



| | | | | | |
|-----------------------|------------------------------------|----------|--|-------------------------------|---------|
| Nombre de la práctica | Contador ascendente de 7 segmentos | | | No. | 1 |
| Asignatura: | Arquitectura de Computadoras | Carrera: | Ingeniería en Sistemas Computacionales | Duración de la práctica (hrs) | 6 horas |

NOMBRE DEL ALUMNO: María Elena Flores Arce

GRUPO: 312

- I. **Competencia(s) específica(s):**
Contador ascendente de 7 segmentos
- II. **Lugar de realización de la práctica (laboratorio, taller, aula u otro):**
Aula
- III. **Material empleado:**
Tabla protoboard
PIC16F84A
Cátodo común de 7 segmentos
Cristal oscilador de 5 picofaradios
Cable UTP
Resistencia
Capacitores cerámicos de 5 MHz
Programador de PIC
Aplicación MPLAB
Aplicación PORTEUS 8 PROFESIONAL

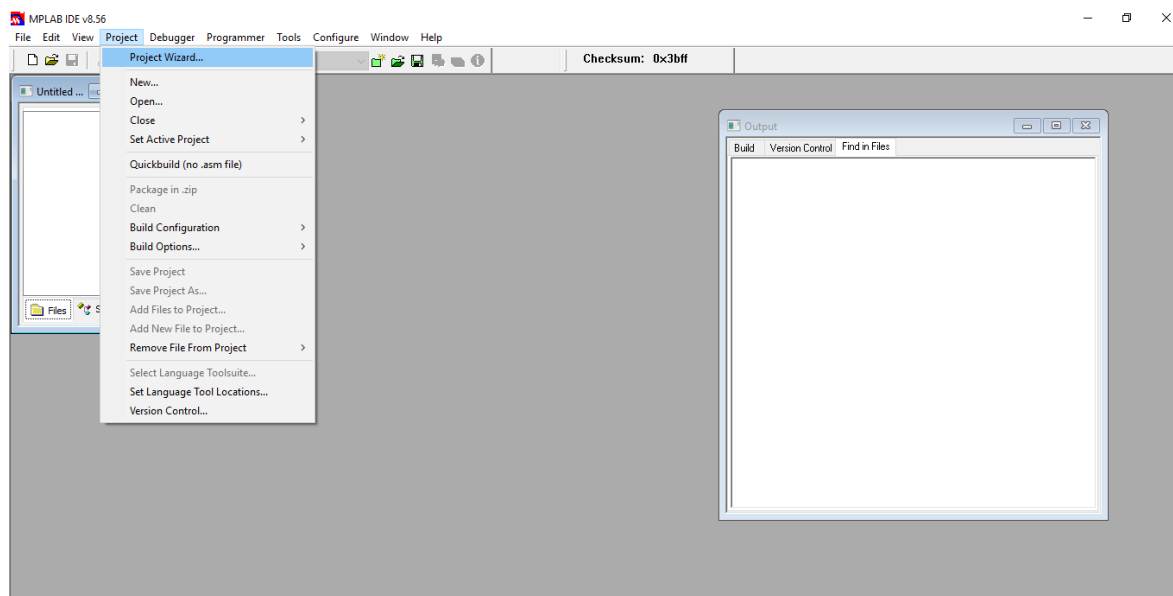
IV. Desarrollo de la práctica:

MPLAB

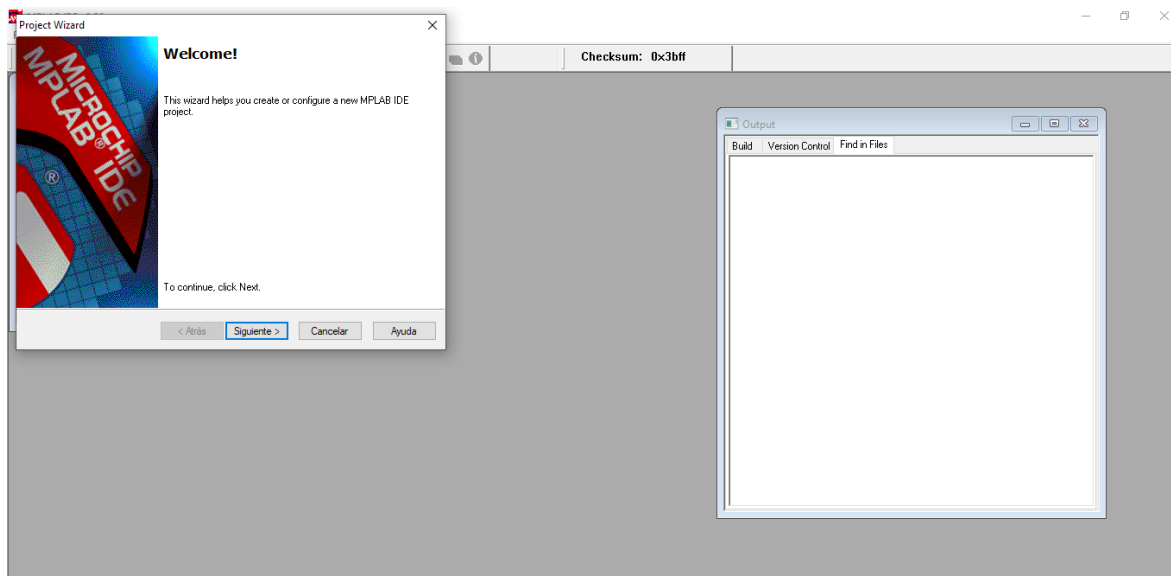
Se dio inicio con la instalación de la aplicación MPLAB para poder realizar la creación del diagrama utilizado



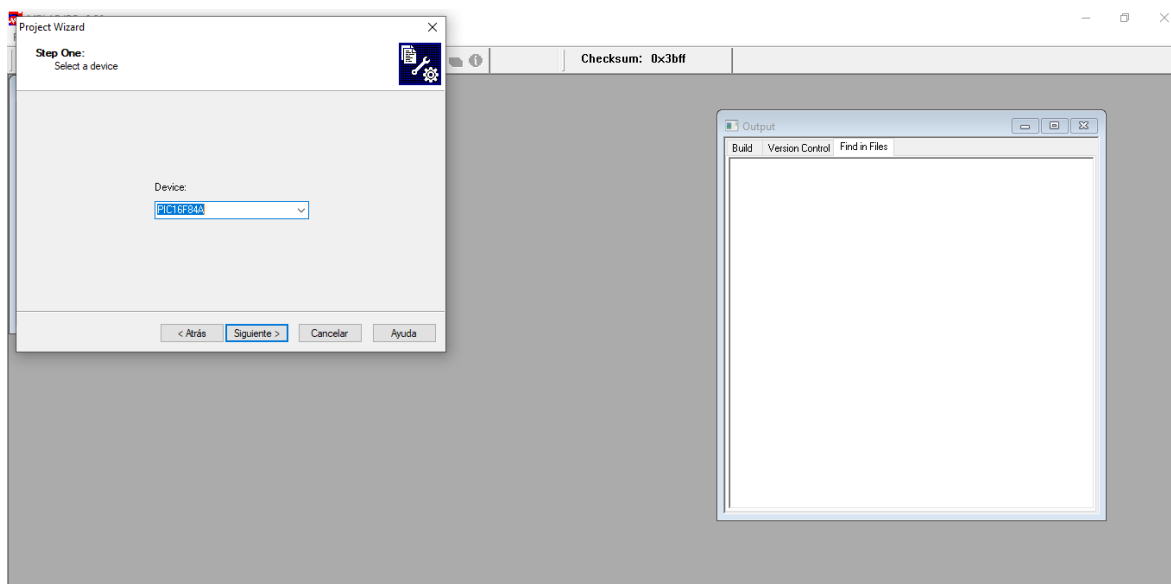
Se dio origen a la elaboración de un nuevo proyecto



