

# Universidad Nacional Autónoma de México

## Facultad de Ingeniería

### Proyecto 1: Programa mediante puertos paralelos

Semestre: 2021-2

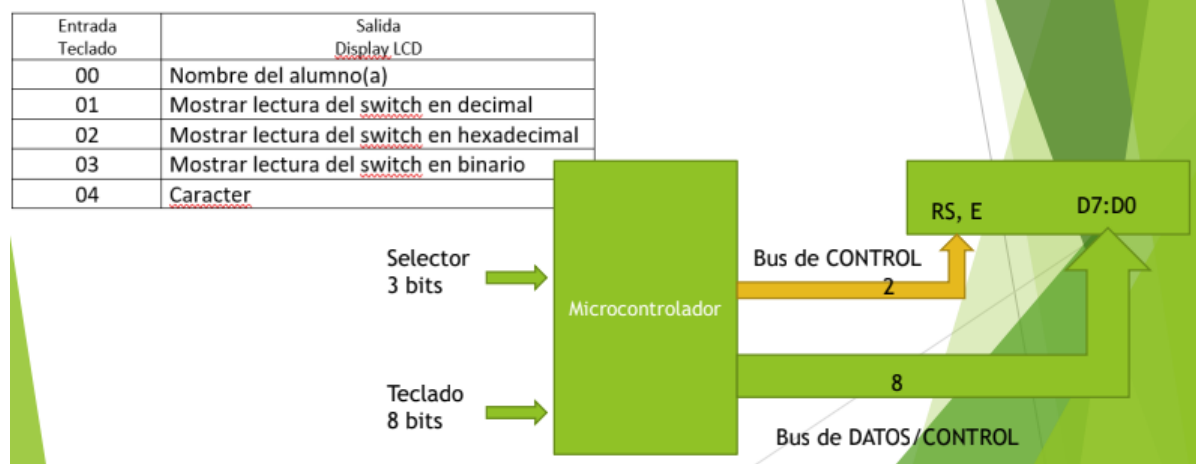
Profesor: Rubén Anaya García

Alumnos: Alfonso Murrieta Villegas

Grupo: 1

#### Descripción:

Para el presente programa es necesario contemplar cada uno de los casos propuestos por el profesor:



#### Apartado de hardware:

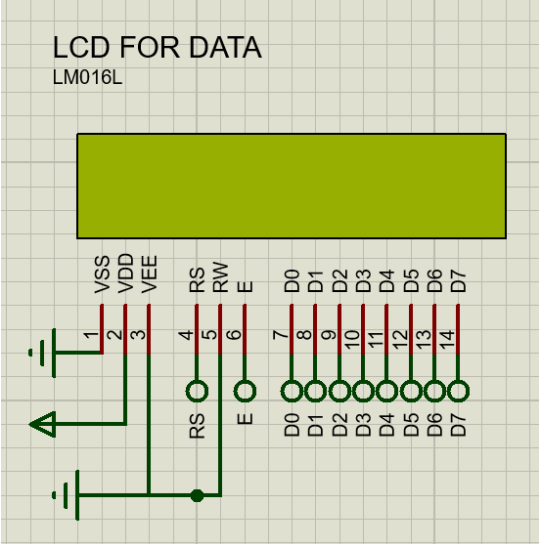
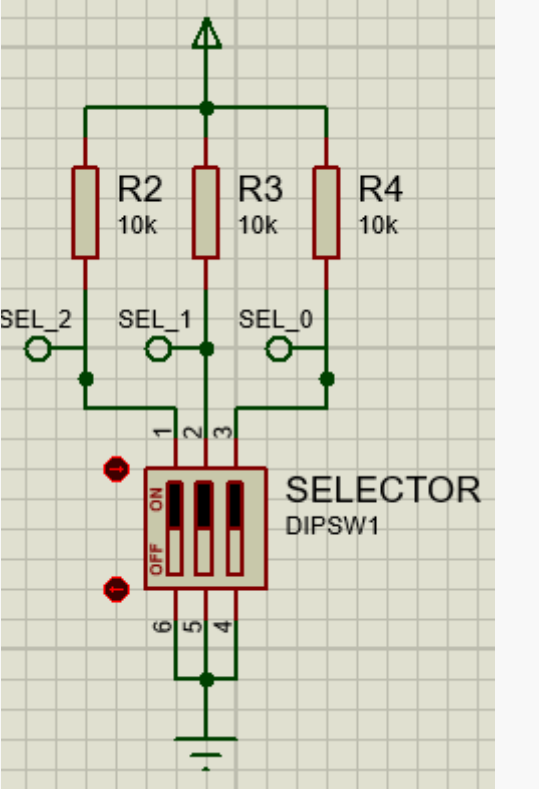
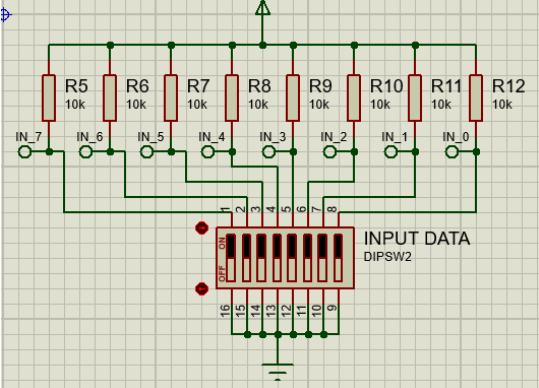
##### Requisitos en Hardware

