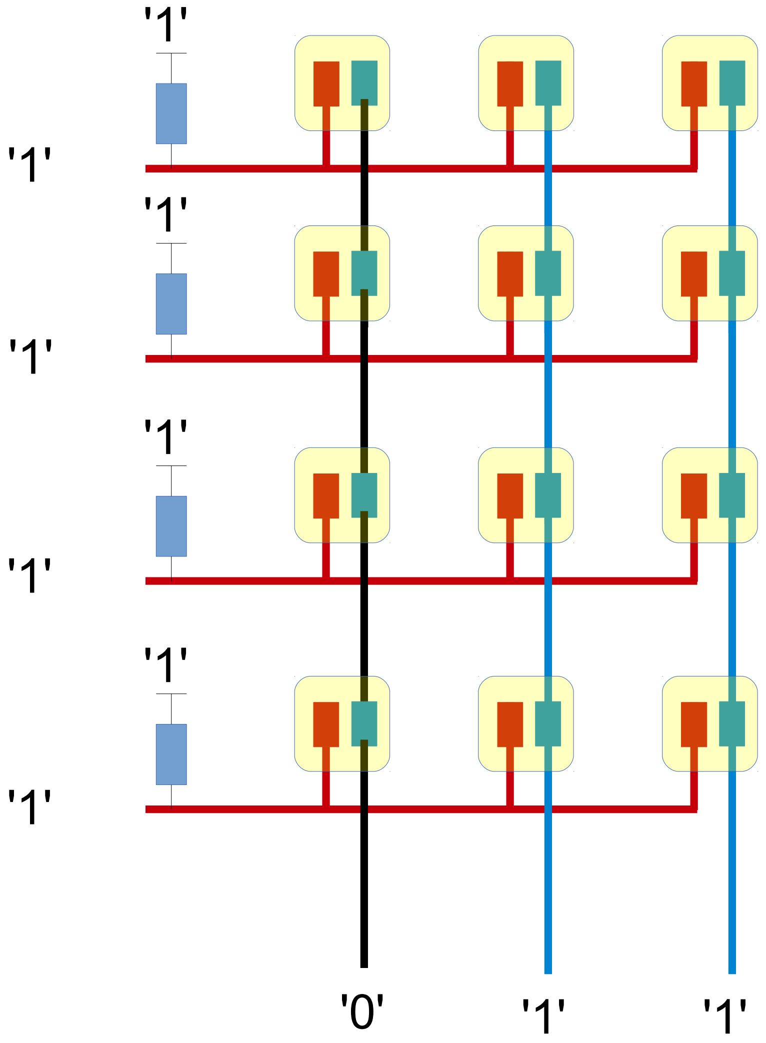
Operation:

No button pressed

Columns are driven to logic 1 by microcontroller.

Row outputs read '1' due to pull-ups.

Interfacing a matrix keypad



Operation:

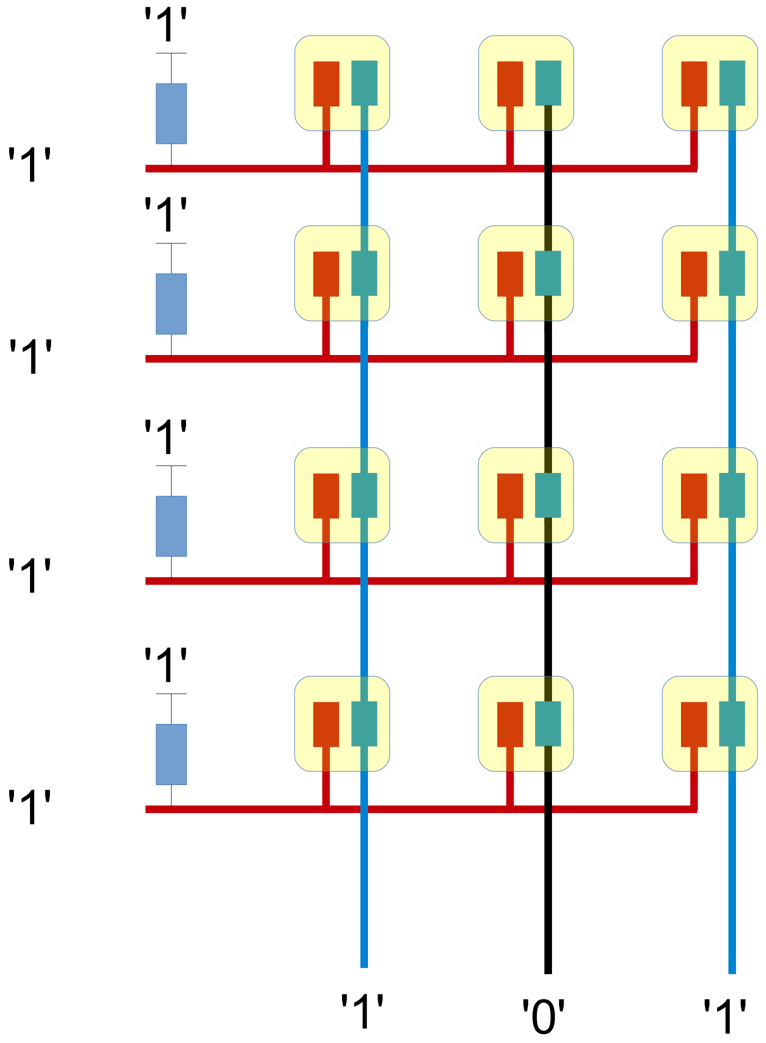
No button pressed

Microcontroller scans first Column by driving it to logic 0.

All Rows read 1.

Conclusion: no button pressed on Column 1

Interfacing a matrix keypad



Operation:

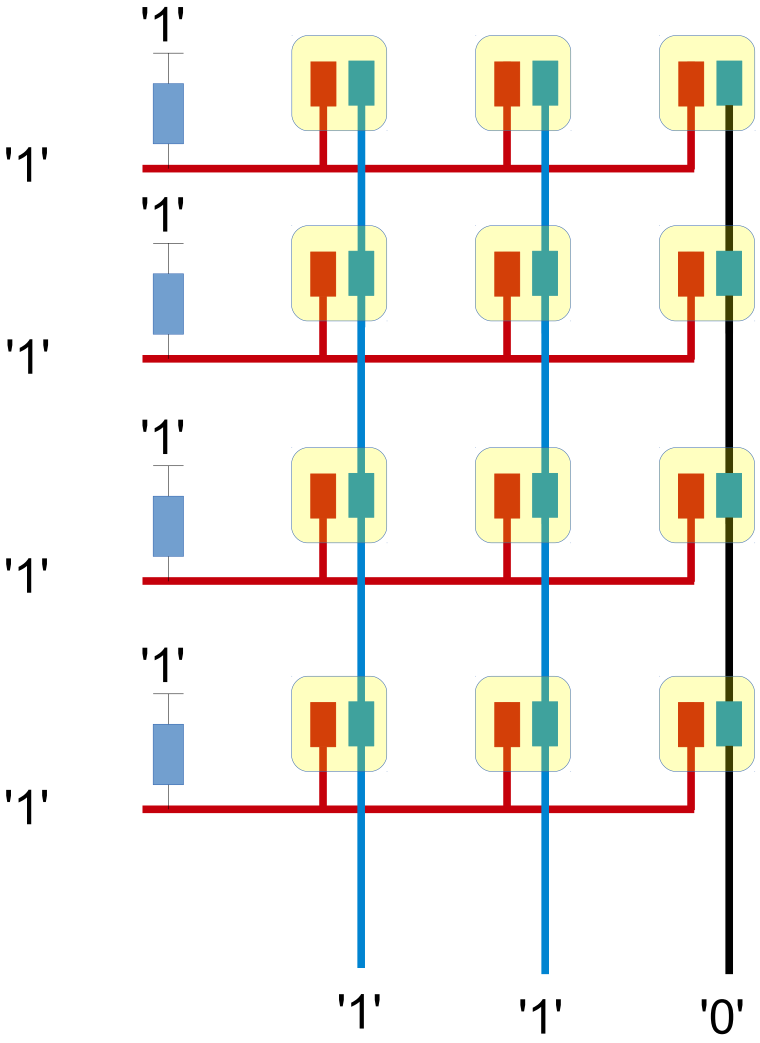
No button pressed

Microcontroller scans second Column by driving it to logic 0.

All Rows read 1.

