Micro Processor & Controller

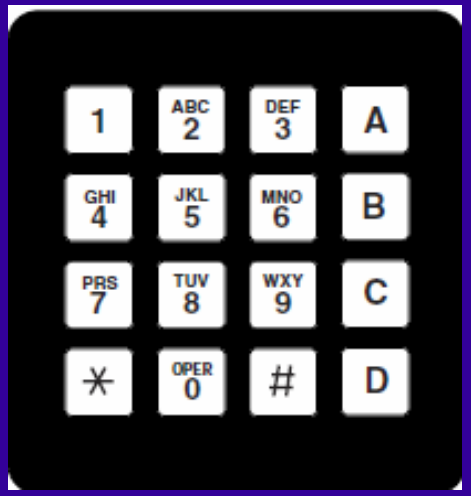
Matrix Keyboard & Ext Interrupt

Delfino EVB External Bus

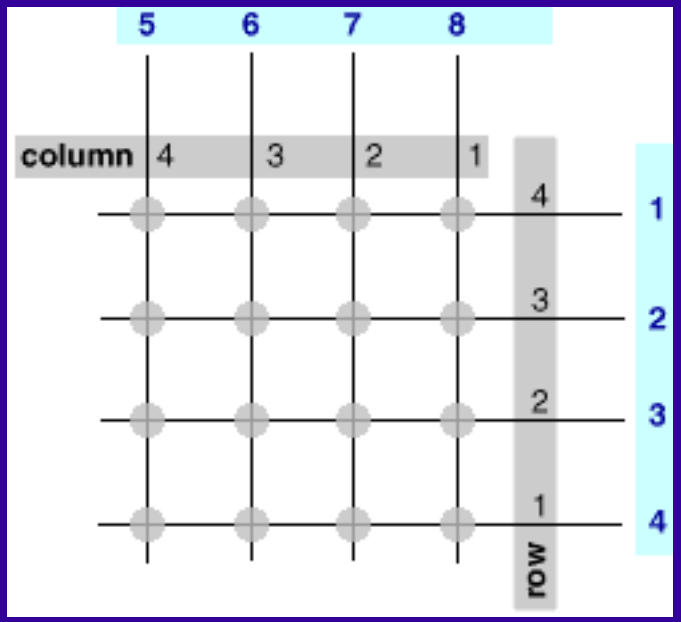
The keyboard connects to the Delfino EVB by External Bus using GPIO40-GPIO47.

Matrix Keyboard Architecture

- Our keyboard is a passive device.
- The architecture is a 4x4 matrix of 4 column and 4 row.
- The keyboard has a bouncing effect.

