The Human and Physical Interface

Chapter 8
Sections 1 - 9

Dr. Iyad Jafar
Outline

- Introduction
- From Switches to Keypads
- LED Displays
- Simple Sensors
- Actuators
- Summary
Introduction

- Humans need to interface with embedded systems; input data and see response
- Input devices: switches, pushbuttons, keypads, sensors
- Output devices: LEDs, seven-segment displays, liquid crystal displays, motors, actuators
Introduction

- Examples

Fridge Control Panel

Photocopier Control Panel
Introduction

- Examples

Car Dashboard
Moving From Switches to Keypads

- Switches are good for conveying information of digital nature.
- They can be used in multiples; each connected to one port pin.
- In complex systems, it might not be feasible to keep adding switches?!
- Use keypads!
  - Can be used to convey alphanumerical values.
  - A group of switches arranged in matrix form.
Moving From Switches to Keypads

Internal Structure of Keypad

- Pull-up resistors
- Port bit
- Row connections
- Column connections

Keypad connections:
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- *
- 0
- #
Moving From Switches to Keypads

How to Determine the Pressed Key

1. Set column bits as outputs
2. Set row bits as inputs
3. Set column bits to 0
4. Read row bits
5. Set column bits as inputs
6. Set row bits as outputs
7. Set row bits to 0
8. Read column bits

<table>
<thead>
<tr>
<th>Key</th>
<th>Value Read</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0111 011X</td>
</tr>
<tr>
<td>2</td>
<td>0111 101X</td>
</tr>
<tr>
<td>3</td>
<td>0111 110X</td>
</tr>
<tr>
<td>4</td>
<td>1011 011X</td>
</tr>
<tr>
<td>5</td>
<td>1011 101X</td>
</tr>
<tr>
<td>6</td>
<td>1011 110X</td>
</tr>
<tr>
<td>7</td>
<td>1101 011X</td>
</tr>
<tr>
<td>8</td>
<td>1101 101X</td>
</tr>
<tr>
<td>9</td>
<td>1101 110X</td>
</tr>
<tr>
<td>*</td>
<td>1110 011X</td>
</tr>
<tr>
<td>0</td>
<td>1110 101X</td>
</tr>
<tr>
<td>#</td>
<td>1110 110X</td>
</tr>
</tbody>
</table>
Moving From Switches to Keypads
Using Keypad in a Microcontroller

1. Initialise
2. Enable interrupt
3. Wait
4. Interrupt
5. Read keypad pattern
6. Convert pattern to ASCII
7. Output to LCD
8. Is key released?
   - No
   - Yes
      - Clear interrupt flag
      - Return from interrupt (RETFIE)
Moving From Switches to Keypads

Example
A program to read an input from a 4x3 keypad and displays the equivalent decimal number on 4 LEDs. If the pressed key is not a number, then the LEDs should be all on.

- The keypad will be connected to MC as follows
  - Rows 0 to 3 connected to RB7 to RB4 respectively.
  - Columns 0 to 2 connected to RB3 to RB1
- Use port B change interrupt
- connect the LEDs to RA0-RA3
- based on the pressed key, convert the row and column values to binary using a lookup table
Keypad Interfacing Example

```c
#include P16F84A.INC

ROW_INDEX EQU 0X20
COL_INDEX EQU 0X21

ORG 0X0000
GOTO START

START BSF STATUS, R0
     MOVLW B'11110000'
     MOVWF TRISB ; SET RB1-RB3 AS OUTPUT AND
                    ; RB4-RB7 AS INPUT
     MOVLW B'00000000'
     MOVWF TRISA ; SET RA0-RA3 AS OUTPUT
     BCF STATUS, R0
     CLRF PORTB ; INITIALIZE PORTB TO ZERO
     MOVF PORTB,W ; CLEAR RBIF FLAG
     BCF INTCON, RBIF
     BSF INTCON, RBIE
     BSF INTCON, GIE ; ENABLE PORT b CHANGE INTERRUPT
     GOTO LOOP ; WAIT FOR PRESSED KEY

LOOP BSF STATUS, R0
```

```c
     MOVLW B'11110000'
     MOVWF TRISB
     MOVLW B'00000000'
     MOVWF TRISA
     BCF STATUS, R0
     CLRF PORTB
     MOVF PORTB,W
     BCF INTCON, RBIF
     BSF INTCON, RBIE
     BSF INTCON, GIE
     GOTO LOOP
```