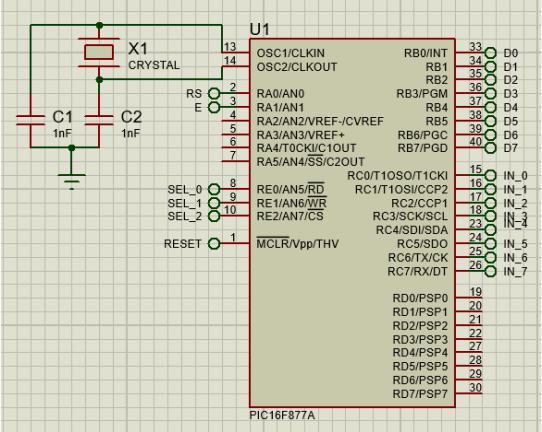
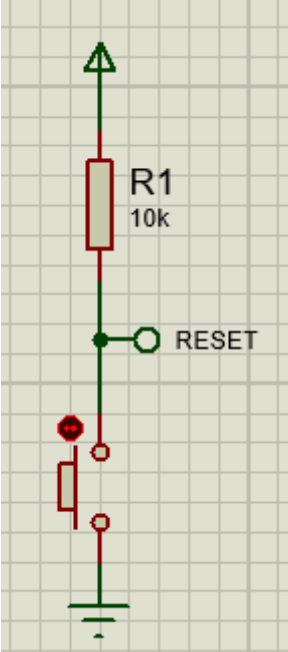
Con base en lo anterior, podemos contemplar los principales componentes de hardware de nuestro proyecto:

1. Un DIP switch que servirá como selector de cada uno de nuestros casos a representar
2. Un DIP switch que servirá como INPUT o entrada de nuestro sistema, es decir el que se empleará para ingresar los números a convertir
3. Un microcontrolador 16F877A que será el que llevará todo el procesamiento de nuestro proyecto
4. Un display LCD para poder reflejar nuestros datos de salida

## **Hardware y componentes en Proteus**

A continuación se hace una captura de pantalla de cada componente de hardware simulado mediante Proteus 8:

