Bahcesehir University

EEE3205 (902) Final Project

# Decimal to Hexadecimal Converter

*Esat Canberk Özçelik*
*1805100*

January 25, 2021

# Contents

# 1 Goal of the Project

The aim of the project is to convert decimal numbers to hexadecimal numbers inputted from a 4x4 keypad and later displayed on an LCD via Microchip PIC microcontrollers.

## 1.1 Components

- **Input:** 4x4 Keypad
  *Constructed from 16 individual push buttons.*
- **Output:** LCD (LM016L)
- **Micro-controller:** PIC16F877A
- **Quartz Crystal**
- **Potential Resistor:** 1kΩ
- **Capacitor:** 20pF *x2*

# 2 Hardware

For the sake of clearance and conciseness I have divided this section in to 3 phases:

- Timer
- LCD
- Keypad

## 2.1 Timer

For our circuit first we shall have a timer. Our micro-controller and the timer shall have the same clock frequency to satisfy the competence for the simultaneity between various components. Capacitors are connected parallel to the crystal which they will resonate together and cause it to oscillate on its fundamental parallel-resonant mode. And the other end of the capacitors are connected to ground. This is also called the crystal load capacitance. And then, it will be connected to the OSC ports of PIC16F877A.

**PIC16F87XA**

**TABLE 1-2: PIC16F873A/876A PINOUT DESCRIPTION**

| Pin Name | PDIP, SOIC, SSOP Pin# | QFN Pin# | I/O/P Type | Buffer Type | Description |
|---|---|---|---|---|---|
| OSC1/CLKI | 9 | 6 | | ST/CMOS(3) | Oscillator crystal or external clock input. |
| OSC1 | | | I | | Oscillator crystal input or external clock source input. ST buffer when configured in RC mode; otherwise CMOS. |
| CLKI | | | I | | External clock source input. Always associated with pin function OSC1 (see OSC1/CLKI, OSC2/CLKO pins). |
| OSC2/CLKO | 10 | 7 | | — | Oscillator crystal or clock output. |
| OSC2 | | | O | | Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. |
| CLKO | | | O | | In RC mode, OSC2 pin outputs CLKO, which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate. |

[1]

## 2.2 LCD

For the visual interface we are using LM016L LCD. LM016L is a 16x2 Liquid Crystal Display and supplied with +5V, which makes it the ideal display for our project. Brightness of the screen is controlled by a potential resistor. Data bus line is connected to the B port of our micro-controller.

