ECE 372 – Microcontroller Design
Parallel IO Ports - Inputs

**Button Inputs - Bounce Problem**
- We want to recognize this as a single press of the switch rather than multiple presses as indicated in the voltage diagram

**Solutions:**
- Some expensive switches do not bounce
- We can debounce switches with hardware
- We can debounce switches with software

---

**Hardware De-bouncing**
- Use hardware to de-bounce button presses
- Button/Switch bounce removed using capacitor and Schmitt Trigger

**Why is this a bad circuit?**
**ECE 372 – Microcontroller Design**  
**Parallel IO Ports - Inputs**

### Hardware De-bouncing
- Choose capacitor value large enough such that the inverter input does not exceed 0.7V threshold.

![Hardware De-bouncing Diagram](image)

*If this exceeds the input threshold, we will not have de-bounced the switch/button*

### Software De-bouncing
- Good hardware de-bounce implementation
  - Limit instantaneous current to avoid sparks

![Software De-bouncing Diagram](image)

### Hardware De-bouncing
- Resistance of contact is 0.1Ω
- Instantaneous current when switch closes is large enough to cause a spark
  - Will shorten life of your button

### Software De-bouncing
- De-bounce switch/button using software code on your microcontroller
  - No additional hardware required

### Steps:
- Wait for key to be pressed
  - Delay 10 ms (should be longer than expected bounce time)
- Wait for key to be released
  - Delay 10 ms
ECE 372 – Microcontroller Design
Parallel IO Ports - Inputs

Software De-bouncing
- Flowchart for software code

Software de-bounce for button press

Software de-bounce for button release

Software De-bouncing for Button Press - Assembly

Assembly Code:
waitPress
    idda PORTT ;wait for press
    anda #$01
    bne waitPress
    ldd DELAY ;wait 10ms
    loop
    decd
    bne loop
    rts

C Code:
void waitPress(void) {
    while (PORTT & 0x01); /*wait for press*/
    for(i=0; i<DELAY; i++); /* delay */
}

Software De-bouncing for Button Release - Assembly

Assembly Code:
waitRelease
    idda PORTT
    anda #$01
    beq waitRelease
    ldd DELAY
    loop
    decd
    bne loop
    rts

ECE 372 – Microcontroller Design
Parallel IO Ports - Inputs

Software De-bouncing for Button Press - Assembly

Software De-bouncing for Button Release - Assembly

ECE 372 – Microcontroller Design
Parallel IO Ports - Inputs
Software De-bouncing for Button Release - Assembly

C Code:
void waitforrelease(void) {
    while(!(PORTT & 0x01));
    for(i=0; i<DELAY; i++);
}

Software De-bouncing
- Timing is not very accurate
- How can we better code the subroutines?
- Use internal timer (TCNT)

Alternative simple software de-bouncing
- Read switch/button until you read the same value twice

Keypads
- Collection of several keys grouped into a single device
  - Can buy keypads in many different configurations
  - Build your own keypad using individual buttons

- How can we read dozens of keys on the keypad without requiring dozens on individual input ports on our microprocessor?
  - Hint: It's similar to how we can interface with dozens of outputs using only a few pins
Keypads

Keypad Scanning
- Similar to scanning for controlling multiple output LEDs
- Scan the keys one row at a time and read the column lines to see if any key on that row were pressed

1. Set each row output sequentially to output 0
2. Check column inputs to see if key was pressed

What if this key is pressed?
**Keypad Scanning**
- Similar to scanning for controlling multiple output LEDs
- Scan the keys one row at a time and read the column lines to see if any key on that row was pressed

1. Set each row output sequentially to output 0
2. Check column inputs to see if key was pressed

**Keypad Scanning**
- Similar to scanning for controlling multiple output LEDs
- Scan the keys one row at a time and read the column lines to see if any key on that row was pressed

1. Set each row output sequentially to output 0
2. Check column inputs to see if key was pressed

**Keypad Scanning**
- Similar to scanning for controlling multiple output LEDs
- Scan the keys one row at a time and read the column lines to see if any key on that row was pressed

1. Set each row output sequentially to output 0
2. Check column inputs to see if key was pressed

**Keypad Multiplexing**
- Could also use keypad multiplexing
  - Enable each row one at a time by outputting the row address to the decoder input
  - Decoder will enable only one row
  - Use encoder to read key press and output column address of keypress to microcontroller

**Keypad Multiplexing**
- Could also use keypad multiplexing
  - Enable each row one at a time by outputting the row address to the decoder input
  - Decoder will enable only one row
  - Use encoder to read key press and output column address of keypress to microcontroller

**Technical Diagram**
- Keypad Multiplexing – Challenges
  - Can one handle one key press at a time
  - How do we recognize the when no keys are pressed?

- Need an encoder to provide the column address of the key that was pressed
- Can use the output E0 of the following encoder to detect if no key are pressed

<table>
<thead>
<tr>
<th>INPUTS</th>
<th>OUTPUTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>A0</td>
<td>A1</td>
</tr>
<tr>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>L</td>
<td>H</td>
</tr>
</tbody>
</table>

* Note: Not used