Keypad Interfacing Example

ISR

- MOVF PORTB, W ; READ ROW NUMBER
- MOVWF ROW_INDEX
- BSF STATUS, RP0 ; READ COLUMN NUMBER
- MOVLW B’00001110’
- MOVWF TRISB
- BCF STATUS, RP0
- CLRF PORTB
- MOVF PORTB, W
- MOVWF COL_INDEX
- CALL CONVERT ; CONVER THE ROW AND COLUMN
- RST_PB_DIRC
- BSF STATUS, RP0 ; PUT THE PORT BACK TO INITIAL SETTINGS
- MOVLW B’11110000’
- MOVWF TRISB ; SET RB1-RB3 AS OUTPUT AND
- MOVLW B’00000000’ ; RB4-RB7 AS INPUT
- MOVWF TRISA ; SET RA0-RA3 AS OUTPUT
- BCF STATUS, RP0
- CLRF PORTB
- MOVF PORTB, W ; REQUIRED TO CLEAR RBIF FLAG
- BCF INTCON, RBIF
- RETFIE
## Keypad Interfacing Example

**CONVERT**

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BTFSS COL_INDEX,3</td>
<td>IF 1(^{\text{ST}}) COLUMN, COL_INDEX=0</td>
</tr>
<tr>
<td>MOVLW 0</td>
<td></td>
</tr>
<tr>
<td>BTFSS COL_INDEX,2</td>
<td>IF 2(^{\text{ND}}) COLUMN, COL_INDEX=1</td>
</tr>
<tr>
<td>MOVLW 1</td>
<td></td>
</tr>
<tr>
<td>BTFSS COL_INDEX,1</td>
<td>IF 3(^{\text{RD}}) COLUMN, COL_INDEX=2</td>
</tr>
<tr>
<td>MOVLW 2</td>
<td></td>
</tr>
<tr>
<td>MOVWF COL_INDEX</td>
<td>STORE THE COLUMN INDEX</td>
</tr>
</tbody>
</table>

**FIND_ROW**

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BTFSS ROW_INDEX,7</td>
<td>IF 1(^{\text{ST}}) ROW, ROW_INDEX=0</td>
</tr>
<tr>
<td>MOVLW 0</td>
<td></td>
</tr>
<tr>
<td>BTFSS ROW_INDEX,6</td>
<td>IF 2(^{\text{ND}}) ROW, ROW_INDEX=1</td>
</tr>
<tr>
<td>MOVLW 1</td>
<td></td>
</tr>
<tr>
<td>BTFSS ROW_INDEX,5</td>
<td>IF 3(^{\text{RD}}) ROW, ROW_INDEX=2</td>
</tr>
<tr>
<td>MOVLW 2</td>
<td></td>
</tr>
<tr>
<td>BTFSS ROW_INDEX,4</td>
<td>IF 4(^{\text{TH}}) ROW, ROW_INDEX=3</td>
</tr>
<tr>
<td>MOVLW 3</td>
<td></td>
</tr>
<tr>
<td>MOVWF ROW_INDEX</td>
<td></td>
</tr>
</tbody>
</table>

; CONTINUED ON NEXT PAGE
Keypad Interfacing Example

```assembly
COMPUTE_VALUE    MOVF ROW_INDEX, W ; KEY # = ROW_INDEX*3 + COL_INDEX
ADDWF ROW_INDEX, W
ADDWF ROW_INDEX, W
ADDWF COL_INDEX, W ; THE VALUE IS IN W

; CHECK IF VALUE IS GREATER THAN 11. THIS HAPPENS WHEN THE BUTTON IS RELEASED
; LATER, AN INTERRUPT OCCURS WITH ALL SWITCHES OPEN, SO THE MAPPED VALUE IS
; ABOVE 11, THE LOOKUP TABLE
MOVWF 0X30
MOVLW 0X0C
SUBWF 0X30,W
BTFSC STATUS, C ; WILL NOT WORK CORRECTLY, OVERFLOW OCCURS
GOTO LL
CALL TABLE
MOVWF PORTA ; DISPLAY THE NUMBER ON PORTA
LL
RETURN
```
Keypad Interfacing Example

