



INSTITUTO POLITÉCNICO NACIONAL



CENTRO DE ESTUDIOS CIENTÍFICOS Y TECNOLÓGICOS No. 9  
JUAN DE DIOS BÁTIZ

**Sistemas Digitales Microelectrónica Programable.**

Avance de Proyecto Aula “Capsule Automatic Dispenser”  
**(Tercer departamental)**

Profesor: Olivares Vargas Jesús Alberto

Grupo: 6IM3

Alumnos:

- Alvarado Rodríguez Fernando Bryan
- Ayala Bernal Israel
- Barrera Castillo Erick Abraham
- Hernández Caballero Mauricio
- Sánchez Martínez Felipe
- Villegas Monroy Emilio

Fecha de entrega: 09-06-2021

## **Objetivo**

Que los alumnos participantes hagan constancia de los avances obtenidos durante el 6to semestre de la carrera de sistemas digitales en el proyecto que se encuentren realizando. Tal constancia se elaborará con base en la estructura de una práctica de Microelectrónica programable llevando un marco teórico, donde se definirán algunos conceptos para facilitar la lectura del documento, un apartado práctico donde se harán algunas mediciones y pruebas de los circuitos como comprobación de su funcionamiento, además del anexo de códigos y diagramas esquemáticos usados para el proyecto, así como una conclusión general.

## **Capsule Automatic Dispenser (CAD)**

### **Eje temático.**

“Cuidado de la salud dentro y fuera del hogar mediante el uso de la electrónica”

### **Diagnóstico.**

El proyecto inicialmente tenía la característica de ser únicamente portátil y ser fácilmente operable, sin embargo, con el tiempo hemos ido realizando mejoras y anexos a lo que refiere del proyecto, por lo que también tenemos la idea de que sea compatible para operar dentro del hogar de manera permanente.

Esto nos permite abarcar una mayor cantidad de lugares en los que será posible colocar el CAD, desde el mismo cuerpo del paciente para el caso del equipo portátil como usarlo en la muñeca a modo de pulsera o un aparato que se lleve centro de la bolsa de la ropa o una de mano; hasta la cocina o habitaciones de una casa común para el caso del equipo fijo.

### **¿En qué consiste nuestro proyecto?**

El proyecto CAD significa “Capsule Automatic Dispenser” el cual se centra en la creación de un dispensador de pastillas que facilite las necesidades del consumidor o paciente con relación a los horarios de consumo de éstas; incluyendo implementos visuales y auditivos para mejorar el sistema de aviso de horarios.

Para todo esto, realizamos una investigación bastante completa sobre aquello que abarcaba este producto, llegando a la conclusión de que era y sigue siendo una buena opción dentro del mercado debido a los aditamentos visuales y auditivos que se aplicaron a ambos modelos del proyecto. También durante este semestre hemos estado implementando por nuestra cuenta un sistema de control de ubicación e intentamos que también sea un complemento para el proyecto CAD, dando mejoras notorias respecto a lo que se comúnmente en el mercado.

Lo que hace el proyecto, de forma general el dispensar pastillas cada cierto tiempo de forma programable. Ahora, siendo más específicos, el proyecto consta de varios elementos que hacen posible este proceso de dispensación. El reloj, que se pretende incluir en los dos modelos (estacionario como portátil) del proyecto para llevar un control del tiempo dentro del horario de ingesta de las pastillas o medicamentos; un sistema de alarma que sea disparado cada vez que el sistema detecte que el paciente no ha ingerido el medicamento o haya olvidado desactivar la misma al ingerirlos; el sistema de control de servomotores que dispensan las pastillas en las cantidades adecuadas controlados con una combinación entre el sistema de reloj y de alarma para que queden sincronizados y se eviten fallas en el sistema; un sistema de ubicación remota para pacientes que requieran cuidados intensivos o de vigilia constante y como complementos actuales, sistemas de medición de temperatura y oxigenación en la sangre si el paciente lo desea para este periodo de pandemia.