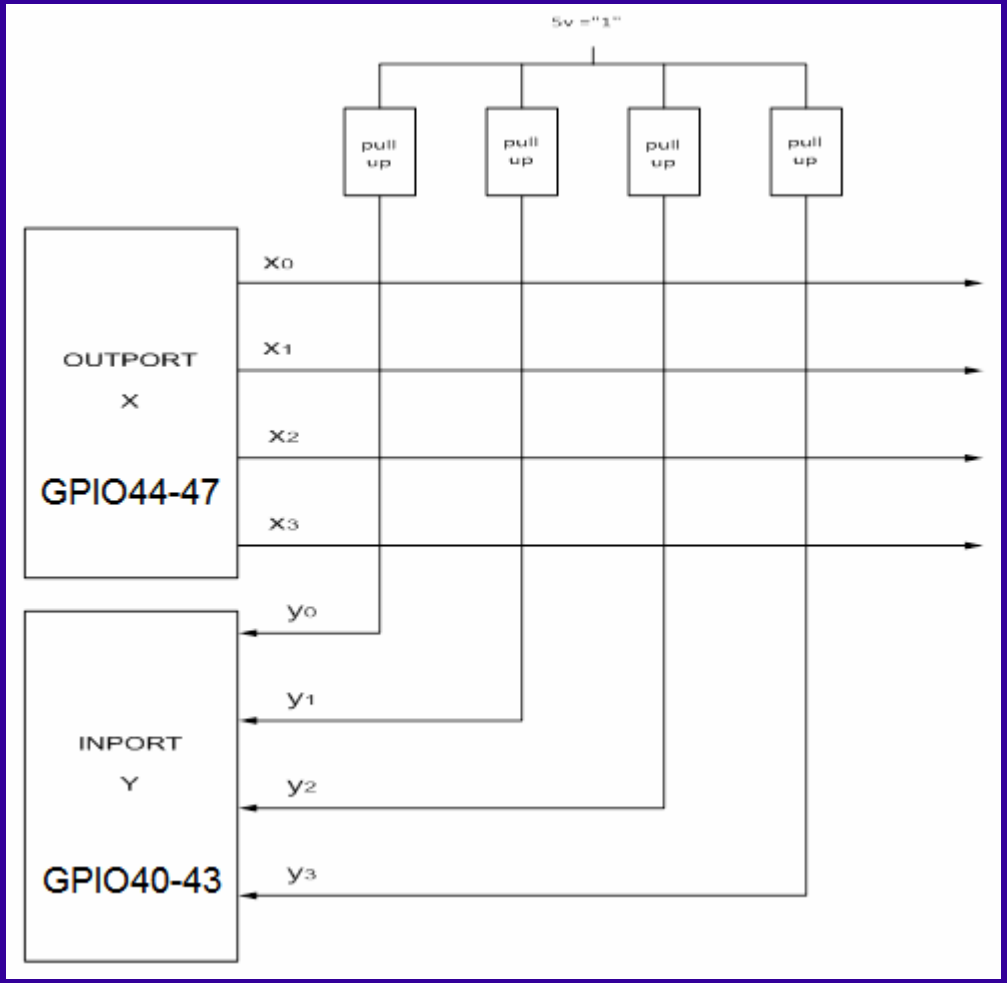


Matrix Keyboard Architecture

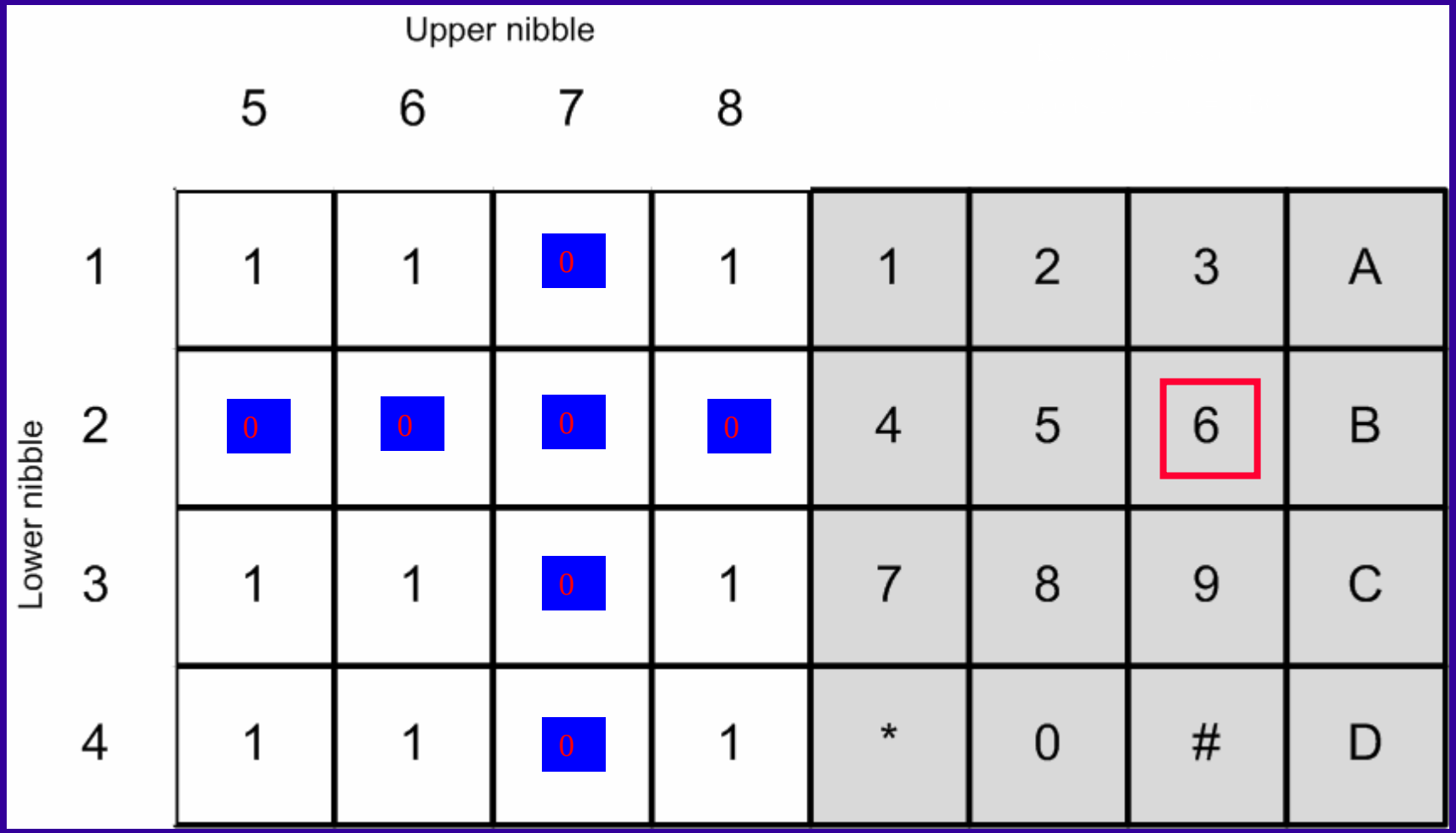


- 4 lines are GPIO outputs.
- 4 lines are GPIO inputs using Pull-up resistors.
- Pressing a button causes shortening between the relevant row and a column.
- The decoding is done by the correlation between the row & column.