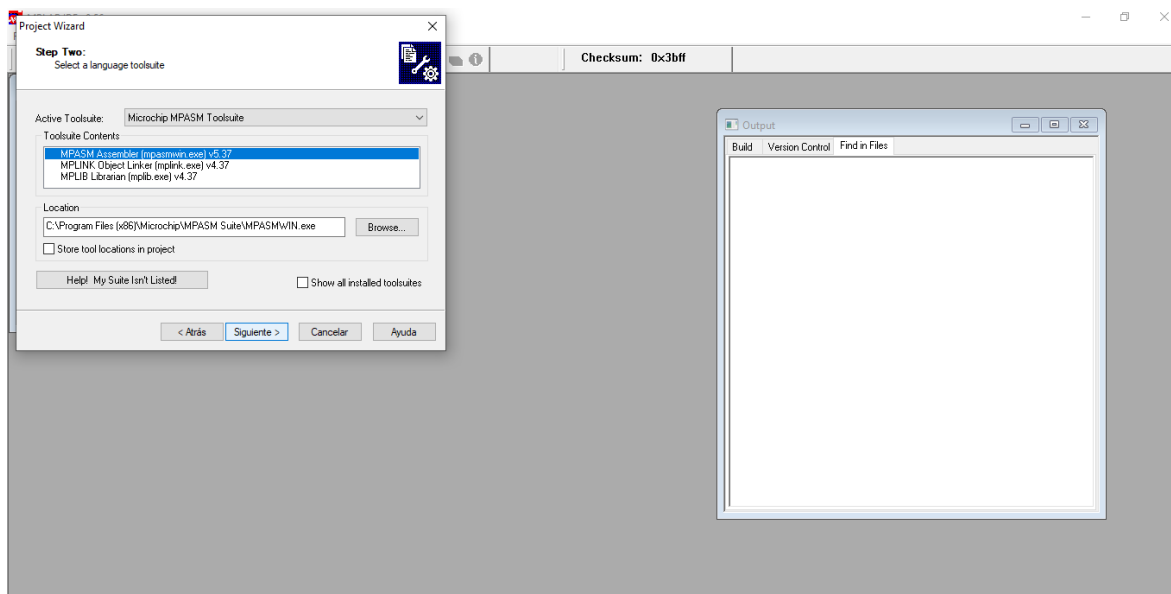
Se creo el proyecto a elaborar



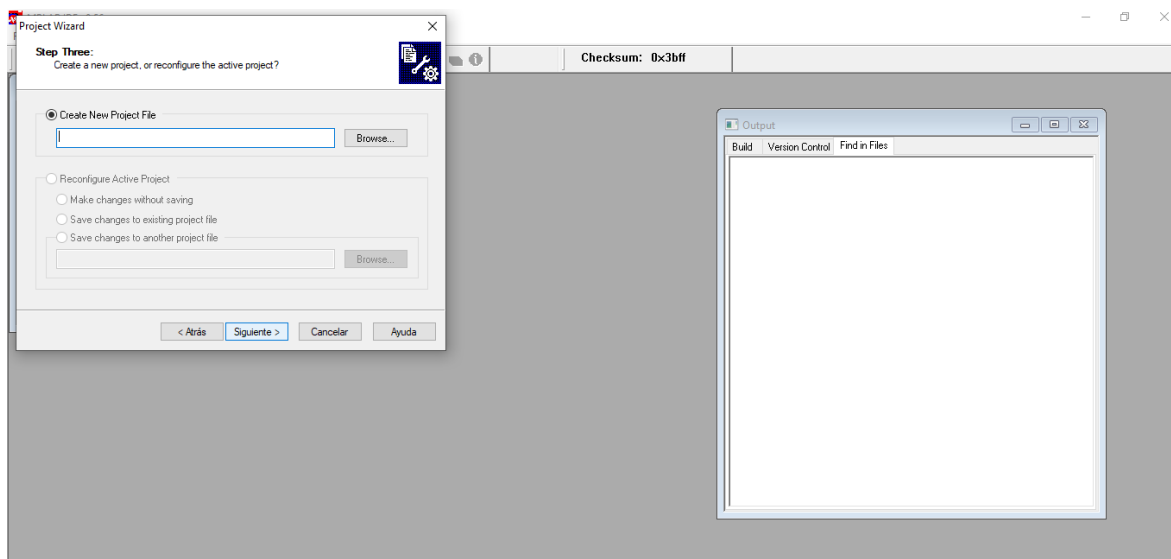
En esta parte se eligio el tipo de PIC que se desea utilizar, en este caso se utilizo el PIC16F84A



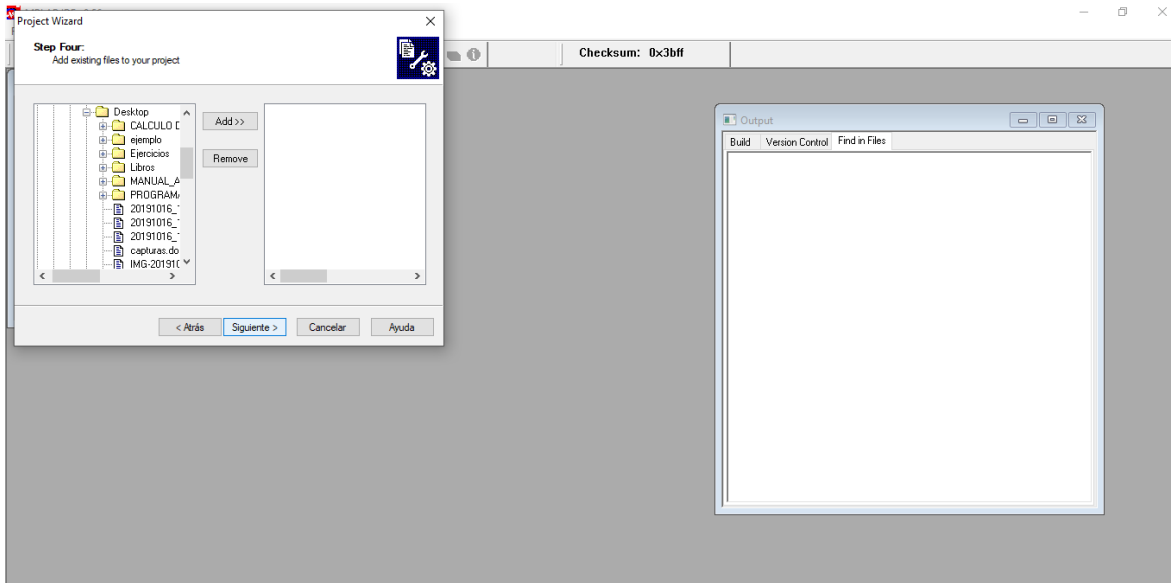
Se asigno el tipo de lenguaje en el cual se realizara el codigo referente al PIC