Conclusion: no button pressed on Column 2

Interfacing a matrix keypad



Operation:

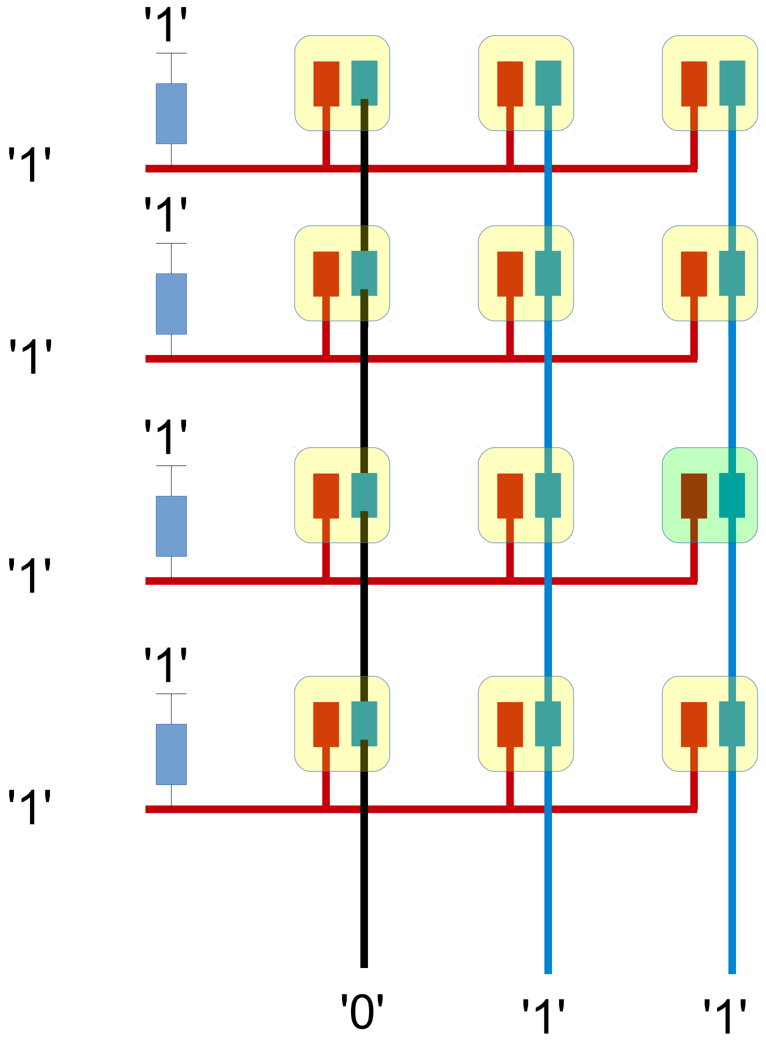
No button pressed

Microcontroller scans third Column by driving it to logic 0.

All Rows read 1.

Conclusion: no button pressed on Column 3

Interfacing a matrix keypad



Operation:

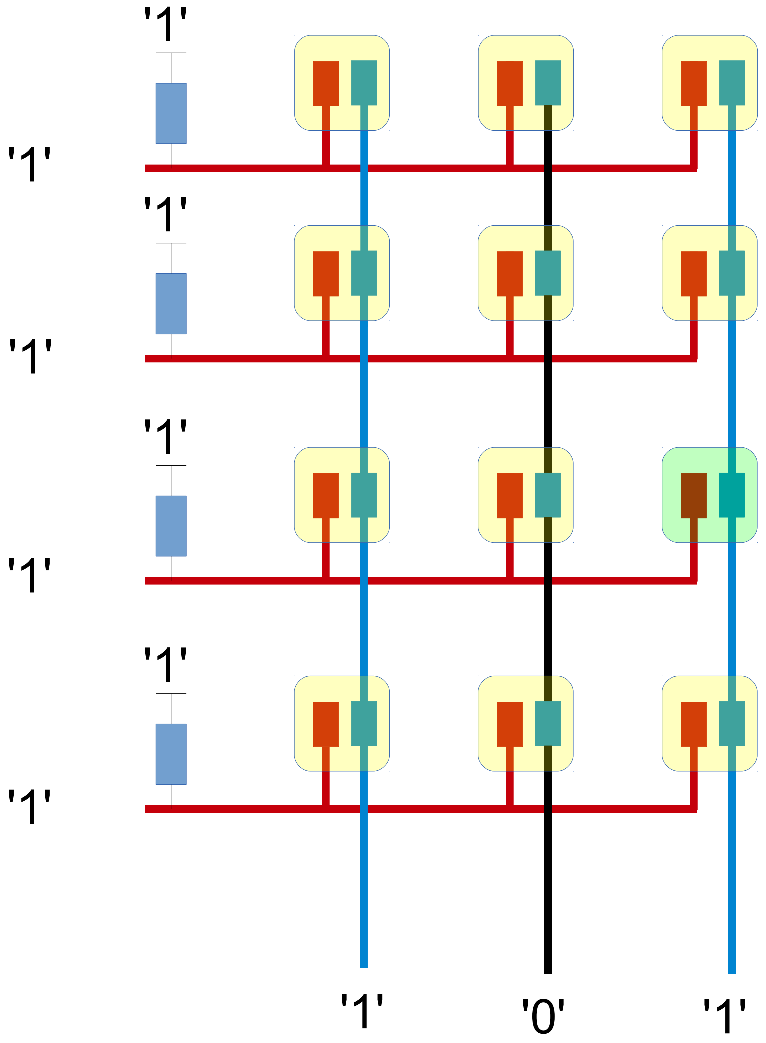
Green button pressed

Microcontroller scans first Column by driving it to logic 0.

All rows read 1.

Conclusion: no button pressed on Column 1

Interfacing a matrix keypad



Operation:

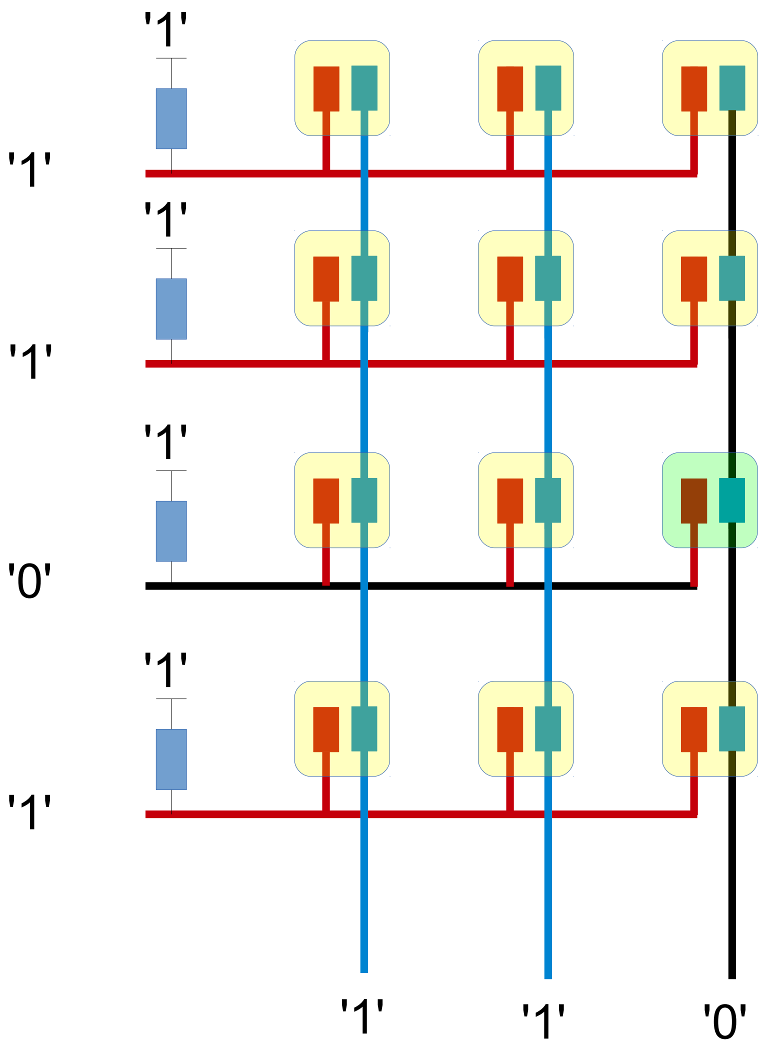
Green button pressed

Microcontroller scans second column by driving it to logic 0.

All rows read 1.