- One can use Polling or Interrupt to read the keyboard.

Matrix Keyboard Connection

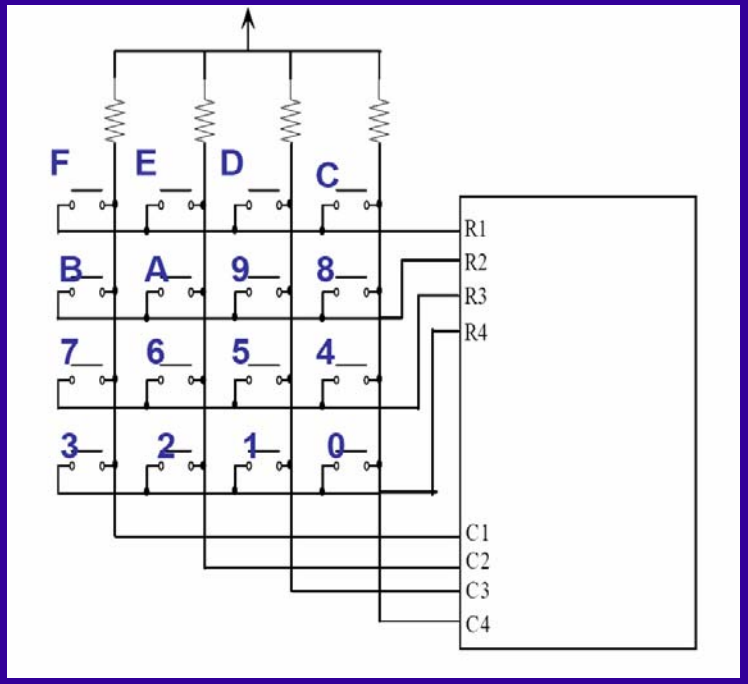


Scanning Method



Scanning Procedure

- Place 0 on R_K bit.
- Wait on end of bouncing.
- Read C port (MyNib).
- If (MyNib != 0xF) then the button has been pressed.
- Build key-code.
- Otherwise, try next row.
- Repeat constantly.