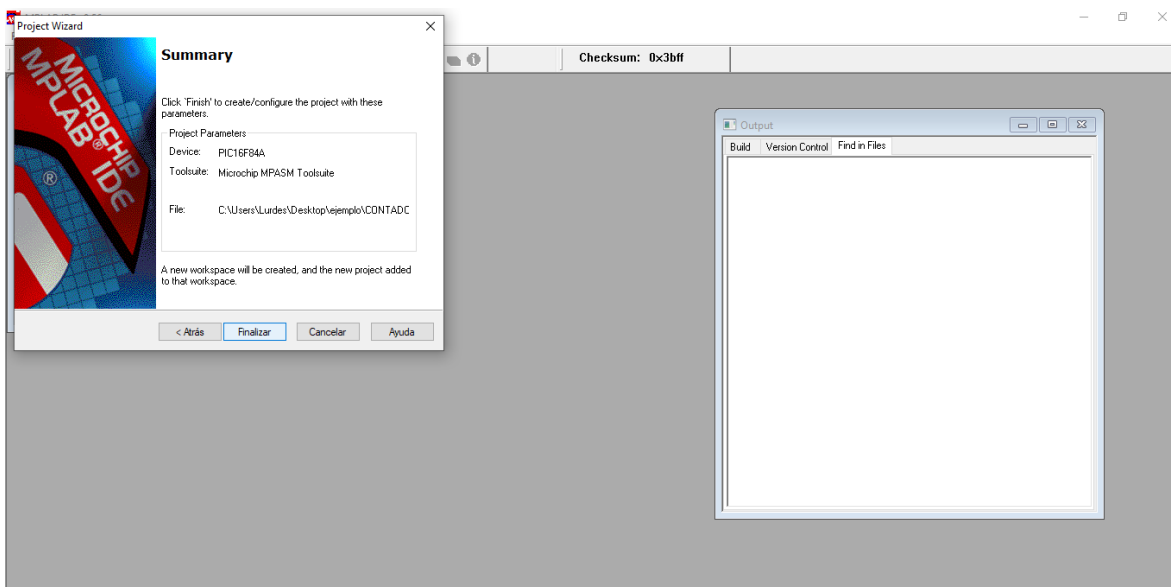
Se asigno el nombre del proyecto



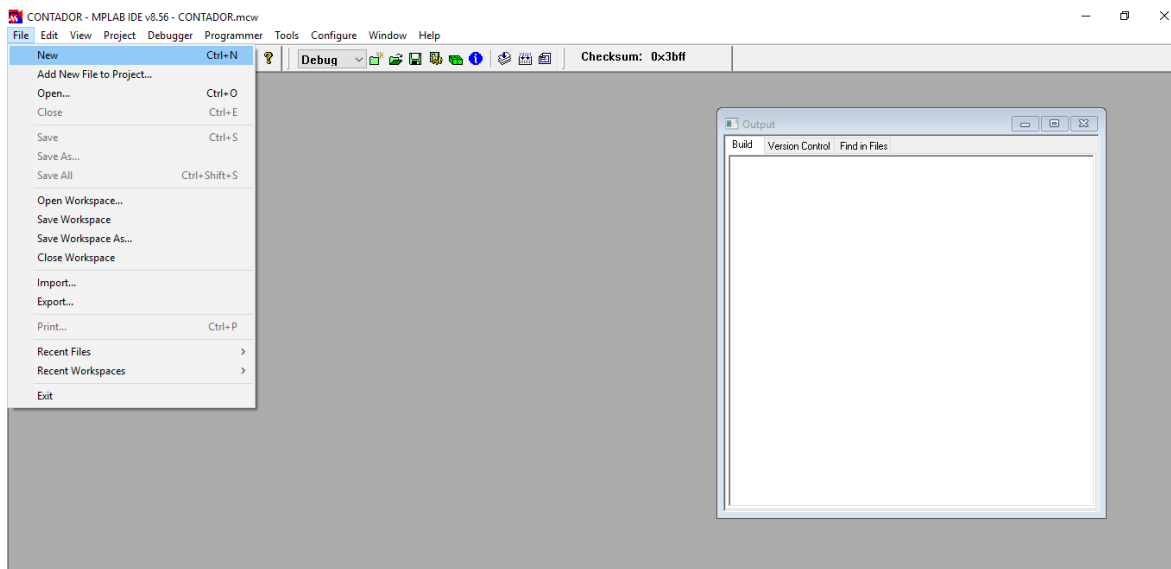
Se elige la carpeta en donde se encontrara la ubicación del proyecto



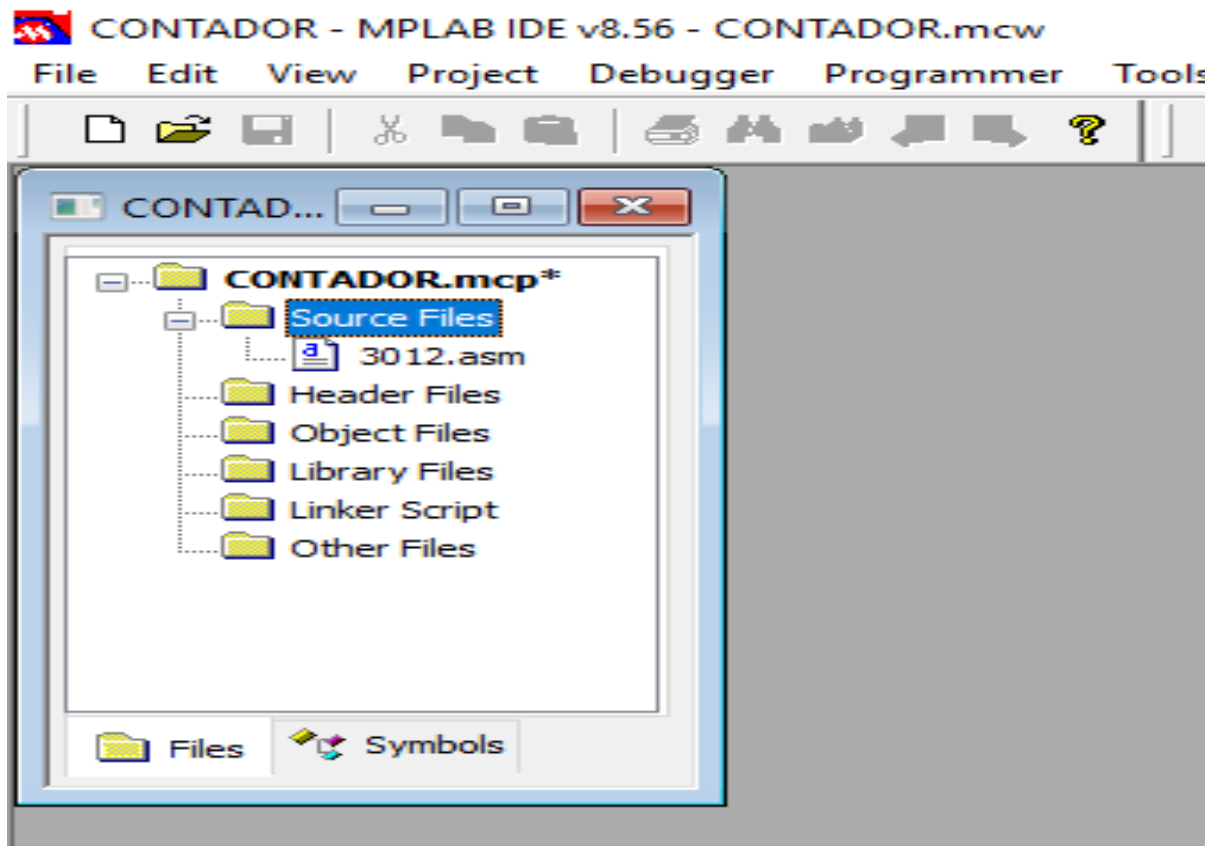
Se inició la escritura del código en ensamblador