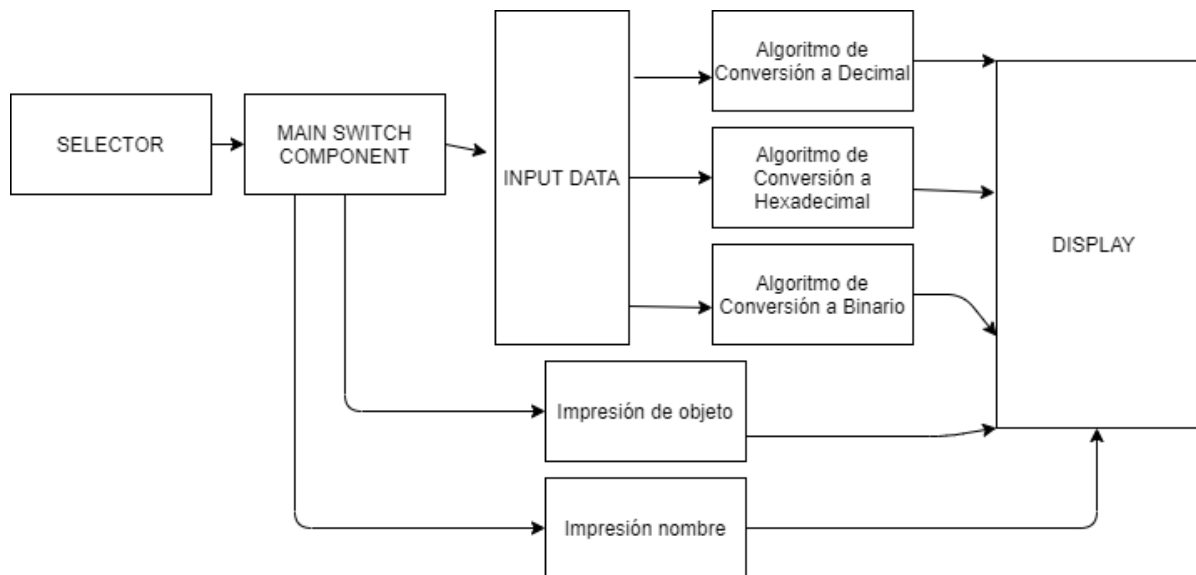
Componente	Descripción de Componente	Imagen
LCD	<p>Nos servirá para poder desplegar la información de salida de cada uno de nuestros casos.</p> <p>Podemos contemplar sus conexiones a nuestro micro además de la alimentación del mismo.</p>	 <p>The diagram shows an LCD module labeled 'LCD FOR DATA LM016L'. It features a green rectangular display area. Below the display, the pin connections are detailed: Pin 1 is VSS (ground), Pin 2 is VDD (positive supply), Pin 3 is VEE (negative supply), Pin 4 is RS (Reset/Select), Pin 5 is RW (Read/Write), Pin 6 is E (Enable), and Pins 7-14 are data lines D0 through D7. The connections are shown with green lines on a grid background.</p>
DIP switch 1 (Selector)	<p>Nos servirá para poder escoger cada uno de los casos de nuestro proyecto. Al ser en total 5 casos por ello es que se escogió de 3 entradas</p>	 <p>The diagram shows a 3-position DIP switch labeled 'SELECTOR DIPSW1'. The switch has positions for 'OFF', 'ON', and a third position. The connections are: Pin 1 to SEL_2, Pin 2 to SEL_1, and Pin 3 to SEL_0. Each selector line is connected to a 10k resistor (R2, R3, R4 respectively) which is then connected to ground. The switch is shown with green lines on a grid background.</p>
DIP switch 2 (INPUT DATA)	<p>Es aquel que se empleará para ingresar los números a convertir a convertir</p>	 <p>The diagram shows an 8-position DIP switch labeled 'INPUT DATA DIPSW2'. The connections are: Pin 1 to IN_7, Pin 2 to IN_6, Pin 3 to IN_5, Pin 4 to IN_4, Pin 5 to IN_3, Pin 6 to IN_2, Pin 7 to IN_1, and Pin 8 to IN_0. Each input line is connected to a 10k resistor (R5 through R12 respectively) which is then connected to ground. The switch is shown with green lines on a grid background.</p>

Componente	Descripción de Componente	Imagen
Microcontrolador PIC16F877A	Se puede apreciar un Cristal conectado al reloj de nuestro micro además de las respectivas conexiones a los switches	
Apartado de RESET	Sobre todo como recomendación del mismo datasheet del micro debemos contemplar un caso para el reset de nuestro microcontrolador, el cuál solo contempla un push botton y su respectiva conexión al micro	

## Apartado de Software:

### Primer acercamiento y diagramas

El primer acercamiento con el proyecto sin duda puede realizarse con un diagrama de componentes con el objetivo de poder entender la lógica que conllevará cada aspecto dentro del mismo:



1. Como se puede apreciar en el diagrama previo, lo primero que debemos emplear o considerar dentro del código es un selector que hará la función de interpretar los datos del switch selector.
2. Posteriormente y con base a los datos de entrada de nuestro selector es como podemos realizar ya sea la conversión debida o directamente mandar los datos de nuestro caso 0 (Impresión del nombre) y 5 (Impresión de nuestro carácter u objeto)
3. Por último, el componente del display tiene varias actividades que debemos contemplar:
  1. Debe refrescarse y configurarse previamente para determinar su tamaño y otros aspectos más
  2. Debemos limpiar los datos que este tiene, esto contemplando cuando se haga alguna otra acción en nuestro proyecto.
  3. Enviar los datos de nuestro micro al display físico

## Descripción del programa

A continuación se describirá a detalle cada uno de los segmentos y subrutinas del programa, además de como están que están ligados entre ellos.

## Definición de puertos y variables

Para la definición de puertos se consideraron como salidas al puerto A y B mientras que para la entrada se consideraron los puertos A, E y C

Cabe destacar que además se contemplaron las siguientes variables globales dentro de nuestro código:

```

PROCESSOR    16f877
INCLUDE      <p16f877.inc>

AUX          EQU  H'50'
COUNT       EQU  H'51'
INPUT        EQU  H'20'
INPUT1       EQU  H'21'
INPUT2       EQU  H'22'
  
```

```

FIN      EQU  H'23'
DATA_INF EQU  H'24'
SPC_PT   EQU  0X20

NUM_HEX   EQU  H'30'
DATA_C_HX EQU  H'31'
AUX_IN_C  EQU  H'32'
AUX_IN_D  EQU  H'33'
AUX_IN_U  EQU  H'34'
AUX_TP_HX EQU  H'36'
AUX_SB_HX EQU  H'37'

```

Donde el primer bloque está dedicado a tratar los datos que serán ingresados, mientras que el segundo bloque es para el manejo de los datos para su posterior impresión en el LCD.