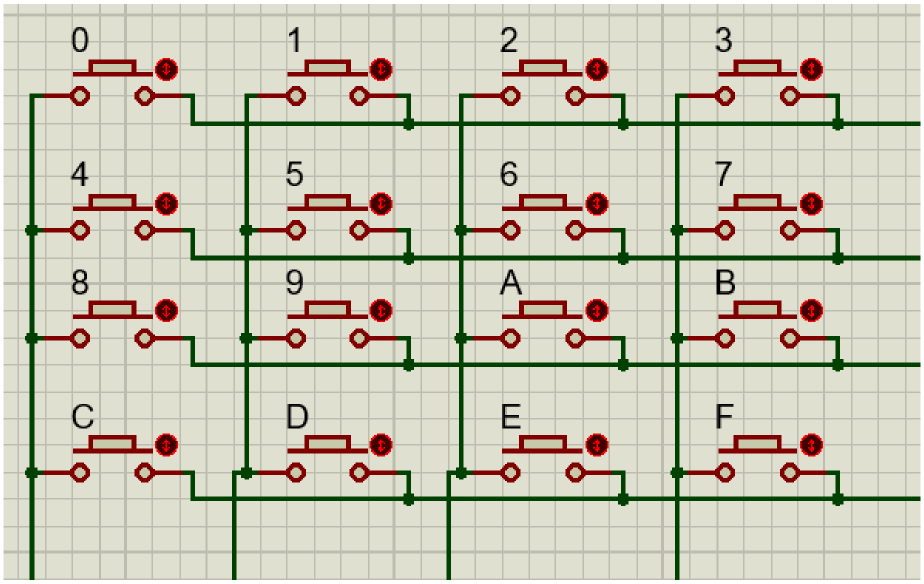
**INTERNAL PIN CONNECTION**

| Pin No. | Symbol | Level | Function | |
|---|---|---|---|---|
| 1 | $V_{SS}$ | – | 0V | Power supply |
| 2 | $V_{DD}$ | – | +5V | |
| 3 | $V_O$ | – | – | |
| 4 | RS | H/L | L: Instruction code input<br>H: Data input | |
| 5 | R/W | H/L | H: Data read (LCD module→MPU)<br>L: Data write (LCD module←MPU) | |
| 6 | E | H, H→L | Enable signal | |
| 7 | DB0 | H/L | Data bus line<br>Note (1), (2) | |
| 8 | DB1 | H/L | | |
| 9 | DB2 | H/L | | |
| 10 | DB3 | H/L | | |
| 11 | DB4 | H/L | | |
| 12 | DB5 | H/L | | |
| 13 | DB6 | H/L | | |
| 14 | DB7 | H/L | | |

[2]

## 2.3 Keypad

For the interface we are using a 4x4 keypad. This keypad is constructed by connecting 16 push-buttons in square formation, in parallel. Keypad is connected to the PORT D of PIC16F877A.



# 3 Software

## 3.1 Explanation

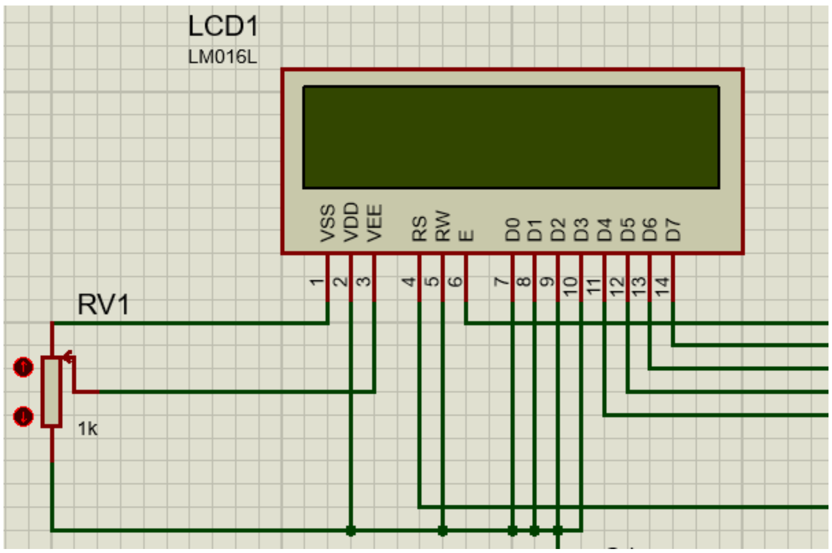
Although the code may be self explanatory by the added explanations here I will again explain code part by part in detail. For the entire code one may see the ***Subsection:*** *Entire Code.*

### 3.1.1 LCD Configurations

We are connecting the LCD to the PORT B of PIC16F877A. Here we set the PORTS, then the directions of these PORTS with TRIS function.

```
// Lcd pinout settings
sbit LCD_RS at RB0_bit;
sbit LCD_EN at RB1_bit;
sbit LCD_D7 at RB5_bit;
sbit LCD_D6 at RB4_bit;
sbit LCD_D5 at RB3_bit;
sbit LCD_D4 at RB2_bit;

// Pin direction
sbit LCD_RS_Direction at TRISB0_bit;
sbit LCD_EN_Direction at TRISB1_bit;
sbit LCD_D7_Direction at TRISB5_bit;
sbit LCD_D6_Direction at TRISB4_bit;
sbit LCD_D5_Direction at TRISB3_bit;
sbit LCD_D4_Direction at TRISB2_bit;
```



### 3.1.2 Initialising Data Types

Here we initialise the integers and chars we will use.