<table>
<thead>
<tr>
<th>TABLE</th>
<th>ADDWF</th>
<th>PCL, F</th>
</tr>
</thead>
<tbody>
<tr>
<td>RETLW</td>
<td>0x01</td>
<td></td>
</tr>
<tr>
<td>RETLW</td>
<td>0x02</td>
<td></td>
</tr>
<tr>
<td>RETLW</td>
<td>0x03</td>
<td></td>
</tr>
<tr>
<td>RETLW</td>
<td>0x04</td>
<td></td>
</tr>
<tr>
<td>RETLW</td>
<td>0x05</td>
<td></td>
</tr>
<tr>
<td>RETLW</td>
<td>0x06</td>
<td></td>
</tr>
<tr>
<td>RETLW</td>
<td>0x07</td>
<td></td>
</tr>
<tr>
<td>RETLW</td>
<td>0x08</td>
<td></td>
</tr>
<tr>
<td>RETLW</td>
<td>0x09</td>
<td></td>
</tr>
<tr>
<td>RETLW</td>
<td>0x0F</td>
<td>; ERROR CODE</td>
</tr>
<tr>
<td>RETLW</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>RETLW</td>
<td>0x0F</td>
<td>; ERROR CODE</td>
</tr>
</tbody>
</table>

END
LED Displays

- Light emitting diodes are simple and effective in conveying information

- However, in complex systems it becomes hard to deal with individual LEDs

- Alternatives
  - Seven segment displays
  - Bargraph
  - Dot matrix
  - Star-burst
Seven Segment Display

Common Cathode

Common Anode
Seven Segment Display

Multiplexing of seven segment digits

Connection | Port Bit
---|---
Segment a | RB7
Segment b | RB6
Segment c | RB5
Segment d | RB4
Segment e | RB3
Segment f | RB2
Segment g | RB1
Segment dp | RB0
Digit 1 drive | RA0
Digit 2 drive | RA1
Digit 3 drive | RA2
Digit 4 drive | RA3
Seven Segment Display

Multiplexing of seven segment digits

Segment drives

Digit drives

Digit 1

Digit 2

Digit 3

Digit 4

segment pattern for Digit 1

segment pattern for Digit 2

segment pattern for Digit 3

segment pattern for Digit 4
Seven Segment Display

Example

A program to count continuously the numbers 0 through 99 and display them on two seven segment displays. The count should be incremented every 1 sec. Oscillator frequency is 3 MHz.

- connect the seven segment inputs a through g to RB0 through RB6 respectively
- connect the gates of the controlling transistors to RA0 (LSD) and RA1 (MSD)
- the main program will be responsible for display and multiplexing every 5 ms
# Seven Segment Display Example

<table>
<thead>
<tr>
<th>Label</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>#INCLUDE</td>
<td>PICF84A.INC</td>
</tr>
<tr>
<td>LOW_DIGIT</td>
<td>EQU 0x20</td>
</tr>
<tr>
<td>HIGH_DIGIT</td>
<td>EQU 0x21</td>
</tr>
<tr>
<td>COUNT</td>
<td>EQU 0x22</td>
</tr>
<tr>
<td>ORG</td>
<td>0x0000</td>
</tr>
<tr>
<td>GOTO</td>
<td>START</td>
</tr>
<tr>
<td>ORG</td>
<td>0x0004</td>
</tr>
</tbody>
</table>

**ISR**

<table>
<thead>
<tr>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>GOTO ISR</td>
</tr>
</tbody>
</table>

**START**

<table>
<thead>
<tr>
<th>Label</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSF</td>
<td>STATUS, RP0</td>
</tr>
<tr>
<td>MOVFW B’00000000’</td>
<td>; set port B as output</td>
</tr>
<tr>
<td>MOVFW TRISB</td>
<td></td>
</tr>
<tr>
<td>MOVFW TRISA</td>
<td>; SET RA0-RA1 AS OUTPUT</td>
</tr>
<tr>
<td>BCF</td>
<td>STATUS, RP0</td>
</tr>
<tr>
<td>CLR PORTB</td>
<td></td>
</tr>
<tr>
<td>CLR PORTA</td>
<td></td>
</tr>
<tr>
<td>CLR LOW_DIGIT</td>
<td>; CLEAR THE COUNT VALUE</td>
</tr>
<tr>
<td>CLR HIGH_DIGIT</td>
<td></td>
</tr>
<tr>
<td>CLR COUNT</td>
<td></td>
</tr>
</tbody>
</table>
Seven Segment Display Example

DISPLAY BSF PORTA, 0
BCF PORTA, 1
MOVF LOW_DIGIT, W ; DISPLAY LOWER DIGIT
CALL TABLE ; GET THE SEVEN SEGMENT CODE
MOVWF PORTB
CALL DELAY_5MS ; KEEP IT ON FOR 5 MS
BCF PORTA, 0
BSF PORTA, 1
MOVF HIGH_DIGIT, W ; DISPLAY HIGH DIGIT
CALL TABLE ; GET THE SEVEN SEGMENT CODE
MOVWF PORTB
CALL DELAY_5MS ; KEEP IT ON FOR 5 MS
; CHECK IF 1 SEC ELAPSED
INCF COUNT, F ; INCREMENT THE COUNT VALUE IF TRUE
MOVF COUNT, W
SUBLW D’100’
BTFSS STATUS, Z
GOTO DISPLAY ; DISPLAY THE SAME COUNT
Seven Segment Display Example