## ¿Qué beneficios se obtienen de la realización del proyecto?

La realización de este proyecto en específico ayudaría a elevar la compra de estos dispensadores debido a la diferencia de precios en el mercado comparándolos a otros dispensadores, así mismo, al tener las dos formas de modelo (estacionario y portátil) y ser lo más sencillamente operable, es muy útil en la vida de personas que suelen estar en la calle o tener vidas activas respecto a casi cualquier ámbito. Los anexos que hemos implementado durante el desarrollo del proyecto serán de igual manera beneficiarios para aún más personas ya que abarcarán una mayor demanda de necesidades específicas de los pacientes, como un control de oxígeno, temperatura o de vigilancia como se mencionó antes.



## Nombre de Proyecto:

**Grupo:** 6IM3 **Tema:** Tecnología en el hogar con Sistemas Digitales **Eje Temático:** Ciencia, Tecnología y Educación

**Propósito:** El proyecto CAD significa “Capsule Automatic Dispenser” el cual se centra en la creación de un dispensador de pastillas que facilite las necesidades del consumidor o paciente con relación a los horarios de consumo de éstas; incluyendo implementos visuales y auditivos para mejorar el sistema de aviso de horarios.

**Impacto Social:** La realización de este proyecto en específico ayudaría a elevar la compra de estos dispensadores debido a la diferencia de precios en el mercado comparándolos a otros dispensadores, así mismo, al tener las dos formas de modelo (estacionario y portátil) y ser lo más sencillamente operable, es muy útil en la vida de personas que suelen estar en la calle o tener vidas activas respecto a casi cualquier ámbito. Los anexos que hemos implementado durante el desarrollo del proyecto serán de igual manera beneficiarios para aún más personas ya que abarcarán una mayor demanda de necesidades específicas de los pacientes, como un control de oxígeno, temperatura o de vigilancia como se mencionó antes.



## **Conceptos y especificaciones del proyecto**

**Dato:** es cualquier conjunto de carácter

**Escritura:** Es “Escribir” un dato en un registro o una microinstrucción en nuestra memoria.

**Lectura:** Es “leer” el dato previamente guardado en el registro.

**Microinstrucción:** es un simbolismo que representa la manera de ejecutar una acción en el sistema basado en registros.

**Motor a pasos:** El motor paso a paso conocido también como motor de pasos es un dispositivo electromecánico que convierte una serie de impulsos eléctricos en desplazamientos angulares discretos, lo que significa que es capaz de girar una cantidad de grados dependiendo de sus entradas de control.

**Microprograma:** Conjunto de microinstrucciones

**NOP:** No operación, significa que durante ese tiempo no se ejecutará ninguna microinstrucción en el circuito, por lo tanto, no estará haciendo algo

**NC:** No conexión, solamente es dejar sin conectado esa terminal

**OE:** Es el output enable, el cual al conectar a 0 deja pasar el dato en sus terminales de salida, si no solo lo deja en alta impedancia

**Canal de comunicación de información:** Eléctricamente, es aquel que transporta bits, donde todos los registros están conectados a sus salidas y entradas correspondientemente

**Unidad de control:** Se encarga de generar todas las señales para ejecutar una microinstrucción

**Líneas de dirección:** Indican el registro en donde se almacenará la información, en el mismo sistema.

**Lenguaje maquina:** Se refiere al lenguaje que puede entender un ordenador el cual está basado en binario, pero para hacerlo más práctico este se usa en hexadecimal.

**Memoria de programa:** Memoria de solo lectura

**Subrutina:** Es otro diagrama de bloques semejante, utilizado en el programa principal

**Programa fuente:** Es un archivo tipo texto que contiene el programa de mi aplicación a resolver.

**Buffer:** Colección de registros en la Ram utilizados para un fin común.

**Menú:** Ventana donde se muestran aplicaciones del programa.

**Canal de comunicación de información:** Eléctricamente, es aquel que transporta bits, donde todos los registros están conectados a sus salidas y

**CPU:** Unida de Control de Procesos

**Pila:** En un sistema de microprocesadores es una colección de registros o localidades de memoria, organizados de la forma en que el último dato es el primero en salir.