Se abrió el proyecto con la ubicación en la que previamente se había guardado



Se abre la ruta de la ubicación del código





Se realizó la escritura del código con las características de la práctica en lenguaje ensamblador y con el sistema hexadecimal para ello ya previamente se tuvo que conocer las características del sistema hexadecimal.

```
contador - MPLAB IDE v8.56 - [E:\contador_\1\contador_Armando.asm]
File Edit View Project Debugger Programmer Tools Configure Window Help
Debug Checksum: 0x3bff

; zona de datos*****
LIST P=PIC16F84A
INCLUDE <P16F84A.INC>

; ZONA DE VARIABLES*****
CBLOCK 0x0C ; SE INICIALIZA LA MEMORIA
    NUMERO ; VARIABLE QUE LLEVARA EL CONTADOR DE 0-9 Y DE A-F
    CONTADOR ; LLEVA EL TIEMPO EN CICLOS DE RELOJ
ENDC

ORG 0
GOTO START
ORG 5

; CONFIGURACION*****
START BSF STATUS,5 ;BANCO 1 ACTIVA EL BIT B DE F
    CLRWF TRISB ;PORTE ES LA SALIDA
    MOVWL 0x1F ;MOVER A LA PARTE DE MEMORIA RA0 RA4SON LAS ENTRADAS
    MOVWF TRISA ;MOVER A F A TRISA
    MOVWL B'11000111' ;ASIGNA 256 AL TIMER
    MOVWF OPTION_REG
    BCF STATUS,5 ;CARGAR EL CONTENIDO DE LA POSICION 5 AL BANCO 0
    CLRW ;DEJA A W=0
    CLRWF NUMERO ;LIMPIA LA VARIABLE NUMERO

; INICIO*****
MAIN MOVF NUMERO,W ;TOMA EL CONTENIDO DE W Y LO CARGA EN F
    CALL TABLA ;LLAMA A LA FUNCION TABLA
    MOVWF PORTB ;MUESTRA EL VALOR QUE TOMO DE LA TABLA
    CALL PAUSE_1000 ;LLAMA A LA FUNCION PAUSE
    INCF NUMERO,F ;REGRESO DE INCREMENTO EN LA VARIABLE EN 1
    MOVF NUMERO,W ;SE CARGA LA VARIABLE EN W
    XORLW 0x10 ;SE COMPARA SI LLEGA AL REGISTRO 10
    BTFSF STATUS,Z ;VERIFICA SI HA LLEGADO
    GOTO MAIN ;SE VA A REGRESAR AL MAIN
    CLRW ;AL LLEGAR A 10 SE RESETEA A F
    CLRWF NUMERO ;SE LIMPIA A NUMERO
    GOTO MAIN ;SE REGRESA A MAIN

;FIN DE INICIO*****
;PAUSE_1000 SE REALIZA UN RETARDO DE 1 SEGUNDO*****
PAUSE_1000 MOVWL 0x02 ;SE LE ASIGNA 1000 AL CONTADOR
    MOVWF CONTADOR ;SE MUEVE EL VALOR DE CONTADOR DE A A F
    BCF INTCON,TOIF ;SE LIBERA EL BIT DE DESBORDAMIENTO THRO
    DELAY ;SE CARGA EL 217 A W
    MOVWL 0x05
    MOVWF THRO ;A THRO
    BTFSF INTCON,TOIF ;SE LIBERA EL BIT DE DESBORDAMIENTO EN THRO
    GOTO DELAY2 ;BUCLE DE DELAY2
    DECFSE CONTADOR,F ;DECREMENTA EN 1 EL CONTADOR
    GOTO DELAY ;BUCLE EN DELAY
    RETURN ;REGRESA

;TABLA*****
TABLA ADDWF PCL,F ;SE INICIALIZA LA FUNCION TABLA CON EL CONTENIDO DE F
    RETLW B'00111111' ;0 EN EL CATODO DE 7 SEGMENTOS
    RETLW B'00000110' ;1
    RETLW B'01011011' ;2
    RETLW B'01001111' ;3
    RETLW B'01100110' ;4
    RETLW B'01101101' ;5
    RETLW B'01111101' ;6
    RETLW B'01000111' ;7
    RETLW B'01111111' ;8
    RETLW B'01100111' ;9
    RETLW B'01110111' ;A
    RETLW B'01111100' ;b
    RETLW B'00111001' ;C
    RETLW B'01011110' ;d
    RETLW B'01111001' ;E
    RETLW B'01110001' ;F
END
```

```
contador - MPLAB IDE v8.56 - [E:\contador_\1\contador_Armando.asm]
File Edit View Project Debugger Programmer Tools Configure Window Help
Debug Checksum: 0x3bff

    INCF NUMERO,F ;REGRESO DE INCREMENTO EN LA VARIABLE EN 1
    MOVF NUMERO,W ;SE CARGA LA VARIABLE EN W
    XORLW 0x10 ;SE COMPARA SI LLEGA AL REGISTRO 10
    BTFSF STATUS,Z ;VERIFICA SI HA LLEGADO
    GOTO MAIN ;SE VA A REGRESAR AL MAIN
    CLRW ;AL LLEGAR A 10 SE RESETEA A F
    CLRWF NUMERO ;SE LIMPIA A NUMERO
    GOTO MAIN ;SE REGRESA A MAIN

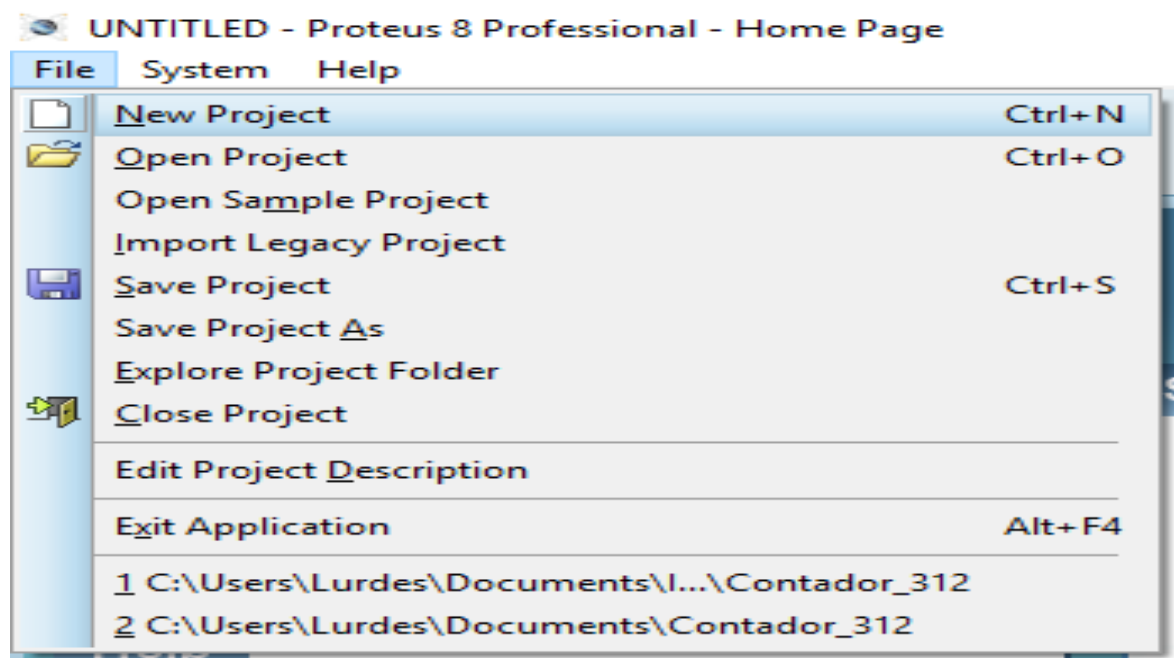
;FIN DE INICIO*****
;PAUSE_1000 SE REALIZA UN RETARDO DE 1 SEGUNDO*****
PAUSE_1000 MOVWL 0x02 ;SE LE ASIGNA 1000 AL CONTADOR
    MOVWF CONTADOR ;SE MUEVE EL VALOR DE CONTADOR DE A A F
    BCF INTCON,TOIF ;SE LIBERA EL BIT DE DESBORDAMIENTO THRO
    MOVWL 0x05
    MOVWF THRO ;A THRO
    BTFSF INTCON,TOIF ;SE LIBERA EL BIT DE DESBORDAMIENTO EN THRO
    GOTO DELAY2 ;BUCLE DE DELAY2
    DECFSE CONTADOR,F ;DECREMENTA EN 1 EL CONTADOR
    GOTO DELAY ;BUCLE EN DELAY
    RETURN ;REGRESA

;TABLA*****
TABLA ADDWF PCL,F ;SE INICIALIZA LA FUNCION TABLA CON EL CONTENIDO DE F
    RETLW B'00111111' ;0 EN EL CATODO DE 7 SEGMENTOS
    RETLW B'00000110' ;1
    RETLW B'01011011' ;2
    RETLW B'01001111' ;3
    RETLW B'01100110' ;4
    RETLW B'01101101' ;5
    RETLW B'01111101' ;6
    RETLW B'01000111' ;7
    RETLW B'01111111' ;8
    RETLW B'01100111' ;9
    RETLW B'01110111' ;A
    RETLW B'01111100' ;b
    RETLW B'00111001' ;C
    RETLW B'01011110' ;d
    RETLW B'01111001' ;E
    RETLW B'01110001' ;F
END
```