Conclusion: no button pressed on Column 2

Interfacing a matrix keypad



Operation:

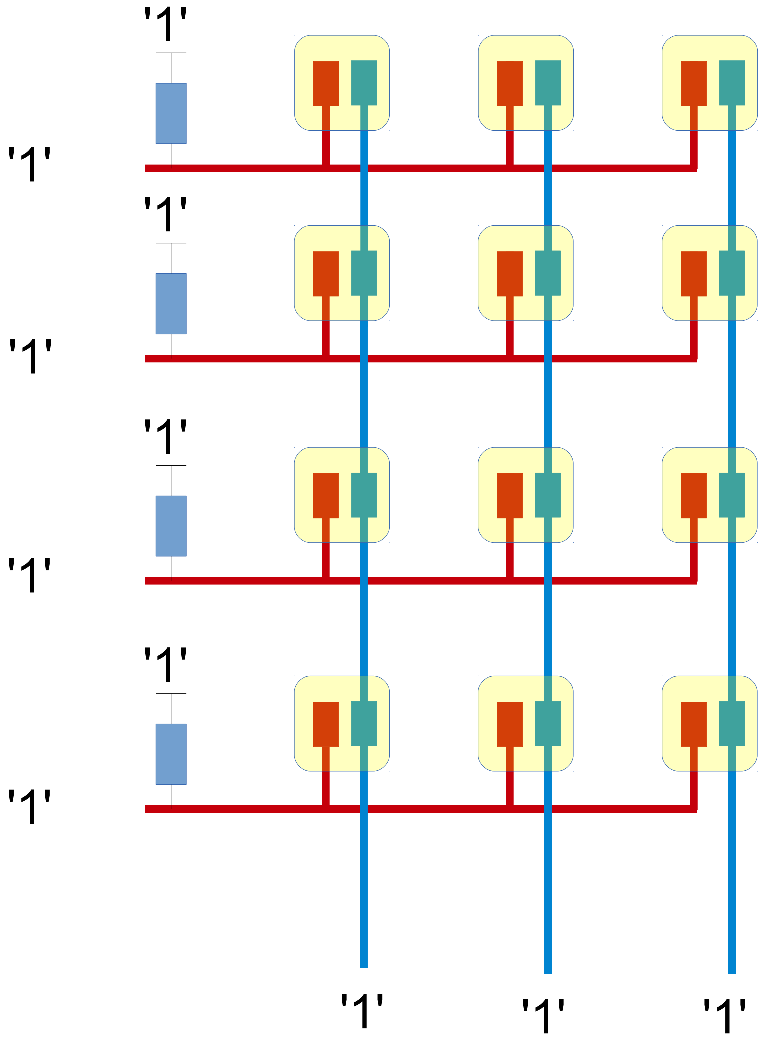
Green button pressed

Microcontroller scans third Column by driving it to logic 0.

Row 3 reads '0'

Conclusion: Button at intersection of Row 3 and Column 2 is pressed

Interfacing a matrix keypad



Operation:

Rows should be connected to input port pins

Columns should be connected to output port pins

Interfacing a matrix keypad

Outline of code:

Initialization code

    Configure port pins as inputs and outputs as appropriate
    If there are internal "pull-up" (quite common) enable them

Scan code.

    Drive Col 0 low
    Read column inputs
    Is Row 1 zero?
    Is Row 2 zero?
    Is Row 3 zero?

Interfacing a matrix keypad

Outline of code:

Initialization code

```
    Configure port pins as inputs and outputs as appropriate
    If there are internal “pull-up” (quite common) enable them
```

Scan code.

```
    Drive Col 0 low
    Read column inputs
    Is Row 1 zero?
    Is Row 2 zero?
    Is Row 3 zero?
```