**Subrutina de pérdida de tiempo:** Como su nombre lo dice es una subrutina encargada de perder el tiempo del microcontrolador para diversos propósitos, el microcontrolador al conectarlo a un reloj de cuarzo de 4MHz generara que cada instrucción la ejecutara en 1 $\mu$ s y los brincos en 2 $\mu$ s, por lo que la subrutina de pérdida de tiempo no es nada más que varias instrucciones para perder tiempo.

**Interrupción:** Se puede definir como una señal proveniente de un dispositivo externo, que llega a una entrada del microprocesador dedicada a este propósito y que le indica al microprocesador que el dispositivo que la originó está solicitando servicio.

**Contador:** cuenta los eventos externos (a través del pin RA4/T0CKI).

**Temporizador:** Cuenta los pulsos internos de reloj.

**Polarizar:** Poner las terminales a un potencial eléctrico.

**Alfanumérico:** es un término colectivo que se utiliza para identificar letras del alfabeto latino y de números arábigos.

**Animación:** es un proceso, utilizado por uno o más animadores, para dar la sensación de movimiento a imágenes, dibujos u otro tipo de objetos inanimados.

**Pixel:** Un píxel o pixel, es la menor unidad homogénea en color que forma parte de una imagen digital.

**Memoria:** Una memoria es un medio de almacenamiento de información permanente o semipermanente, la información almacenada posteriormente será recuperada.

**RAM:** La Memoria de Acceso Aleatorio o RAM (Random Access Memory) es donde la CPU (Central Processing Unit) almacena los datos que está utilizando en el momento presente. Una memoria RAM es un chip de almacenamiento temporal por lo que se dice que es volátil; si se corta el suministro de energía la memoria pierde la información.

**EEPROM:** (Electrically Erasable PROM) ó EEPROM conserva la estructura de compuerta flotante de la EPROM, pero con una inclusión de una región muy delgada encima del electrodo de drenaje de la celda de memoria MOS. Esta modificación es la principal característica de la EEPROM para facilitar el borrado eléctrico.

**Flash:** La memoria FLASH (instantánea o ráfaga) se trata de una memoria no volátil, de bajo consumo, que se puede programar en campo al igual que la EEPROM. La memoria FLASH es una modificación de la EEPROM para lograr densidades y costos más cercanos a las EPROMs,

**Algoritmo:** Especificación paso a paso de un problema dado, el cual puede ser representado en cualquier lenguaje o símbolo y deberá tener un numero finito de pasos.

**Barrido en el teclado:** Esto hace referencia a que pondrá el programa constantemente un 0 en el renglón de nuestro teclado, con el fin de conocer la localización exacta de una tecla, recordar que previamente se necesitara poner pull-ups en el puerto B para evitar poner resistencias y conexiones a Vcc dentro del circuito.

**LCD 16x2:**



Es un dispositivo con pantalla de cristal líquido que cuenta con dos filas, de diecisésis caracteres c/u, que se utiliza para operaciones de lectura, por lo general alfanumérica. Las capacidades de estos dispositivos son altas, pues se puede mostrar todo tipo de información sin importar qué tipo de símbolos o caracteres sean, el idioma o el lenguaje, pues el sistema puede mostrar cualquier carácter alfanumérico, símbolos y algunas figuras, el número de píxeles que

tiene cada símbolo o carácter varía dependiendo del modelo del dispositivo y cada artefacto está controlado por un microcontrolador que está programado para dirigir el funcionamiento y la imagen mostrada en la pantalla.