A continuación el apartado de definición de puertos y limpieza de datos tras cada llamada de casos del proyecto:

```

INICIO
    BSF     STATUS,RP0
    BCF     STATUS,RP1
    MOVLW   0X0F
    MOVWF   ADCON1
    MOVLW   0X00
    MOVWF   TRISB
    MOVWF   TRISA
    MOVLW   0X07
    MOVWF   TRISE
    MOVLW   0XFF
    MOVWF   TRISC
    BCF     STATUS,RP0
    ;DONT MOVE OR DELETE THIS
    CLRF    PORTA
    CLRF    PORTB
    CLRF    PORTC
    CLRF    PORTE
    CLRF    AUX_IN_C
    CLRF    AUX_IN_D
    CLRF    AUX_IN_U
    CALL    DISP_IN

```

## Switch principal

Con base a la definición de los puertos, directamente ya podemos hacer un apartado dedicado únicamente a la selección de cada uno de los casos contemplados en este proyecto, a continuación se muestra el código:

```

CHECK_CASE ; SWITCH MENU
    MOVLW   H'00'
    XORWF   PORTE,W
    BTFSC   STATUS,Z
    GOTO    DATA_NAME ; FOR MY NAME
    MOVLW   H'01'
    XORWF   PORTE,W
    BTFSC   STATUS,Z

```

```

GOTO    DC_DATA ; FOR DECIMAL
MOVLW   H'02'
XORWF   PORTE,W
BTFSC   STATUS,Z
GOTO    HX_DATA ; FOR HEXADECIMAL
MOVLW   H'03'
XORWF   PORTE,W
BTFSC   STATUS,Z
GOTO    BIN_DATA; FOR BINARY
MOVLW   H'04'
XORWF   PORTE,W
BTFSC   STATUS,Z
GOTO    MAIN_FIGURE ; FOR FIGURE
GOTO    CHECK_CASE

```

### Subrutinas principales (Main) de cada caso

Antes de poder mandar la información a ser tratada debemos hacer una separación de cada caso, por ejemplo:

El caso 0 y 4 (El de impresión de nombre y de la figura) son casos "estáticos", es decir no serán cambiados a lo largo de la ejecución del programa, por lo que la información puede estar directamente ingresada en nuestro código.

A continuación algunos segmentos de estos códigos:

```

DATA_NAME
    MOVLW   0X80
    CALL    DATA__AUX1
    CALL    CLS_DSP
    MOVLW   A'A'
    CALL    DATA_AUX2
    MOVLW   A'L'
    CALL    DATA_AUX2
    MOVLW   A'F'
    CALL    DATA_AUX2
    MOVLW   A'O'
    CALL    DATA_AUX2

```

Respecto a los casos 1, 2, 3 que son los de la conversión del dato ingresado a su respectivo sistema numérico, aquí debemos contemplar varias acciones:

1. Obtener los datos ingresados
2. Hacer la conversión del dato ingresado a su respectiva conversión
  1. Aquí debemos contemplar que caso fue seleccionado
3. Mandar el dato obtenido a una función encargada de la impresión de los datos en el LCD

A continuación el método de datos para los números en decimal

```

DC_DATA ; INPUT TO DECIMAL
    MOVLW   0X80
    CALL    DATA__AUX1
    CLRF    AUX

```

```

MOVF    PORTC,W
MOVWF   NUM_HEX
MOVWF   DATA_C_HX
CALL    GET_DC
CALL    SEND_N_C_D
CALL    SEND_N_D_D
CALL    SEND_N_U_D
MOVLW   A' '
CALL    DATA_AUX2
MOVLW   A'D'
CALL    DATA_AUX2
MOVLW   0X09
MOVWF   COUNT
CALL    WHT_SPC
MOVLW   0XC0
CALL    DATA__AUX1
MOVLW   0X0F
MOVWF   COUNT
CALL    WHT_SPC
GOTO    SELECT

```

### Subrutinas auxiliares del display

Para este apartado principalmente se tiene 3 subrutinas pequeñas que se encargan principalmente de 2 aspectos importantes :

1. Hacer la limpieza del display mediante un "rellenado" de espacios en blanco, cabe mencionar que esta subrutina tiene que ser llamada en cada caso de nuestro proyecto debido a que no debe existir ningún percance en la información impresa en nuestro display.
2. Hacer un tipo de "pausa" o separador entre cada impresión y limpieza esto con el propósito de poder hacer un funcionamiento adecuado y correcto de nuestro display.

```

;CLEAN LCD
CLS_DSP
    MOVLW   SPC_PT
    MOVLW   A' '
    CALL    DATA_AUX2
    DECFSZ  SPC_PT
    GOTO    CLS_DSP

    MOVLW   0X80
    CALL    DATA__AUX1
    RETURN

STP_DSP
    MOVF    PORTE,W
    MOVWF   FIN
L_STP_DSP
    MOVF    PORTE,W
    XORWF   FIN,W
    BTFSS   STATUS,Z
    GOTO    SELECT
    GOTO    L_STP_DSP