; TIME TO INCREMENT THE COUNT
CLRF COUNT
INCF LOW_DIGIT, F ; INCREMENT LOW DIGIT AND CHECK IF > 9
MOVF LOW_DIGIT, W
SUBLW 0X0A
BTFSS STATUS, Z
GOTO DISPLAY
CLRF LOW_DIGIT

INCF HIGH_DIGIT, F ; INCREMENT HIGH DIGIT AND CHECK IF > 9
MOVF HIGH_DIGIT, W
SUBLW 0X0A
BTFSS STATUS, Z
GOTO DISPLAY
CLRF HIGH_DIGIT
GOTO DISPLAY
Seven Segment Display Example

DELAY_5MS

MOVLW D’250’

MOVWF 0X40

REPEAT

NOP

NOP

NOP

NOP

NOP

NOP

NOP

NOP

NOP

NOP

NOP

NOP

DECFSZ 0X40,1

GOTO REPEAT

RETURN
# Seven Segment Display Example

<table>
<thead>
<tr>
<th>TABLE</th>
<th>ADDWF</th>
<th>PCL,</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>RETLW</td>
<td>B'00111111'</td>
<td>;'0'</td>
<td></td>
</tr>
<tr>
<td>RETLW</td>
<td>B'00000110'</td>
<td>;'1'</td>
<td></td>
</tr>
<tr>
<td>RETLW</td>
<td>B'01011011'</td>
<td>;'2'</td>
<td></td>
</tr>
<tr>
<td>RETLW</td>
<td>B'01001111'</td>
<td>;'3'</td>
<td></td>
</tr>
<tr>
<td>RETLW</td>
<td>B'01100110'</td>
<td>;'4'</td>
<td></td>
</tr>
<tr>
<td>RETLW</td>
<td>B'01101101'</td>
<td>;'5'</td>
<td></td>
</tr>
<tr>
<td>RETLW</td>
<td>B'01111101'</td>
<td>;'6'</td>
<td></td>
</tr>
<tr>
<td>RETLW</td>
<td>B'00000111'</td>
<td>;'7'</td>
<td></td>
</tr>
<tr>
<td>RETLW</td>
<td>B'01111111'</td>
<td>;'8'</td>
<td></td>
</tr>
<tr>
<td>RETLW</td>
<td>B'01101111'</td>
<td>;'9'</td>
<td></td>
</tr>
</tbody>
</table>

END
Sensors

- Embedded systems need to interface with the physical world
- Must be able to detect the state of the physical variables and control them
- Input transducers or sensors are used to convert physical variables into electrical variables
  - Light sensors
  - Temperature sensors
- Output transducers convert electrical variables to physical; actuators
Sensors
The Microswitch
Sensors

Light-dependent Resistors

- A light-dependent resistor (LDR) is made from a piece of exposed semiconductor material.
- When light falls on it, it creates hole–electron pairs in the material, which improves the conductivity.

<table>
<thead>
<tr>
<th>Illumination (lux)</th>
<th>$R_{LDR}$ (Ohms)</th>
<th>$V_o$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dark</td>
<td>2M</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>9000</td>
<td>2.36</td>
</tr>
<tr>
<td>1000</td>
<td>400</td>
<td>0.19</td>
</tr>
</tbody>
</table>
Sensors
Optical Object Sensing

- Useful in sensing the presence or closeness of objects
- The presence of an object can be detected
  - If it breaks the light beam
  - If it reflects the light beam
Sensors

Opto-sensor as a Shaft Encoder

- Useful in measuring distance and speed
Sensors
Ultrasonic Object Sensor

- Based on reflective principle of ultrasonic waves
- An ultrasonic transmitter sends out a burst of ultrasonic pulses and then the receiver detects the echo
- If the time-to-echo is measured, distance can be measured

![Diagram showing the ultrasonic object sensor process.](image)
Actuators: motors and servos

- Embedded systems need to cause physical movement
- Linear or rotary motion
- Most actuators are electrical in nature
  - Solenoids (linear motion)
  - Servo motors (rotary motion)
  - DC or stepper motors (rotary motion)
DC Motors

- Range from the extremely powerful to the very small
- Wide speed range
- Controllable speed
- Good efficiency
- Can provide accurate angular positioning with angular shafts
- Only the armature winding needs to be driven
Stepper Motors