} Repeat for other rows

Interfacing a matrix keypad

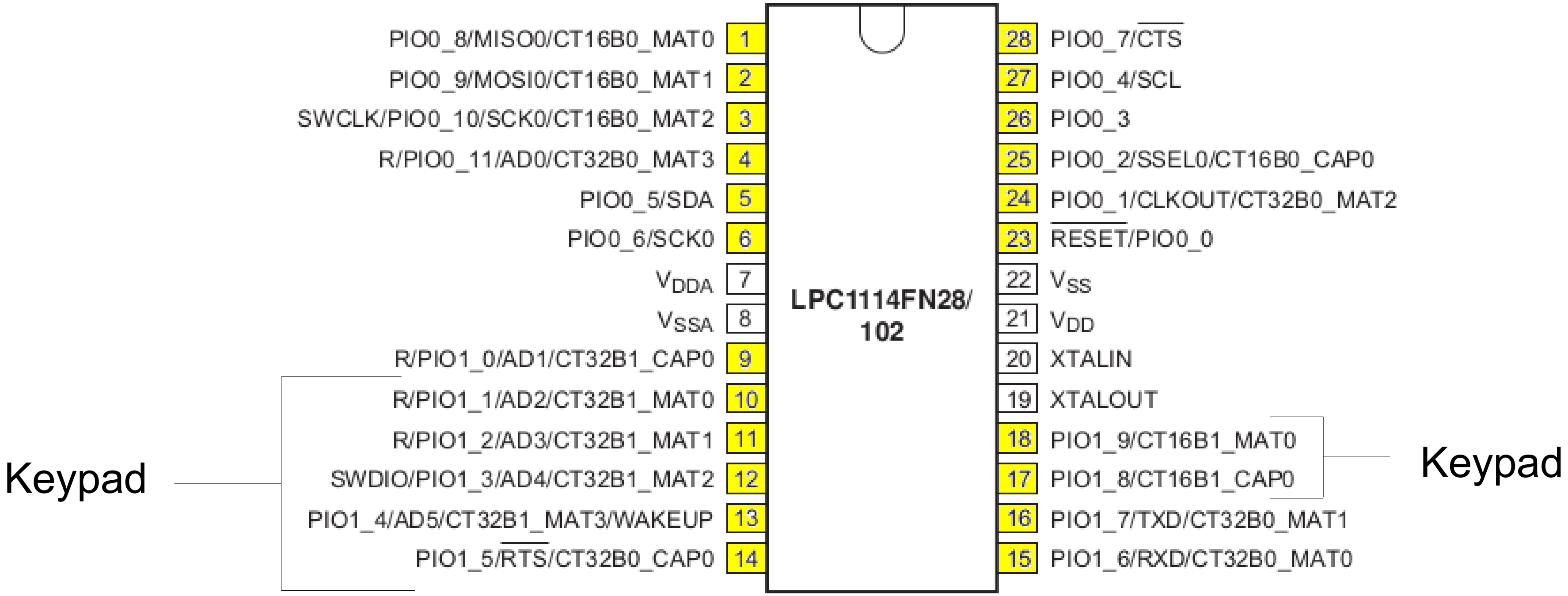
Outline of code:

```
Initialization code
    Configure port pins as inputs and outputs as appropriate
    If there are internal “pull-up” (quite common) enable them

Scan code.

    Drive Col 0 low
    Read column inputs
    Is Row 1 zero?
    Is Row 2 zero?
    Is Row 3 zero?
```

Repeat for other columns

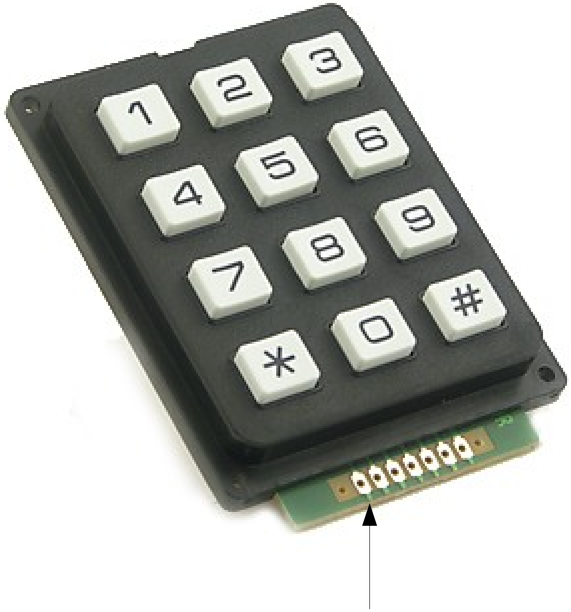
Return scan code representing button – may or may not be ASCII.