```



## Algoritmos de conversiones de datos

Como bien sabemos existen 3 conversiones de datos solicitadas, de INPUT a hexadecimal, de INPUT a decimal y de INPUT a binario. A continuación se muestran los algoritmos de conversión:

### 1. Conversión de INPUT a binario

Como es de esperarse para este algoritmo lo único que se debe hacer es mandar la información directamente ingresada en el switch físico de datos hacia la subrutina de impresión y menú del display

```
GET_B
    RLF    DATA_C_HX
    BTFSS  STATUS,C
    GOTO   C_B_0
    GOTO   C_B_1
C_B_0
    CALL   NUM_0
    GOTO   GO_BN
C_B_1
    CALL   NUM_1
    GOTO   GO_BN
GO_BN
    DECFSZ CONT
    GOTO   GET_B
    RETURN
```

NOTA: Es necesario mencionar que la entrada de datos antes de llegar a la subrutina anterior pasa por la subrutina de obtención de binario del switch de entrada

```
BIN_DATA
    MOVLW  0x80
    CALL   DATA__AUX1
    CLRF   AUX
    MOVF   PORTC,W
    MOVWF  NUM_HEX
    MOVWF  DATA_C_HX
    MOVLW  0x08
    MOVWF  COUNT
    CALL   GET_B
    MOVLW  A' '
    CALL   DATA_AUX2
    MOVLW  A'B'
    CALL   DATA_AUX2
    MOVLW  0x05
    MOVWF  COUNT
    CALL   WHT_SPC
    MOVLW  0xC0
    CALL   DATA__AUX1
    MOVLW  0x10
    MOVWF  COUNT
    CALL   WHT_SPC
    GOTO   SELECT
```

### 2. Conversión de INPUT a hexadecimal

Para este algoritmo directamente se recuperó lo realizado en prácticas previas, donde recordemos que para la obtención del número en hexadecimal debemos realizar una división que recorrerá el data ingresado además de contemplar en cada iteración respecto la posición su respectivo residuo.

```
GET_T_HX
    MOVLW    0X04
    MOVWF    COUNT
GET_TP_HX_AUX
    RRF      DATA_C_HX, F
    BCF      STATUS, C
    DECFSZ   COUNT
    GOTO     GET_TP_HX_AUX
    MOVF     DATA_C_HX, W
    MOVWF    TOP_H
    MOVLW    0X04
    MOVWF    COUNT
    MOVF     NUM_HEX, W
    MOVWF    DATA_C_HX
GET_S_H
    RLF      DATA_C_HX, F
    BCF      STATUS, C
    DECFSZ   COUNT
    GOTO     GET_S_H
    SWAPF    DATA_C_HX
    MOVF     DATA_C_HX, W
    MOVWF    SUB_H
    RETURN
DIV_TP_HX
    MOVF     TOP_H, W
    MOVWF    AUX
    GOTO     PRINT_N
DIV_RS_HX
    MOVF     SUB_H, W
    MOVWF    AUX
    GOTO     PRINT_N
```

### 3. Conversión de INPUT a decimal

Al igual que el algoritmo previo, se considera una división de cada posición donde se contemplan tanto el residuo como el número obtenido en la posición.

La principal diferencia respecto al algoritmo previo, es el sistema a convertir además de considerar un apartado para mandar los datos obtenidos directamente a la subrutina del menú-display

```
GET_DC      ; CONV AND SEND DEC
    MOVLW    0X64
    CALL     DIV_DC
    MOVF     AUX, W
    MOVWF    NUM_CN
GET_DC_DC
    MOVLW    0X0A
    CLRF     AUX
    CALL     DIV_DC
```

```

MOVF    AUX,W
MOVWF   NUM_DC
GET_U_DC
    MOVLW 0X01
    CLRF  AUX
    CALL  DIV_DC
    MOVF  AUX,W
    MOVWF NUM_UN
    RETURN
DIV_DC
    SUBWF NUM_HEX,F
    BTFSS STATUS,C
    GOTO  AUX_FUNC
    INCF  AUX
    GOTO  DIV_DC
AUX_FUNC
    ADDWF NUM_HEX
    RETURN
SEND_N_C_D
    MOVF  NUM_CN,W
    MOVWF AUX
    GOTO  PRINT_N
SEND_N_D_D
    MOVF  NUM_DC,W
    MOVWF AUX
    GOTO  PRINT_N
SEND_N_U_D
    MOVF  NUM_UN,W
    MOVWF AUX
    GOTO  PRINT_N