- **kp** is for the ASCII values of the buttons pressed.
- **count** is to count the amount of times buttons pressed in a session.
- **str** is for the converted value of n1 (from dec to hex).
- **i** resembles the decimal conversion of ASCII value of kp.
- **n1** is the summation of the pressed keys by their place value.
- **char keypadPort at PORTD;** is an initilaisation for us to use Keypad library.

```
unsigned short kp, count = 0;
char str[5];
int i;
int n1=0;
// Keypad module connections
char  keypadPort at PORTD;
```

### 3.1.3 Initialising Library Functions

- Initial count value is set to 0.
- Initialising Keypad_Init function of library.
- PORT D is configured as digital I/O.
- Initialise LCD function of library.
- Clear the display and turn off the cursor.
- Explanatory welcome message and a delay to keep it on screen for 1 second. Then clear the screen, making it ready for input.

```
void main() {
  count = 0;                          // Reset counter
  Keypad_Init();                    // Initialize Keypad
 PORTD = 0;                        // Configure AN pins as digital I/O

  Lcd_Init();                       // Initialize LCD
  Lcd_Cmd(_LCD_CLEAR);              // Clear display
  Lcd_Cmd(_LCD_CURSOR_OFF);         // Cursor off

  Lcd_Out(1, 1, "Press any Key");        // Write message text on LCD
  Lcd_Out(2, 1, "After 1 sec.");
  delay_ms(1000);
  Lcd_Cmd(_LCD_CLEAR);
```

#### 3.1.4 Keypad

- kp is set to 0.
- store values of pressed button to kp.
- In the switch function we have 16 different cases for each button valued from 0 to F (4x4 hex keypad).
  - Values written in the switch function are the ASCII codes of the given values.

```
do {
  kp = 0;                              // Reset key code variable
  // Wait for key to be pressed and released
  do
    kp = Keypad_Key_Click();           // Store key code in kp variable
  while (!kp);
 // Prepare value for output, transform key to it's ASCII value

  switch (kp) {
    case  1: kp = 67; break; // C
    case  2: kp = 68; break; // D
    case  3: kp = 69; break; // E
    case  4: kp = 70; break; // F
    case  5: kp = 56; break; // 8
    case  6: kp = 57; break; // 9
    case  7: kp = 65; break; // A
    case  8: kp = 66; break; // B
    case  9: kp = 52; break; // 4
    case 10: kp = 53; break; // 5
    case 11: kp = 54; break; // 6
    case 12: kp = 55; break; // 7
    case 13: kp = 48; break; // 0
    case 14: kp = 49; break; // 1
    case 15: kp = 50; break; // 2
    case 16: kp = 51; break; // 3
  }
```