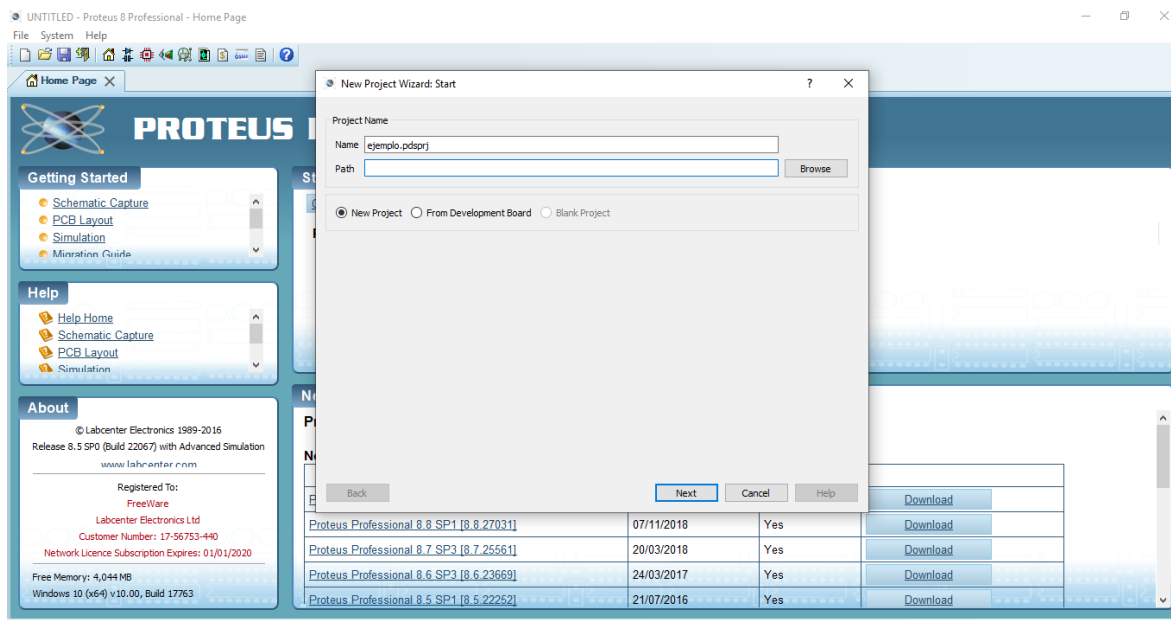
PORTEUS 8 PROFESIONAL

Tras la elaboración del código en la aplicación MPLAB se realizó el diagrama correspondiente al PIC16F84A

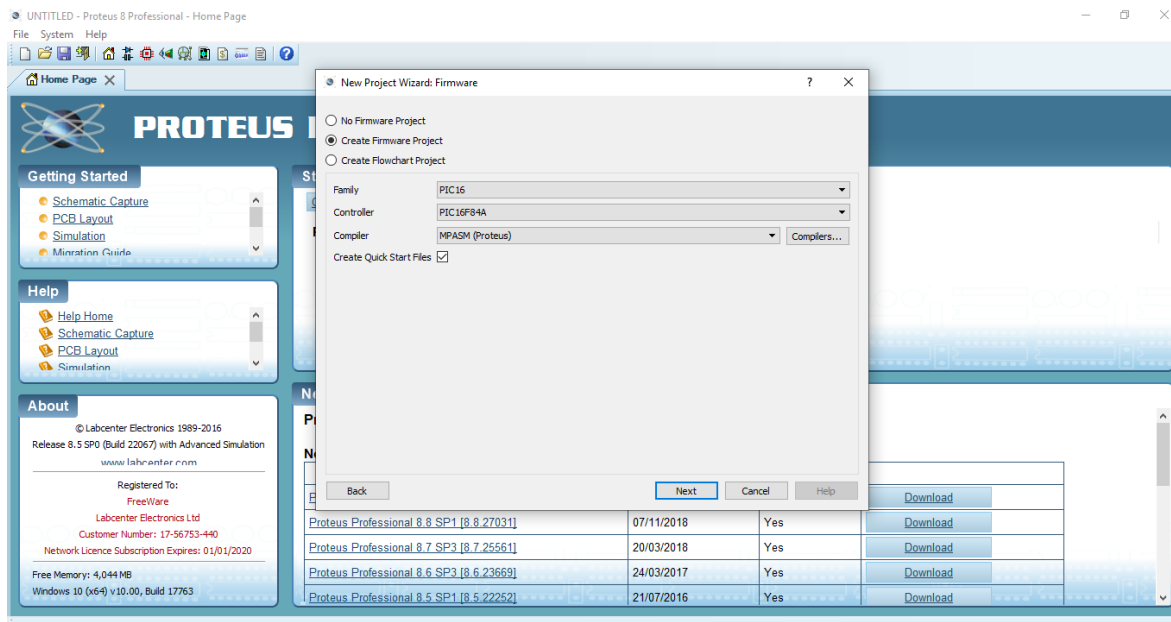
Se dio origen a un nuevo proyecto en la aplicación