```

## Impresión del carácter(caso 5)

Para este apartado solamente se empleó una subrutina, sin embargo, es necesario entender la lógica del código, debemos aterrizar el como pasaremos los datos a pixeles en nuestro display LCD:

A continuación se muestra un diagrama de como se escogieron los pixeles que estarían o no encendidos:

Podemos notar como es que el segmento del display fue partido en 4 secciones principales, de azul la sección 1, de color naranja la sección 2, de color gris la sección 3 y por último, la sección 4 de color verde, por otro lado, los pixeles de color negro son el carácter a representar

Con base a lo anterior, a continuación se muestra nuestra subrutina que está partida en 4 principales coordenadas donde cada MOVLW es un "renglón" de nuestra pantalla LCD. A continuación se muestra un segmento de la subrutina, mostrando la configuración en la coordenada o sección 1

```
PRINT_OBJ
    MOVLW    0X40
    CALL     DATA_AUX1
    MOVLW    B'00000'    ; COOR 1
    CALL     DATA_AUX2
    MOVLW    B'00000'
    CALL     DATA_AUX2
    MOVLW    B'00100'
    CALL     DATA_AUX2
    MOVLW    B'00100'
    CALL     DATA_AUX2
    MOVLW    B'00100'
    CALL     DATA_AUX2
    MOVLW    B'00100'
    CALL     DATA_AUX2
    MOVLW    B'00100'
    CALL     DATA_AUX2
    MOVLW    B'00100'
    CALL     DATA_AUX2
    MOVLW    B'00100'
    CALL     DATA_AUX2
```

### Selección de datos y menú de selección (Casos de conversión)

Para este apartado hubo dos principales subrutinas, la primera es la encargada de poder llevar a cabo la selección de los datos que van a ser dirigidos al display

```
WHT_SPC
    MOVLW    A' '
    CALL     DATA_AUX2
    DECFSZ   CONT
    GOTO     WHT_SPC
    RETURN

PRINT_N
    MOVLW    0X00
    XORWF    AUX,W
    BTFSC    STATUS,Z
    GOTO     NUM_0
    MOVLW    0X01
    XORWF    AUX,W
    BTFSC    STATUS,Z
    GOTO     NUM_1
    ...
    MOVLW    0X0D
    XORWF    AUX,W
    BTFSC    STATUS,Z
```

```

GOTO    NUM_D
MOVLW   0x0E
XORWF   AUX,W
BTFSC   STATUS,Z
GOTO    NUM_E
GOTO    NUM_F

```

Por otro lado, un apartado encargado de la asignación de cada uno de los números a desplegar, la cual es llamada por la subrutina anterior

```

; PRINT DATA
NUM_0
    MOVLW   A'0'
    CALL    DATA_AUX2
    RETURN
NUM_1
    MOVLW   A'1'
    CALL    DATA_AUX2
    RETURN
NUM_2
    MOVLW   A'2'
    CALL    DATA_AUX2
    RETURN
...
NUM_F
    MOVLW   A'F'
    CALL    DATA_AUX2
    RETURN

```

## Datos en el display

Para la impresión de los datos principalmente se emplearon 3 etiquetas o subrutinas:

1. La primera DISP\_IN es la encargada de configurar los datos del display como es el caso de que tantos datos van a desplegarse
2. Las subrutinas DATA\_R y DATA\_C son los encargadas de poder hacer la espera dentro del display (El refresco)

```

; Configuración del display para mostrar datos de 8 bits
DISP_IN
    MOVLW   0x30
    CALL    DATA_R
    CALL    WAIT_PRINT_2
    MOVLW   0x30
    CALL    DATA_R
    CALL    WAIT_PRINT_2
    MOVLW   0x38
    CALL    DATA_R
    MOVLW   0x0C
    CALL    DATA_R
    MOVLW   0x01
    CALL    DATA_R
    MOVLW   0x06

```

```

CALL    DATA_R
MOVLW   0X02
CALL    DATA_R
RETURN

```

; Este apartado es el encargado de hacer esperar dentro de display LCD

```

DATA_R
    MOVWF    PORTB
    CALL     WAIT_PRINT_1
    MOVLW    H'02'
    MOVWF    PORTA
    CALL     WAIT_PRINT_1
    MOVLW    H'00'
    MOVWF    PORTA
    CALL     WAIT_PRINT_1
    CALL     WAIT_PRINT_1
    RETURN

```

```

DATA_C
    MOVWF    PORTB
    CALL     WAIT_PRINT_1
    MOVLW    H'03'
    MOVWF    PORTA
    CALL     WAIT_PRINT_1
    MOVLW    H'01'
    MOVWF    PORTA
    CALL     WAIT_PRINT_1
    CALL     WAIT_PRINT_1
    RETURN