Interfacing a matrix keypad



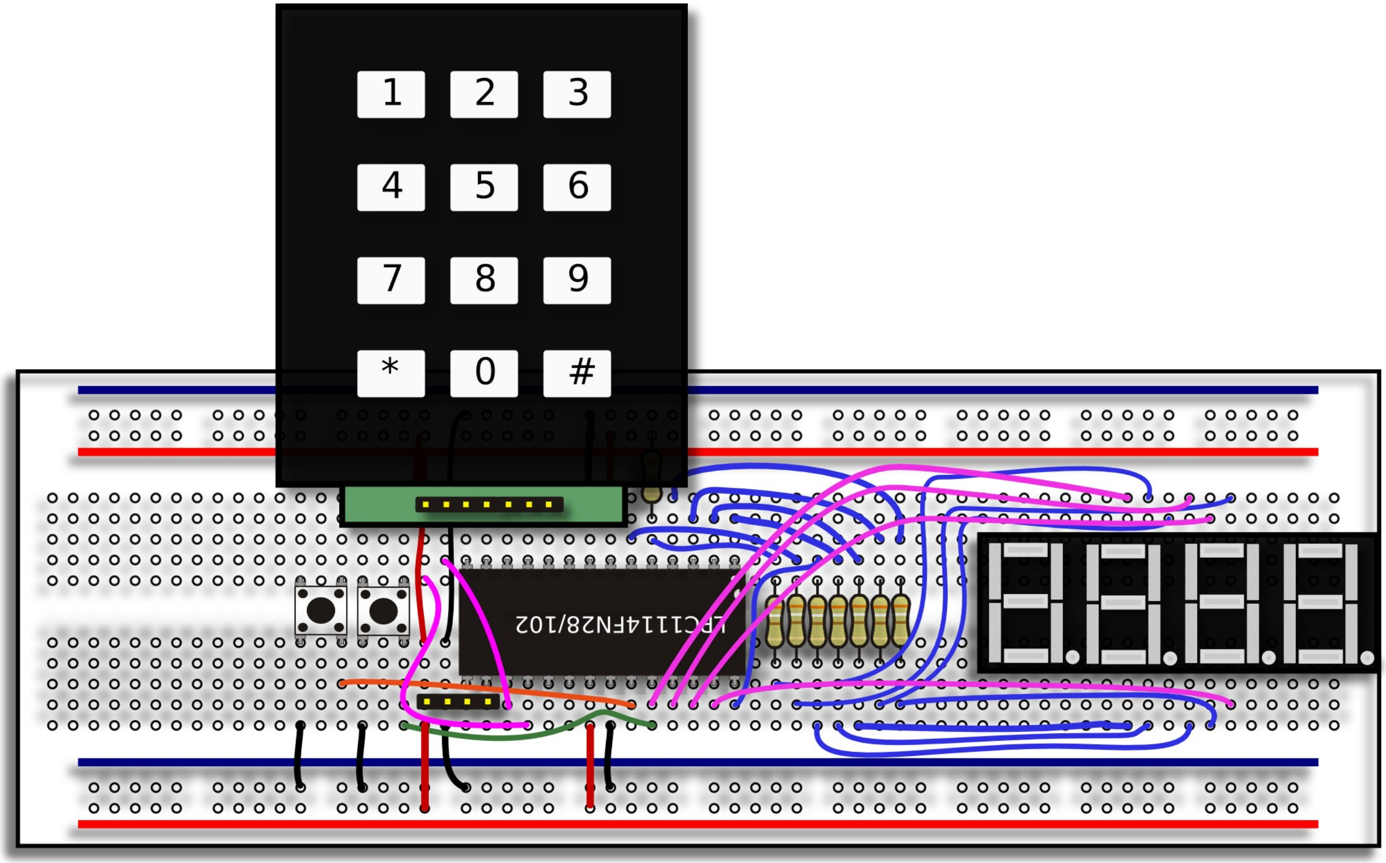
Interfacing a matrix keypad

The keypad you will use has the following pin arrangement (pin 1 is the one nearest the * key)



Keypad	Output Pins
1	2-3
2	2-1
3	2-5
4	7-3
5	7-1
6	7-5
7	6-3
8	6-1
9	6-5
0	4-1
*	4-3
#	4-5

Interfacing a matrix keypad



Interfacing a matrix keypad

Define symbols representing rows and columns.
If the wiring changes you just need to redefine the constants.

```
// Keypad is on Port 1
#define COL_1  BIT5
#define COL_2  BIT9
#define COL_3  BIT3
#define ROW_1  BIT8
#define ROW_2  BIT1
#define ROW_3  BIT2
#define ROW_4  BIT4
```