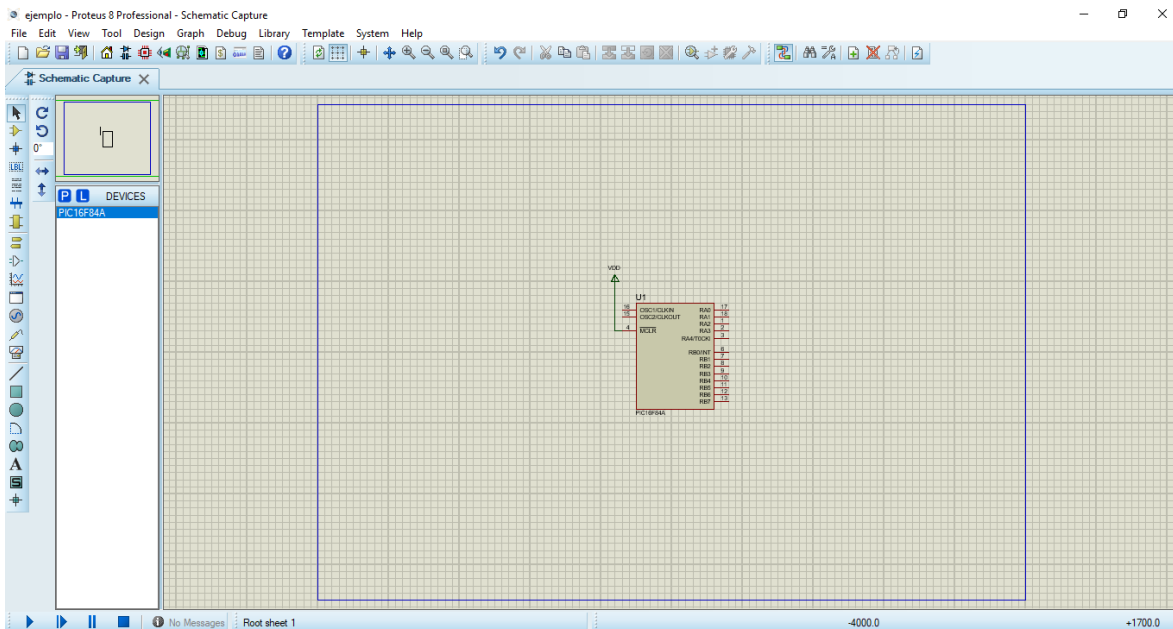
Se le agino un nombre al diagrama que se deseaba utilizar



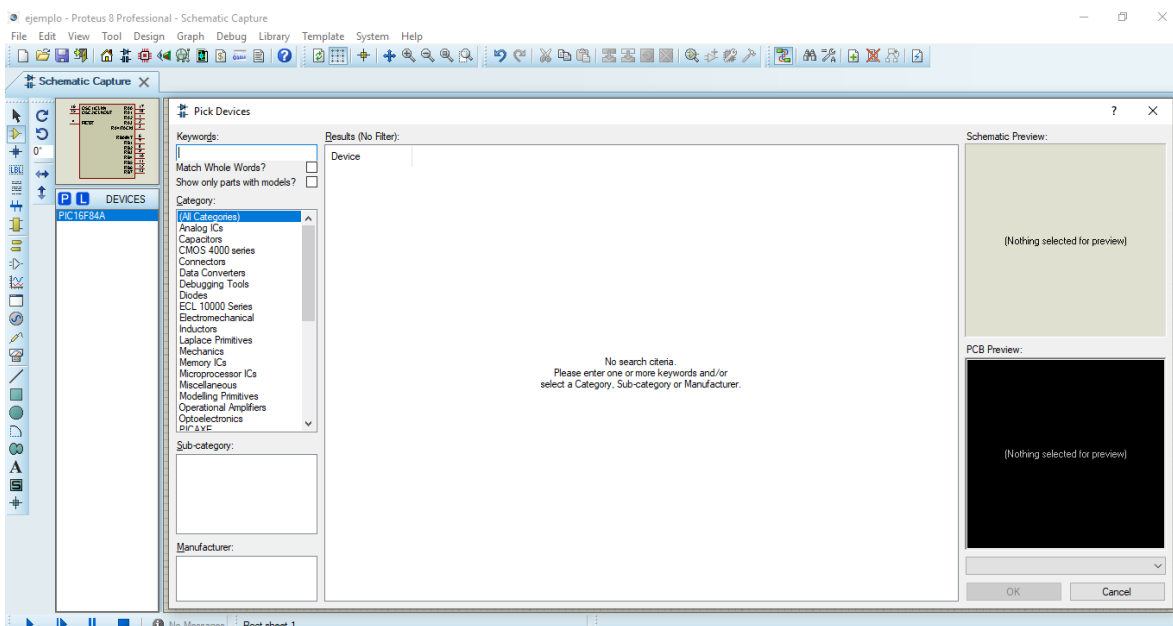
En esta parte de asigno el nombre y tipo de PIC que se utilizaría para la elaboración del proyecto, en este caso el PIC16F84A



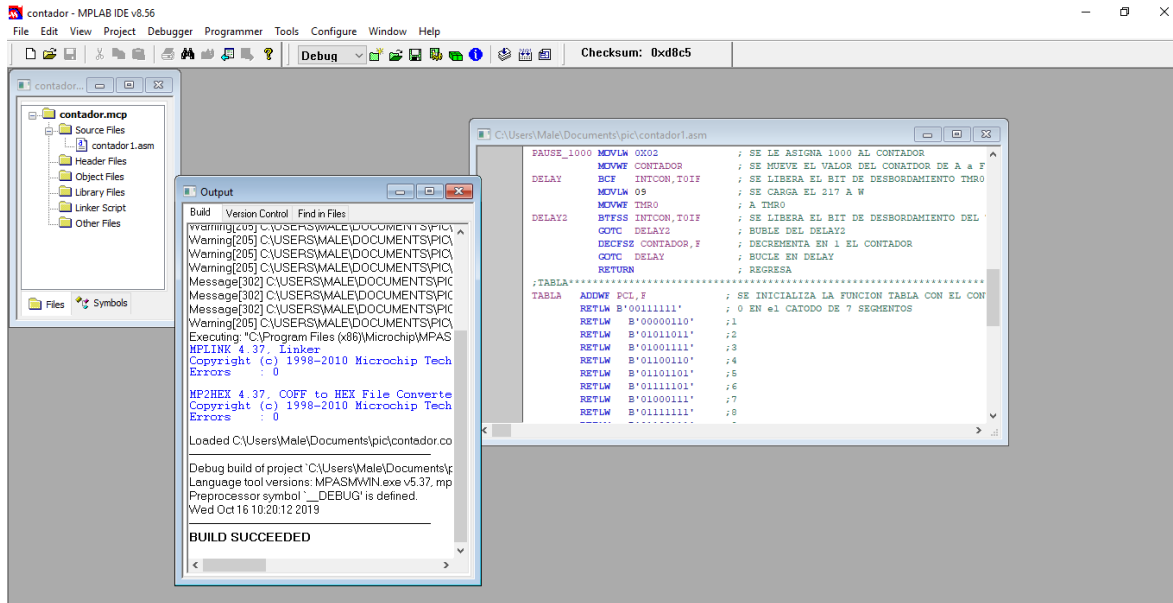
Se inserto el tipo de PIC16F84A



A este se le indico que se le agregarían materiales para el previo funcionamiento en la tabla protoboard



Tras la elaboración del código en lenguaje ensamblador, se realizó la compilación del diagrama en MPLAB y el código elaborado en PORTEUS 8 PROFESIONAL.