```

```

WAIT_PRINT_1
    MOVLW    0X02
    MOVWF    INPUT1

```

```

LOOP
    MOVLW    D'164'
    MOVFW    INPUT2

```

```

LOOP1
    DECFSZ   INPUT2
    GOTO     LOOP1
    DECFSZ   INPUT1
    GOTO     LOOP
    RETURN

```

```

WAIT_PRINT_2
    MOVLW    0X03
    MOVWF    INPUT

```

```

AUX_3
    MOVLW    0XFF
    MOVWF    INPUT1

```

```

AUX_2
    MOVLW    0XFF
    MOVWF    INPUT2

```

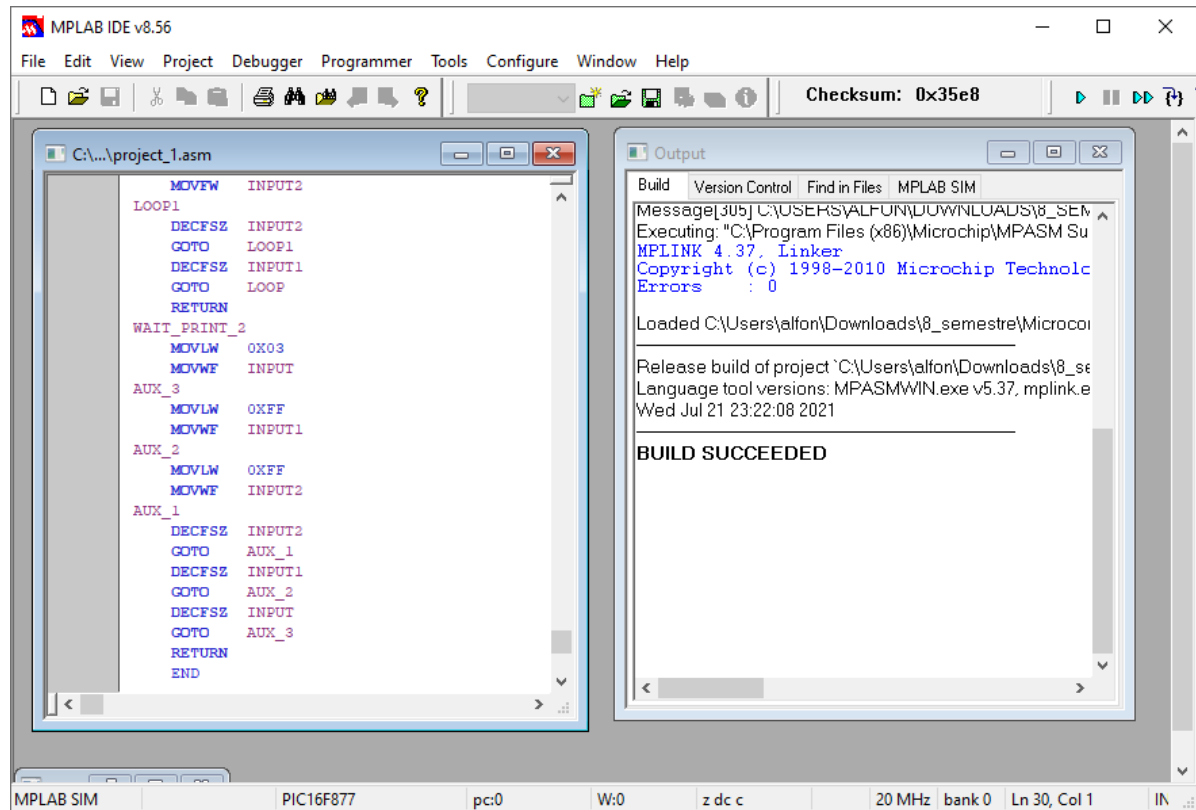
```

AUX_1
    DECFSZ   INPUT2
    GOTO     AUX_1
    DECFSZ   INPUT1
    GOTO     AUX_2
    DECFSZ   INPUT
    GOTO     AUX_3
    RETURN
END