| Dec | Hx | Oct | Char | | Dec | Hx | Oct | Html | Chr | Dec | Hx | Oct | Html | Chr | Dec | Hx | Oct | Html | Chr |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 000 | NUL | (null) | 32 | 20 | 040 |   | Space | 64 | 40 | 100 | @ | @ | 96 | 60 | 140 | ` | ` |
| 1 | 1 | 001 | SOH | (start of heading) | 33 | 21 | 041 | ! | ! | 65 | 41 | 101 | A | A | 97 | 61 | 141 | a | a |
| 2 | 2 | 002 | STX | (start of text) | 34 | 22 | 042 | " | " | 66 | 42 | 102 | B | B | 98 | 62 | 142 | b | b |
| 3 | 3 | 003 | ETX | (end of text) | 35 | 23 | 043 | # | # | 67 | 43 | 103 | C | C | 99 | 63 | 143 | c | c |
| 4 | 4 | 004 | EOT | (end of transmission) | 36 | 24 | 044 | $ | $ | 68 | 44 | 104 | D | D | 100 | 64 | 144 | d | d |
| 5 | 5 | 005 | ENQ | (enquiry) | 37 | 25 | 045 | % | % | 69 | 45 | 105 | E | E | 101 | 65 | 145 | e | e |
| 6 | 6 | 006 | ACK | (acknowledge) | 38 | 26 | 046 | & | & | 70 | 46 | 106 | F | F | 102 | 66 | 146 | f | f |
| 7 | 7 | 007 | BEL | (bell) | 39 | 27 | 047 | ' | ' | 71 | 47 | 107 | G | G | 103 | 67 | 147 | g | g |
| 8 | 8 | 010 | BS | (backspace) | 40 | 28 | 050 | ( | ( | 72 | 48 | 110 | H | H | 104 | 68 | 150 | h | h |
| 9 | 9 | 011 | TAB | (horizontal tab) | 41 | 29 | 051 | ) | ) | 73 | 49 | 111 | I | I | 105 | 69 | 151 | i | i |
| 10 | A | 012 | LF | (NL line feed, new line) | 42 | 2A | 052 | * | * | 74 | 4A | 112 | J | J | 106 | 6A | 152 | j | j |
| 11 | B | 013 | VT | (vertical tab) | 43 | 2B | 053 | + | + | 75 | 4B | 113 | K | K | 107 | 6B | 153 | k | k |
| 12 | C | 014 | FF | (NP form feed, new page) | 44 | 2C | 054 | , | , | 76 | 4C | 114 | L | L | 108 | 6C | 154 | l | l |
| 13 | D | 015 | CR | (carriage return) | 45 | 2D | 055 | - | - | 77 | 4D | 115 | M | M | 109 | 6D | 155 | m | m |
| 14 | E | 016 | SO | (shift out) | 46 | 2E | 056 | . | . | 78 | 4E | 116 | N | N | 110 | 6E | 156 | n | n |
| 15 | F | 017 | SI | (shift in) | 47 | 2F | 057 | / | / | 79 | 4F | 117 | O | O | 111 | 6F | 157 | o | o |
| 16 | 10 | 020 | DLE | (data link escape) | 48 | 30 | 060 | 0 | 0 | 80 | 50 | 120 | P | P | 112 | 70 | 160 | p | p |
| 17 | 11 | 021 | DC1 | (device control 1) | 49 | 31 | 061 | 1 | 1 | 81 | 51 | 121 | Q | Q | 113 | 71 | 161 | q | q |
| 18 | 12 | 022 | DC2 | (device control 2) | 50 | 32 | 062 | 2 | 2 | 82 | 52 | 122 | R | R | 114 | 72 | 162 | r | r |
| 19 | 13 | 023 | DC3 | (device control 3) | 51 | 33 | 063 | 3 | 3 | 83 | 53 | 123 | S | S | 115 | 73 | 163 | s | s |
| 20 | 14 | 024 | DC4 | (device control 4) | 52 | 34 | 064 | 4 | 4 | 84 | 54 | 124 | T | T | 116 | 74 | 164 | t | t |
| 21 | 15 | 025 | NAK | (negative acknowledge) | 53 | 35 | 065 | 5 | 5 | 85 | 55 | 125 | U | U | 117 | 75 | 165 | u | u |
| 22 | 16 | 026 | SYN | (synchronous idle) | 54 | 36 | 066 | 6 | 6 | 86 | 56 | 126 | V | V | 118 | 76 | 166 | v | v |
| 23 | 17 | 027 | ETB | (end of trans. block) | 55 | 37 | 067 | 7 | 7 | 87 | 57 | 127 | W | W | 119 | 77 | 167 | w | w |
| 24 | 18 | 030 | CAN | (cancel) | 56 | 38 | 070 | 8 | 8 | 88 | 58 | 130 | X | X | 120 | 78 | 170 | x | x |
| 25 | 19 | 031 | EM | (end of medium) | 57 | 39 | 071 | 9 | 9 | 89 | 59 | 131 | Y | Y | 121 | 79 | 171 | y | y |
| 26 | 1A | 032 | SUB | (substitute) | 58 | 3A | 072 | : | : | 90 | 5A | 132 | Z | Z | 122 | 7A | 172 | z | z |
| 27 | 1B | 033 | ESC | (escape) | 59 | 3B | 073 | ; | ; | 91 | 5B | 133 | [ | [ | 123 | 7B | 173 | { | { |
| 28 | 1C | 034 | FS | (file separator) | 60 | 3C | 074 | < | < | 92 | 5C | 134 | \ | \ | 124 | 7C | 174 | | | \| |
| 29 | 1D | 035 | GS | (group separator) | 61 | 3D | 075 | = | = | 93 | 5D | 135 | ] | ] | 125 | 7D | 175 | } | } |
| 30 | 1E | 036 | RS | (record separator) | 62 | 3E | 076 | > | > | 94 | 5E | 136 | ^ | ^ | 126 | 7E | 176 | ~ | ~ |
| 31 | 1F | 037 | US | (unit separator) | 63 | 3F | 077 | ? | ? | 95 | 5F | 137 | _ | _ | 127 | 7F | 177 | &#127; | DEL |