7.3 character generator code map (Please refer to ST7066U datasheet for other character code map)																
A7-A4	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
CG RAM (1)	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
	0001 (2)	! 1 A Q a q							■ ア チ ハ a q							
	0010 (3)	" 2 B R b r							「 イ ツ ハ β β							
	0011 (4)	# 3 C S c s							■ ウ テ モ β β							
	0100 (5)	\$ 4 D T d t							× エ ド ハ p β							
	0101 (6)	% 5 E U e u							・ オ ナ ュ 6 6							
	0110 (7)	& 6 F U f v							ヲ カ ニ ョ ピ ピ							
	0111 (8)	* 7 G U g w							ア キ フ ラ ピ ピ							
	1000 (1)	( 8 H X h x							イ ク ネ リ 5 5							
	1001 (2)	) 9 I Y i g							ウ ケ リ ル 7 7							
	1010 (3)	*: J Z j z							エ コ ハ レ j 7							
	1011 (4)	+; K C k {							オ サ ハ ロ 8 8							
	1100 (5)	, < L # 1							ト ヲ フ ワ ♦ ♦							
	1101 (6)	- = M J m }							ユ ス ハ ノ ム ノ							
	1110 (7)	. > N ^ n *							ミ ハ ノ ^ ノ							
	1111 (8)	/ ? O _ o *							ミ ハ ノ ^ ノ							

Entre las ventajas que se tienen al utilizar este tipo de artefactos están el mínimo consumo de energía o corriente eléctrica, la programación es sumamente sencilla y es por lo general cargada por el fabricante, dependiendo del uso y tipo de equipo que se trate.

Estos módulos de pantalla de cristal líquido tienen usos muy variados y en diferentes campos como la informática, las telecomunicaciones, la telefonía, la industria automotriz, la relojería, la robótica, entre muchas otras.

Las pantallas de cristal líquido suelen disponer de una matriz de 5x8 puntos que se utiliza para representar los diferentes caracteres. Comúnmente, el total de caracteres que se pueden representar en dispositivo es de 256, de los cuales los primeros 240 son los símbolos alfanuméricos comunes (letras mayúsculas y minúsculas, signos de puntuación, algunos caracteres especiales, comunes y números) y el resto son caracteres que pueden ser introducidos o definidos por el usuario.

Los dispositivos LCD 16x2 deben contar con un controlador, el más común que se utiliza en la industria es el chipset HD44780, sobre todo por su flexibilidad, soporte y compatibilidad con prácticamente cualquier plataforma. Existen, sin embargo, otros controladores bastante efectivos, la mayoría desarrollados a partir del controlador HD44780.

En cuestiones de programación y recursos informáticos, estos dispositivos requieren de 6 pines del microcontrolador para su conexión, los caracteres con los que se programe el display deben estar en código ASCII. Dependiendo del microcontrolador que se utilice, los dispositivos LCD 16x2 pueden conectarse a través de un bus de datos de 4 u 8 bits, dependiendo del número de pines a disposición y son compatibles con diversas plataformas como Arduino, PIC, MSP430, AVR, entre otras.

La LCD 16x2 quiere decir que tendrá lugar para poner 16 caracteres en dos renglones, si fuera 32x2, podría poner 32 caracteres y dos renglones, ósea 64 caracteres en total tanto en el renglón uno como en el dos.

También funciona con comandos la LCD algunos son:

Más que nada la LCD nos ayudará para ver el mensaje que queremos transmitir.

comando	función	hexa
clear	limpia display	01h
home	posiciona cursor al inicio del lcd	03h
cursor	muestra cursor con parpadeo	0fh
8 bits	selecciona interfaz de 8 bits	38h
renglon1	selecciona escritura en el primer renglón	80h
renglon2	selecciona escritura en el segundo renglón	c0h

## Teclado Matricial 4x4

- A) Teclado Paralelo. Se requieren muchos recursos de hardware, o sea, muchos puertos.
- B) Teclado Matricial. Se requieren menos puertos en comparación del paralelo además de que en este arreglo las salidas del teclado serán por la parte inferior de este.