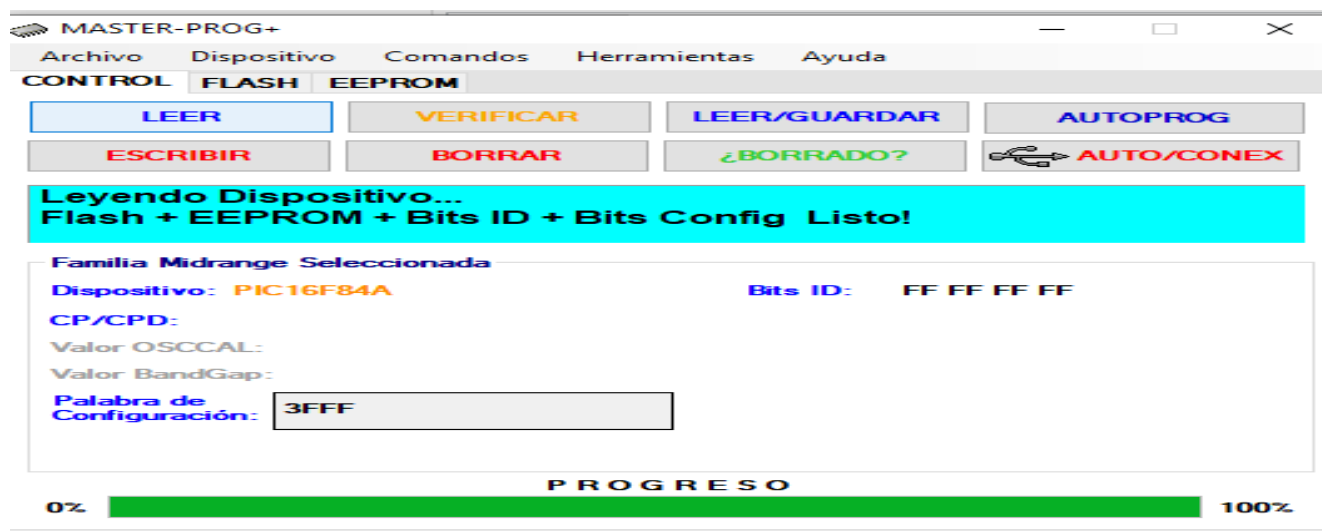
PROGRAMACIÓN DE PIC16F84A

Como principal instrucción se conectó el PIC16F84A al programador conectado de modo USB a la computadora que ejecuto el software indicado para la previa programación del PIC

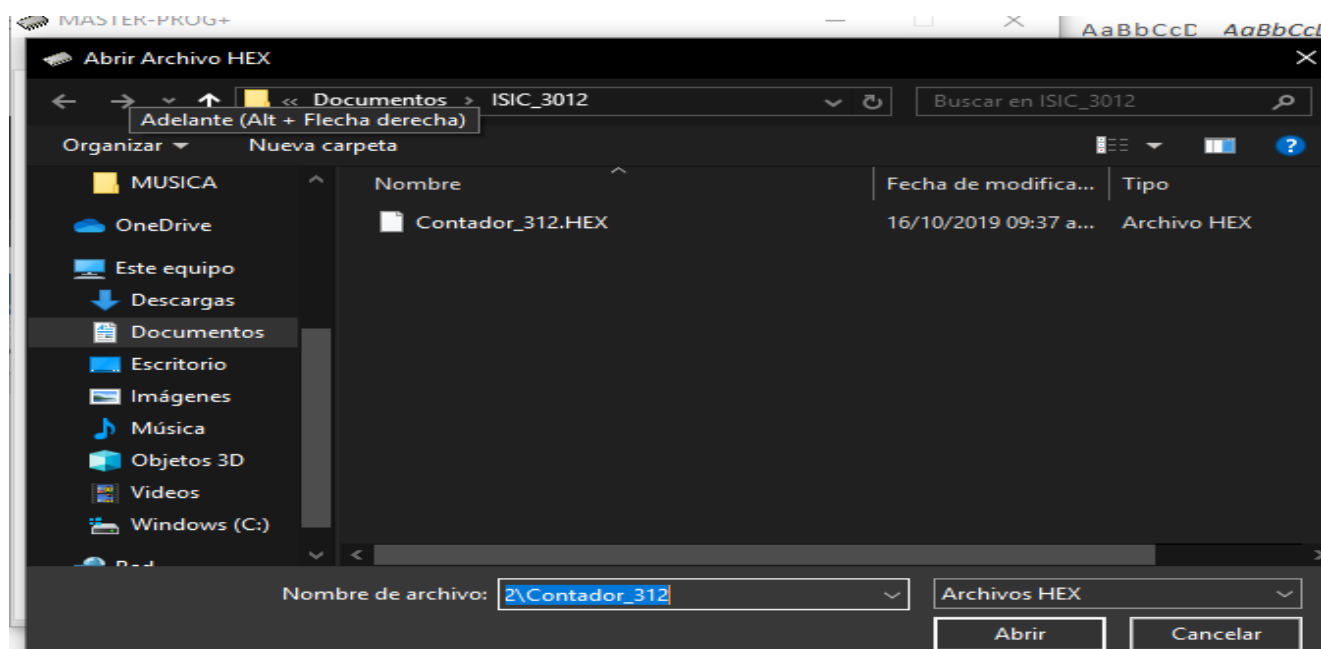




Se dio reconocimiento del PIC y previamente instrucción al programador de la lectura de este



Se abrió la ruta de acceso para poder programarle el código requerido para la ejecución de PIC, guardado como contador_312 hex, el cual fue compilado previamente para su utilización



En esta parte se le cargo el código necesario al PIC para previamente ejecutarlo, al analizarlo el programador indico que el archivo HEX se había cargado correctamente





El programador dio el reconocimiento de la correcta escritura del código previo a ejecutar



DIAGRAMA FISICO DE LA COMPILACION DE MPLAB Y PORTEUS

Para la elaboración del diagrama físico de la compilación de MPLAB y PORTEUS, se requirió conocer el DATA SHEET del PIC para conocer las ranuras en las cuales se llevarían a cabo las conexiones de cada uno de los materiales utilizados para su elaboración y ejecución.