```

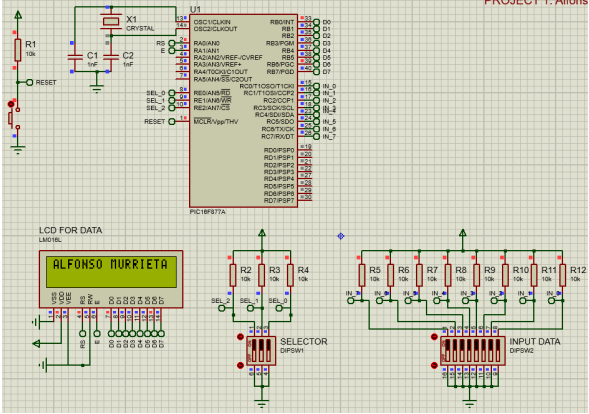
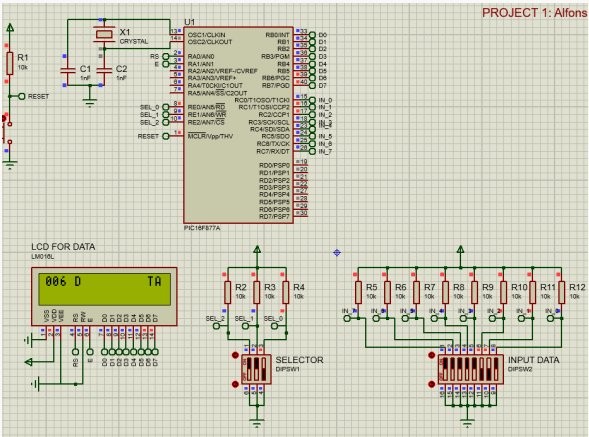
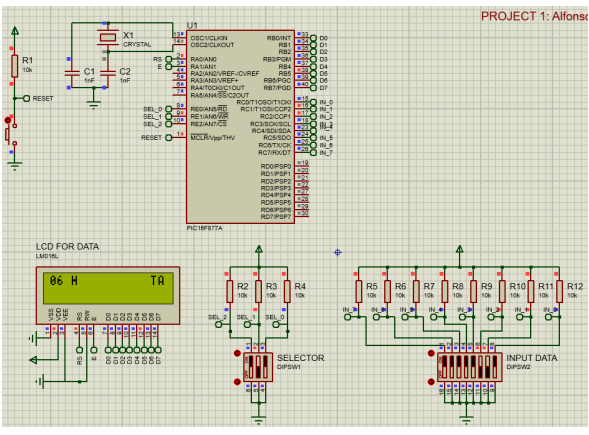
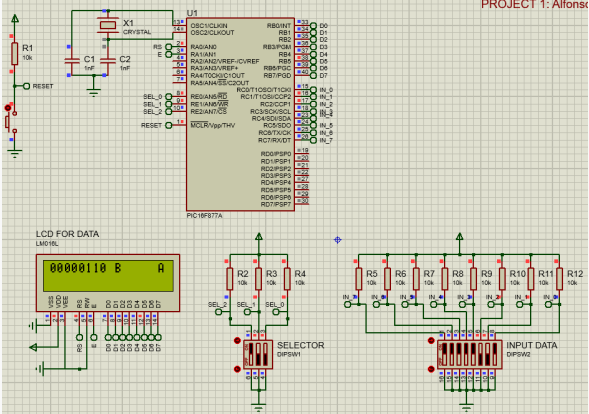
## Compilación y obtención de archivo .hex

Para poder simular nuestro código previo es necesario obtener un archivo.hex, a continuación, se muestra una captura de pantalla de la compilación en MPLAB:



## Simulación y resultados:

A continuación se muestran los resultados obtenidos mediante Proteus:

Caso	Switch Selector	Switch de datos	Resultado obtenido
Impresión de nombre	000	N.A	 <p>The circuit diagram shows a PIC16F877A microcontroller connected to a 10k resistor (R1), a crystal (X1), and capacitors (C1, C2). It is interfaced with an LCD1601. The LCD displays the name 'ALFONSO MURRIETA' in two lines. The circuit includes a selector switch (DIPSW1) and input data switches (DIPSW2) connected to the PIC's I/O pins.</p>
Conversión a Decimal	001	00000110	 <p>The circuit diagram is identical to the first one, but the LCD now displays '006 D' in two lines, indicating a conversion to decimal format. The switch selector (DIPSW1) is set to position 001.</p>
Conversión a Hexadecimal	010	00000110	 <p>The circuit diagram is identical to the first one, but the LCD now displays '06 H' in two lines, indicating a conversion to hexadecimal format. The switch selector (DIPSW1) is set to position 010.</p>
Conversión a Binario	011	00000110	 <p>The circuit diagram is identical to the first one, but the LCD now displays '00000110 B' in two lines, indicating a conversion to binary format. The switch selector (DIPSW1) is set to position 011.</p>





Datos ingresados	Simulación y resultado
11111111 a binario	
00000011 a binario	
00000011 a decimal	
00001011 a decimal	
00001011 a hexadecimal	

**Conclusiones:**

Sin duda este primer proyecto es una recopilación del uso de los puertos paralelos , del manejo de entrada y salida de datos y además del funcionamiento general de nuestro microcontrolador contemplando hardware externo como forma de poder interactuar con nuestro usuario o cliente.

Por otro lado, y como futuras modificaciones o mejoras al código podría realizarse un código más genérico y optimizado para la limpieza de nuestro microcontrolador esto debido a que existen residuos escritos en algunos casos de nuestro proyecto