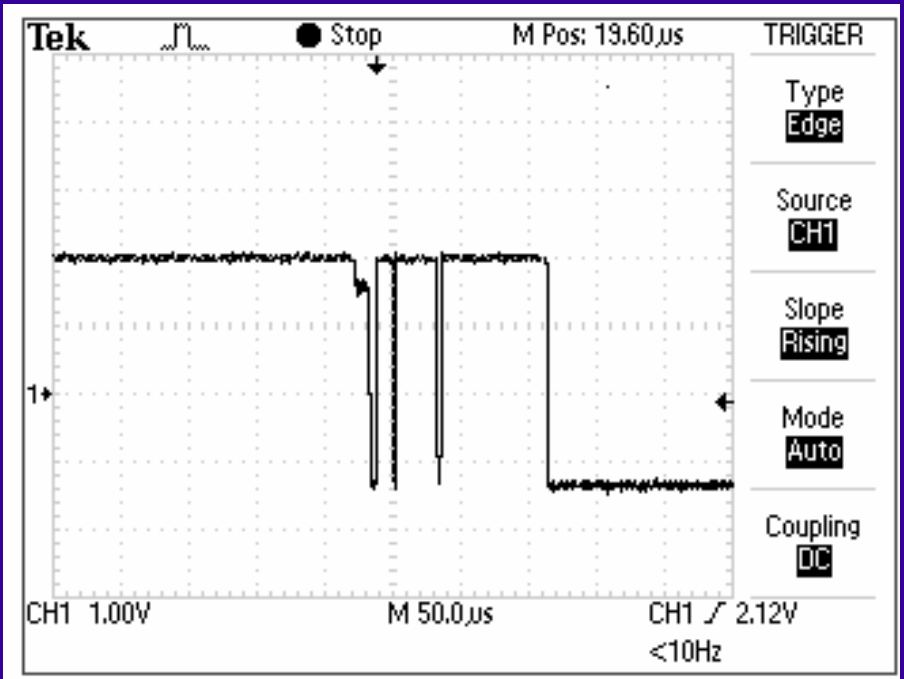


Contact Bouncing Effect

When a switch is actuated and contacts touch one another under the force of actuation, they are supposed to establish continuity in a single, crisp moment. Unfortunately, though, switches do not exactly achieve this goal. Due to the mass of the moving contact and any elasticity inherent in the mechanism and/or contact materials, contacts will “bounce” upon closure for a period of milliseconds before coming to a full rest and providing unbroken contact.

if the switch is used to send a signal to an electronic amplifier or some other circuit with a fast response time, contact bounce may produce very noticeable and undesired effects