[3] ASCII Table

### 3.1.5 Operations on kp

- In the *if* loop, it is watching the count variable.
  - *count* watches the times the key is pressed.
- It is operating 3 times to get numbers upto 3 decimal inputs.
- *kp-'0'* converts the ASCII code to decimal value and we use *i* to represent it in the following line.
- In every loop, we multiply the initial value with 10 and add the new input.
  - For example: The inputs are 1,2,3.
    - **(initially n1=0)*
    - first loop: n1(0)=10x0+1
    - second loop: n1(1)=10x1+2
    - third loop: n1(12)=10x12+3
    - n1(final value)=123
- +1 is added to the integer *count* and the value of pressed button *kp* is displayed on LCD.

```
//ASCII code of kp is converted to a decimal value
if(count<3){
```

```
        i=kp-'0';
        n1=10*n1+i;
         }

    Lcd_Chr(1,++count, kp);      // Print key ASCII value on LCD
```

#### 3.1.6 Printing the Result

- If the button is pressed 3 times, this loop start to flow. The count variable can be increased or decreased according the needs. This number represents the number of decimal places.
- After all 3 decimal places have been entered, after 1 second the LCD is cleared and a message is given indicating the operation is being conducted.
- The *"loading.."* message is cleared.
- *IntToHex* function converts the decimal value of *n1* to hexadecimal value of *str*.
- *str* i.e. the final result is displayed on the screen.
- *count* is set to 0 for new operations.
- LCD is cleared.
- *n1* is set to 0 for new operations.
- do loop is closed with *while(1)* indicating to run indefinitely.

```
    if (count == 3) {            // If counter variable overflow
      delay_ms(1000);
      Lcd_Cmd(_LCD_CLEAR);
      Lcd_Out(2, 1, "Loading...");
      delay_ms(1000);
      Lcd_Cmd(_LCD_CLEAR);           // Clear display
      IntToHex(n1, str);         // Convert decimal str to hexadecimal
      Lcd_Out(2,1, str);           // Print the final result
      delay_ms(1000);
      count = 0;                     //Reset counter
      Lcd_Cmd(_LCD_CLEAR);            // Clear display
      n1=0;
      }

    } while (1);
}
```