MICROCHIP PIC16F84A

18-pin Enhanced FLASH/EEPROM 8-Bit Microcontroller

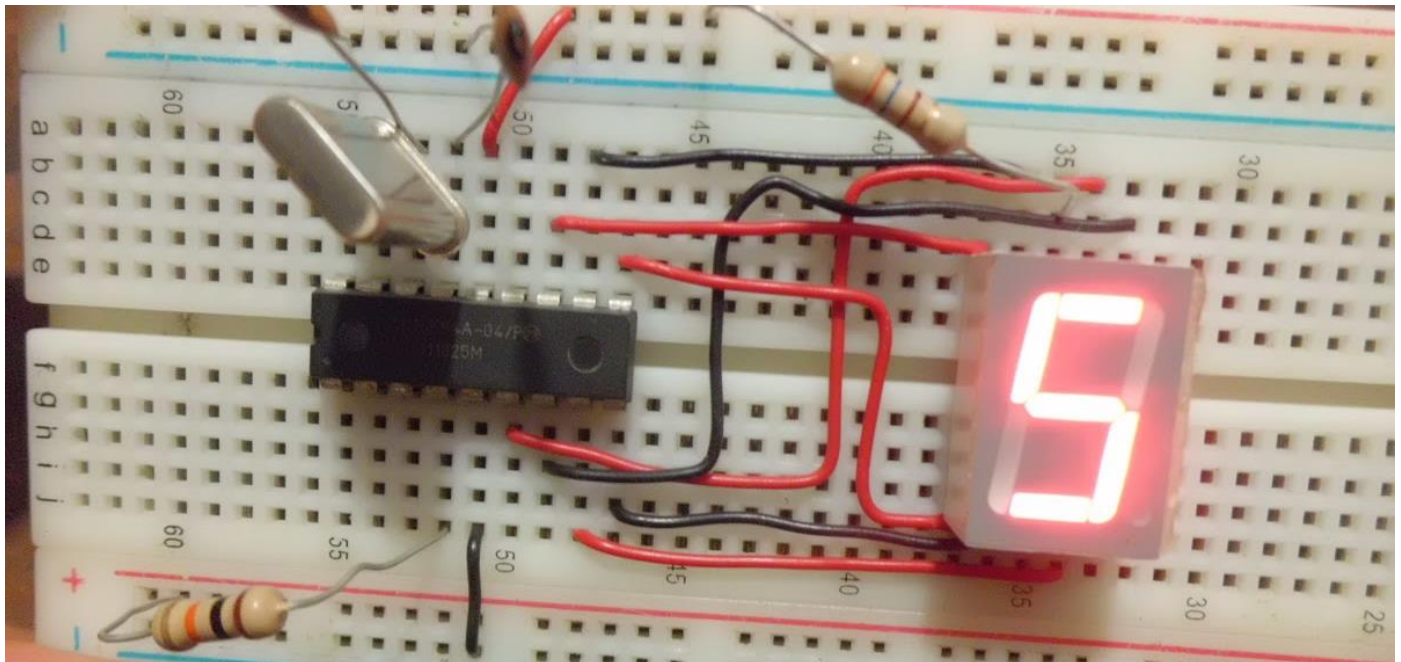
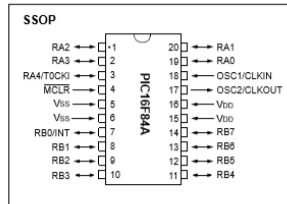
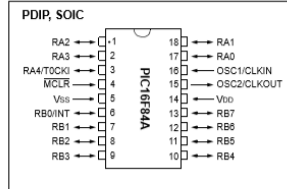
High Performance RISC CPU Features:

- Only 35 single word instructions to learn
- All instructions single-cycle except for program branches which are two-cycle
- Operating speed: DC - 20 MHz clock input
DC - 200 ns instruction cycle
- 1024 words of program memory
- 68 bytes of Data RAM
- 64 bytes of Data EEPROM
- 14-bit wide instruction words
- 8-bit wide data bytes
- 15 Special Function Hardware registers
- Eight-level deep hardware stack
- Direct, indirect and relative addressing modes
- Four interrupt sources:
 - External RB0/INT pin
 - TMR0 timer overflow
 - PORTB<7-4> interrupt-on-change
 - Data EEPROM write complete

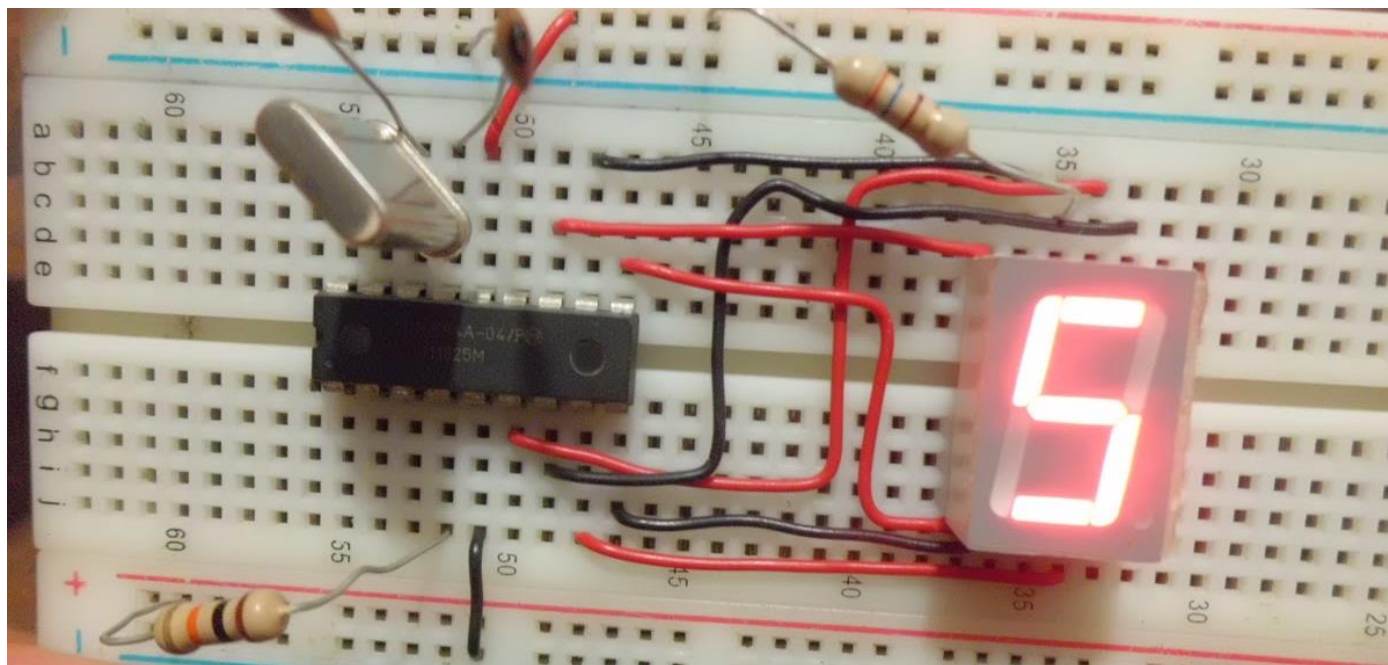
Peripheral Features:

- 13 I/O pins with individual direction control
- High current sink/source for direct LED drive
 - 25 mA sink max. per pin
 - 25 mA source max. per pin
- TMR0: 8-bit timer/counter with 8-bit programmable prescaler

Pin Diagrams



Dicha imagen muestra la culminación del diagrama físico del PIC16F84A en lenguaje ensamblador con el sistema hexadecimal



V. Conclusiones:

La elaboración de esta practica ayuda a tener un conocimiento mas amplio y practico de la utilización del PIC16F84A y esta no pudo haberse llevado acabo sin los previos conocimientos de lenguaje ensamblador y el sistema hexadecimal, para ello se debe tener conocimiento de alto nivel en cuestión de lenguajes de programación y los materiales utilizados para dicha práctica.