- A stepper motor (or step motor) is a synchronous electric motor that can divide a full rotation into a large number of steps.
Stepper Motors

• **Features**
  - Simple interface with digital systems
  - Can control speed and position
  - More complex to drive
  - Awkward start-up characteristics
  - Lose torque at high speed
  - Limited top speed
  - Less efficient
Servo Motors

- Allows precise angular motion
- The output is a shaft that can take an angular position over a range of 180°
- The input to the servo is a pulse stream whose width determines the angular position of the shaft
Interfacing to Actuators

- Microcontrollers can drive loads with small electrical requirements

- Some devices, like actuators, require high currents or supply voltages

- Use switching devices
  - Simple DC switching using BJTs or MOSFETs
  - Reversible DC switching using H-bridge
Interfacing to Actuators

Simple DC interfacing
Interfacing to Actuators

Simple DC interfacing

Resistive load

Inductive load
Interfacing to Actuators

Simple DC interfacing

Characteristics of two popular logic-compatible MOSFETs

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>ZVN4206A</th>
<th>ZVN4306A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum drain-to-source voltage, $V_{DS}$ (V)</td>
<td>60</td>
<td>60</td>
</tr>
<tr>
<td>Maximum gate-to-source threshold, $V_{GS(th)}$ (V)</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Maximum drain-to-source resistance when ‘on’, $R_{DS(on)}$ (Ω)</td>
<td>1.5</td>
<td>0.33</td>
</tr>
<tr>
<td>Maximum continuous drain current, $I_D$</td>
<td>600 mA</td>
<td>1.1 A</td>
</tr>
<tr>
<td>Maximum power dissipation (W)</td>
<td>0.7</td>
<td>1.1</td>
</tr>
<tr>
<td>Input capacitance (pF)</td>
<td>100</td>
<td>350</td>
</tr>
</tbody>
</table>
Interfacing to Actuators

Driving Piezo Sounder and Opto-sensors

- Piezo sounder ratings: 9mA, 3-20 V
- The opto-sensor found to operate well with 91 Ohm resistor. The diode forward voltage is 1.7V. The required current is about 17.6 mA
Interfacing to Actuators

Reversible DC Switching

- DC switching allows driving loads with current flowing in one direction.
- Some loads require the applied voltage to be reversible; DC motors rotation depends on the direction of current.
- Use H-bridge!
Interfacing to Actuators

Reversible DC Switching

L293D Dual H-bridge

Peak output current 1.2 A per channel
Interfacing to Actuators
Reversible DC Switching
Driving three motors using L293D
More on Digital Input

- When acquiring digital inputs into the microcontroller, it is essential that the input voltage is within the permissible and recognizable range of the MC

- Voltage range depends on the logic family; TTL, CMOS, ...

- Interfacing within the same family is safe

- What for the case
  - Interfacing to digital sensors
  - Signal corruption
  - Interference
More on Digital Input

PIC16F873A Port Characteristics

- 5.3 V: Irreversible device damage
- 5.0 V: Logic 1
- 2.0 V: Undefined logic level
- 0.8 V: Logic 0
- 0 V
- -0.3 V: Irreversible device damage
More on Digital Input

Forms of Signal Corruption

1. Spikes in the signal
2. Slow edge
3. DC Offset in the signal
More on Digital Input

Clamping Voltage Spikes

- All ports are usually protected by a pair of diodes
- An optional current limiting resistor can be added if high spikes are expected

If $R_{prot} = 1 \text{K} \Omega$ and the maximum diode current is 20 mA when $V_d = 0.3 \text{v}$, then what is the maximum positive voltage spike that can be suppressed?
More on Digital Input

Analog Input Filtering

- Can use Schmitt trigger for speeding up slow logic edges.
- Schmitt trigger with RC filter can be used to filter voltage spikes.
More on Digital Input

Switch Debouncing

- Mechanical switches exhibit bouncing behavior
- The switch contact bounces between open and closed
- A serious problem for digital devices ?!

![Simple switch interface](image)

- Switch debouncing!! hardware and/or software techniques
More on Digital Input

Switch Debouncing
More on Digital Input

Switch Debouncing

Switch output

\[ V_O \]

\[ V_S \]

Input polling

Perceived input

1

0

Switch bounce

\( t \)
Summary

- Microcontrollers must be able to interface with the physical world and possibly the human world
- Switches, keypads and displays represent typical examples for interfacing embedded systems with the humans
- Microcontrollers must be able to interface with a range of input and output transducers.
- Interfacing with sensors requires a reasonable knowledge of signal conditioning techniques
- Interfacing with actuators requires a reasonable knowledge of power switching techniques