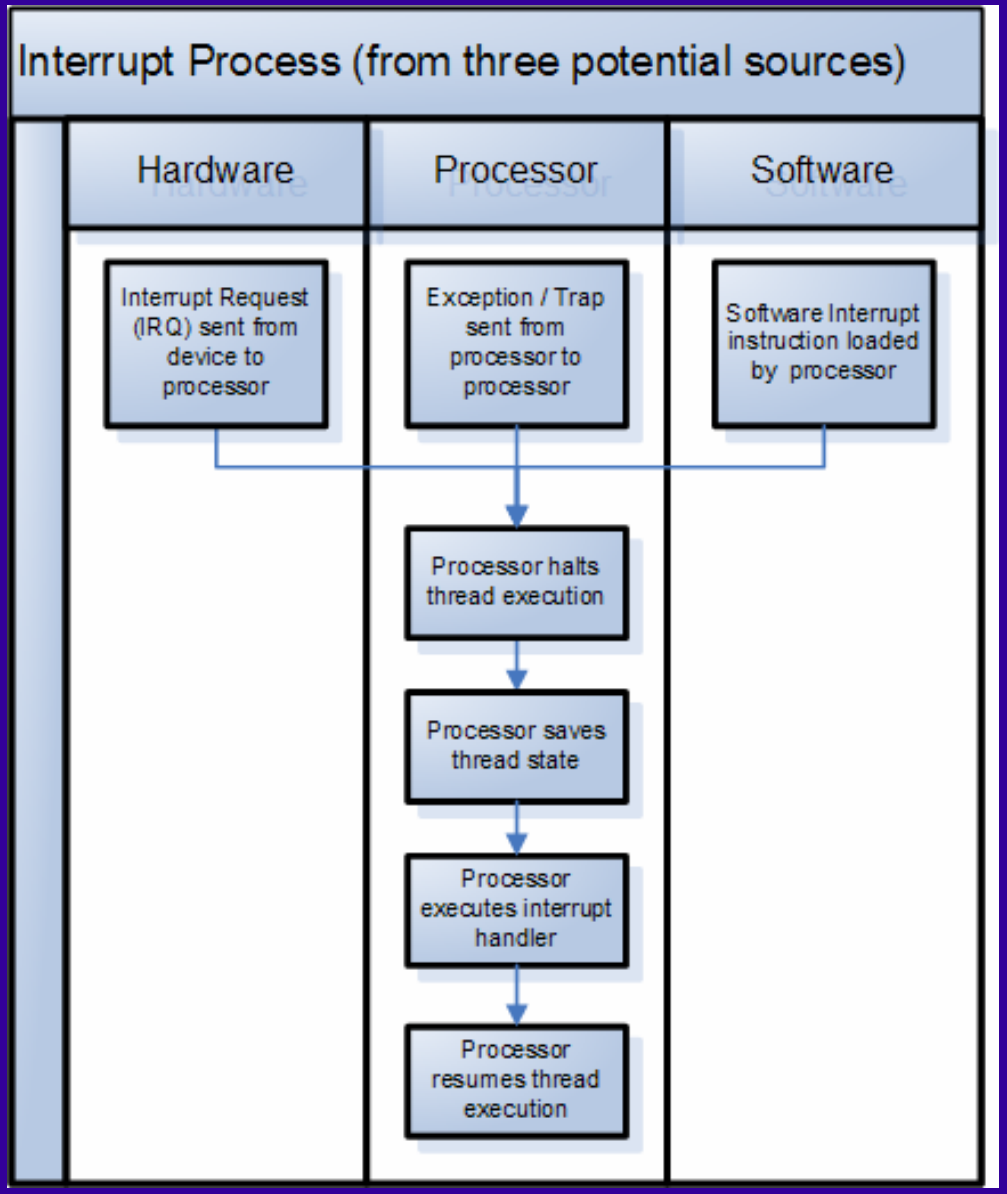
Contact Bouncing Effect



Switch bounce produced on switch press

- Effect that we find in mechanical switching devices.
- At switching time there has a bouncing in order of milliseconds.
- One must use an analog or digital filter to eliminate the effect.

Interrupt - Source



Scanning Code

```
char ReadKB(char wait)
{
static char code[] = {0xE, 0xD, 0xB, 0x7};
char data;
char i;
  KeboardWriteCode(0x0);
  DELAY_US(1000);
  while(KeboardReadCode() == 0x0F)  // Check 4 data bits GPIO44-GPIO47
    if (!wait) return(0);
  Beep(20);
  for(i=0; i<4; i++)
    {
    KeboardWriteCode(code[i]);
    DELAY_US(1000);
    data = KeboardReadCode();
    if (data != 0x0F)
        break;
    }
  while(KeboardReadCode() != 0x0F); // Wait for button release
  DELAY_US(1000);
  KeboardWriteCode(0x0);
  return(scan2ascii((data << 4) | i));
}
```