Interfacing a matrix keypad

Define symbols representing rows and columns.
If the wiring changes you just need to redefine the constants.

```
// select gpio mode with pull-ups for keypad pins
  IOCON_R_PIO1_0 |= 1+BIT4;
  IOCON_R_PIO1_1 |= 1+BIT4;
  IOCON_R_PIO1_2 |= 1+BIT4;
  IOCON_SWDIO_PIO1_3 |= 1+BIT4;

  GPIO1DATA = 0xfff;  // drive all cols high
  // Make column bits outputs
  GPIO1DIR |= COL_1 | COL_2 | COL_3;
  // Make row bits inputs
  GPIO1DIR &= ~(ROW_1 | ROW_2 | ROW_3);
```

7.4.29 IOCON_R_PIO1_0

Table 85. IOCON_R_PIO1_0 register (IOCON_R_PIO1_0, address 0x4004 4078) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Selects function R. This function is reserved. Select one of the alternate functions below.	
		0x1	Selects function PIO1_0.	
		0x2	Selects function AD1.	
		0x3	Selects function CT32B1_CAP0.	
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control).	10
		0x0	Inactive (no pull-down/pull-up resistor enabled).	
		0x1	Pull-down resistor enabled.	
		0x2	Pull-up resistor enabled.	
		0x3	Repeater mode.	
5	HYS		Hysteresis.	0
		0	Disable.	
		1	Enable.	
6	-	-	Reserved	1
7	ADMODE		Selects Analog/Digital mode	1
		0	Analog input mode	
		1	Digital functional mode	
9:8	-	-	Reserved	00
10	OD		Selects pseudo open-drain mode. See Section 7.1 for part specific details.	0
		0	Standard GPIO output	
		1	Open-drain output	
31:11	-	-	Reserved	-

7.4.30 IOCON_R_PIO1_1

Table 86. IOCON_R_PIO1_1 register (IOCON_R_PIO1_1, address 0x4004 407C) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Selects function R. This function is reserved. Select one of the alternate functions below.	
		0x1	Selects function PIO1_1.	
		0x2	Selects function AD2.	
		0x3	Selects function CT32B1_MAT0.	

7.4.31 IOCON_R_PIO1_2

Table 87. IOCON_R_PIO1_2 register (IOCON_R_PIO1_2, address 0x4004 4080) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. All other values are reserved.	000
		0x0	Selects function R. This function is reserved. Select one of the alternate functions below.	
		0x1	Selects function PIO1_2.	
		0x2	Selects function AD3.	
		0x3	Selects function CT32B1_MAT1.	
4:3	MODE		Selects function mode (on-chip pull-up/pull-down resistor control).	10
		0x0	Inactive (no pull-down/pull-up resistor enabled).	
		0x1	Pull-down resistor enabled.	
		0x2	Pull-up resistor enabled.	
		0x3	Repeater mode.	
5	HYS		Hysteresis.	0
		0	Disable.	
		1	Enable.	
6	-	-	Reserved	1
7	ADMODE		Selects Analog/Digital mode	1
		0	Analog input mode	
		1	Digital functional mode	
9:8	-	-	Reserved	00
10	OD		Selects pseudo open-drain mode. See Section 7.1 for part specific details.	0
		0	Standard GPIO output	
		1	Open-drain output	
31:11	-	-	Reserved	-

Interfacing a matrix keypad

Define symbols representing rows and columns.
If the wiring changes you just need to redefine the constants.

```
char ScanKeys()
{
   GPIO1DATA |= COL_1 | COL_2 | COL_3;
   GPIO1DATA &= ~COL_1;
   if ((GPIO1DATA & ROW_1) == 0)
      return '1';
   if ((GPIO1DATA & ROW_2) == 0)
      return '4';
   if ((GPIO1DATA & ROW_3) == 0)
      return '7';
   if ((GPIO1DATA & ROW_4) == 0)
      return '*';
```

Interfacing a matrix keypad

Define symbols representing rows and columns.
If the wiring changes you just need to redefine the constants.

```
  GPIO1DATA |= COL_1 | COL_2 | COL_3;
  GPIO1DATA &= ~COL_2;
  if ((GPIO1DATA & ROW_1) == 0)
     return '2';
  if ((GPIO1DATA & ROW_2) == 0)
     return '5';
  if ((GPIO1DATA & ROW_3) == 0)
     return '8';
  if ((GPIO1DATA & ROW_4) == 0)
     return '0';
  Return 0;  // no key
}
```