### 3.2 Entire Code

```
// Lcd pinout settings
sbit LCD_RS at RB0_bit;
sbit LCD_EN at RB1_bit;
sbit LCD_D7 at RB5_bit;
sbit LCD_D6 at RB4_bit;
sbit LCD_D5 at RB3_bit;
sbit LCD_D4 at RB2_bit;

// Pin direction
sbit LCD_RS_Direction at TRISB0_bit;
sbit LCD_EN_Direction at TRISB1_bit;
sbit LCD_D7_Direction at TRISB5_bit;
sbit LCD_D6_Direction at TRISB4_bit;
sbit LCD_D5_Direction at TRISB3_bit;
sbit LCD_D4_Direction at TRISB2_bit;

unsigned short kp, count = 0;
char str[5];
int i;
int n1=0;
// Keypad module connections
char  keypadPort at PORTD;

 void main() {
  count = 0;                                  // Reset counter
  Keypad_Init();                              // Initialize Keypad
  PORTD = 0;                                  // Configure AN pins as digital I/O

  Lcd_Init();                                  // Initialize LCD
  Lcd_Cmd(_LCD_CLEAR);                         // Clear display
  Lcd_Cmd(_LCD_CURSOR_OFF);                    // Cursor off

  Lcd_Out(1, 1, "Press any Key");   // Write message text on LCD
  Lcd_Out(2, 1, "After 1 sec.");
  delay_ms(1000);
  Lcd_Cmd(_LCD_CLEAR);
  do {
    kp = 0;           // Reset key code variable
    // Wait for key to be pressed and released
    do
      kp = Keypad_Key_Click();           // Store key code in kp variable
    while (!kp);
   // Prepare value for output, transform key to it's ASCII value
```

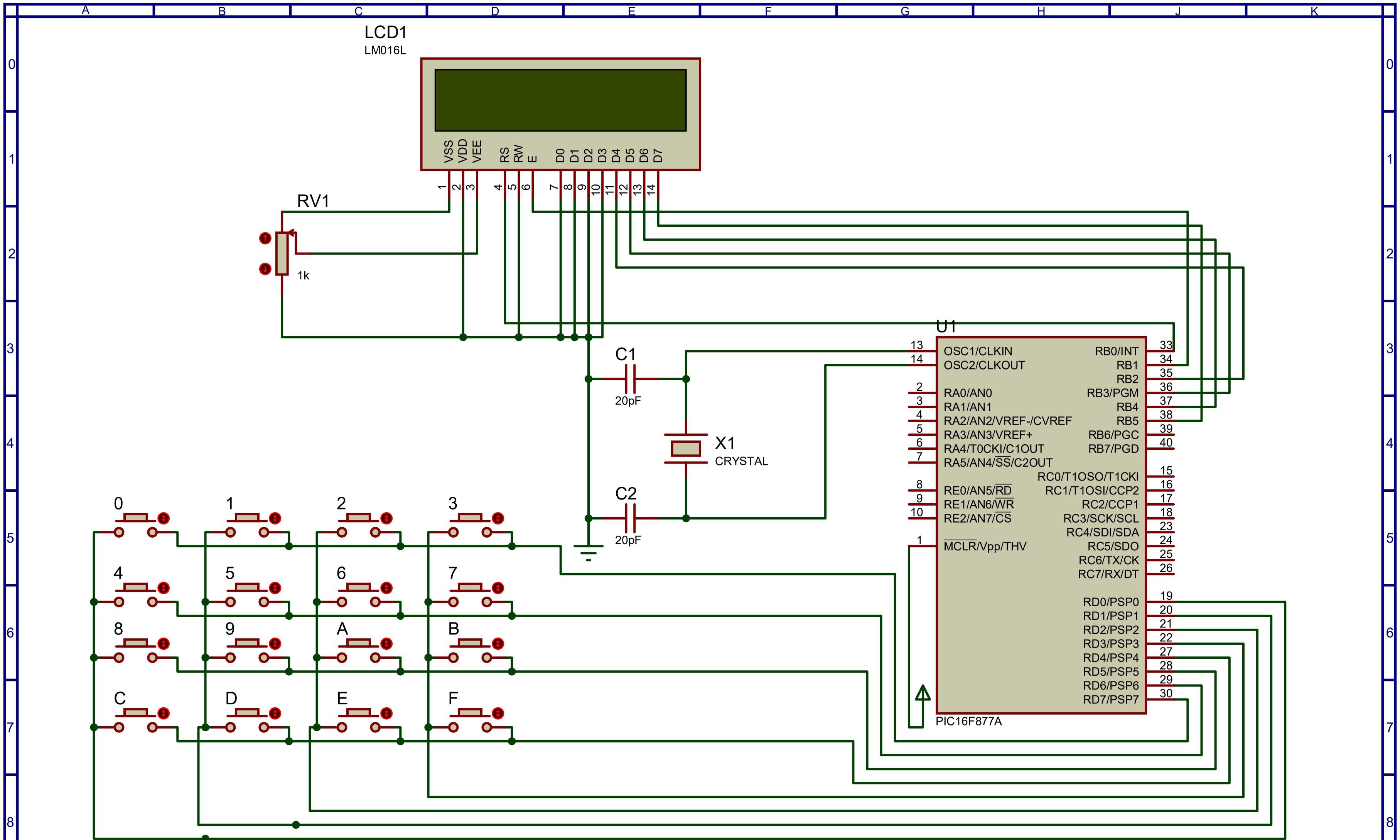
```
    switch (kp) {
      case  1: kp = 67; break; // 1
      case  2: kp = 68; break; // 2
      case  3: kp = 69; break; // 3
      case  4: kp = 70; break; // A
      case  5: kp = 56; break; // 4
      case  6: kp = 57; break; // 5
      case  7: kp = 65; break; // 6
      case  8: kp = 66; break; // B
      case  9: kp = 52; break; // 7
      case 10: kp = 53; break; // 8
      case 11: kp = 54; break; // 9
      case 12: kp = 55; break; // C
      case 13: kp = 48; break; // *
      case 14: kp = 49; break; // 0
      case 15: kp = 50; break; // #
      case 16: kp = 51; break; // D
    }
       //ASCII code of kp is converted to a decimal value
       if(count<3){
       i=kp-'0';
       n1=10*n1+i;
        }

    Lcd_Chr(1,++count, kp);           // Print key ASCII value on LCD

    if (count == 3) {                 // If counter variable overflow
      delay_ms(1000);
      Lcd_Cmd(_LCD_CLEAR);
      Lcd_Out(2, 1, "Loading...");
      delay_ms(1000);
      Lcd_Cmd(_LCD_CLEAR);            // Clear display
      IntToHex(n1, str);        // Convert decimal str to hexadecimal
      Lcd_Out(2,1, str);              // Print the final result
      delay_ms(1000);
      count = 0;                      //Reset counter
      Lcd_Cmd(_LCD_CLEAR);            // Clear display
      n1=0;
      }

    } while (1);
}
```

# 4 Circuit




| | | | |
|---|---|---|---|
| FILE NAME: | Hex.pdsprj | DATE: | 25.01.2021 |
| DESIGN TITLE: | Decimal to Hexadecimal Converter | PAGE: | 1 of 1 |
| PATH: | C:\Users\HP\Documents\Hex.pdsprj | | |
| BY: | Esat Canberk Özçelik REV:@REV | TIME: | 15:46:12 |

# References

[1] PIC16F87XA Data Sheet, © 2003 Microchip Technology Inc.

[2] LM061L.LM016XMBL Data Sheet, © Hitachi Semiconductor.

[3] ASCII table `http://www.asciitable.com/`

Esat Canberk OZCELIK
Student ID:1805100
Address: Mevlana mah. Serkan sok.
No:6 D:1 Maltepe/İstanbul-TURKEY
e-mail: esatcanberk.ozcelik@bahcesehir.edu.tr
Phone: +905078467183