El Teclado matricial de botones plásticos formado por 4 filas y 4 columnas para un total de 16 teclas permite agregar una entrada de usuario a tus proyectos. El teclado es de tipo membrana, por lo que entre sus ventajas se encuentra el poco espacio que requiere para ser instalado. Posee una cubierta adhesiva y un cable flexible de conexión. Puede ser conectado a cualquier microcontrolador o tarjetas de desarrollo como Arduino.

El teclado matricial 4x4 está formado por una matriz de pulsadores dispuestos en filas (L1, L2, L3, L4) y columnas (C1, C2, C3, C4), con la intención de reducir el número de pines necesarios para su conexión. Las 16 teclas necesitan sólo 8 pines del microcontrolador en lugar de los 16 pines que se requerirían para la conexión de 16 teclas independientes. Para poder leer que tecla ha sido pulsada se debe de utilizar una técnica de barrido y no solo leer un pin de microcontrolador.

La conexión del teclado matricial 4x4 con Arduino u otra plataforma de microcontroladores es simple: se necesitan 8 pines digitales en total. Puede trabajar con microcontroladores de 3.3V o 5V sin problema. Es necesario colocar resistencias pull-up entre los pines de las columnas y VCC o activar por software las resistencias pull-up internas en el Arduino. En cuanto a la programación, la lectura de las teclas se debe realizar mediante un "barrido" de las filas. Si bien es posible realizar este procedimiento dentro del loop principal del programa, es una mejor práctica realizar el barrido utilizando interrupciones por TIMER y así asegurar la lectura de las teclas en un intervalo conocido y exacto, además de dejar al loop libre para realizar otras operaciones.

Cuenta con:

- 16 botones con organización matricial (4 filas x 4 columnas)
- Tiempo de rebote (Bounce time): ≤5 ms
- Máximo voltaje operativo: 24 V DC
- Máxima corriente operativa: 30 mA
- Resistencia de aislamiento: 100 MΩ (100 V)
- Voltaje que soporta el dieléctrico: 250 VRMS (60Hz, por 1 min)
- Dimensiones teclado: 69\*77mm
- Cable de cinta plana de 8.5 cm de largo aprox. (incluido el conector)
- Conector tipo DuPont hembra de una fila y 8 contactos con separación estándar 0.1" (2.54mm)
- Temperatura de operación: 0 a 50 °

Finalmente, su uso es en: sistemas de seguridad, selección de menús, ingreso de datos, etc.

**PIC16F877A:** Veremos ahora las características que posee este microcontrolador, estos datos son obtenidos del datasheet PIC16F87X, la hoja principal se muestra a continuación.

## 28/40-pin 8-Bit CMOS FLASH Microcontrollers

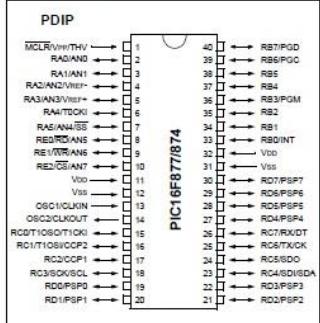
### Devices Included in this Data Sheet:

- PIC16F873
- PIC16F876
- PIC16F874
- PIC16F877

### Microcontroller Core Features:

- High-performance RISC CPU
- Only 35 single word instructions to learn
- All single cycle instructions except for program branches which are two cycle
- Operating speed: DC - 20 MHz clock input DC - 200 ns instruction cycle
- Up to 8K x 14 words of FLASH Program Memory.
- Up to 368 x 8 bytes of Data Memory (RAM)
- Up to 256 x 8 bytes of EEPROM data memory
- Pinout compatible to the PIC16C73B/74B/76/77
- Interrupt capability (up to 14 sources)
- Eight level deep hardware stack
- Direct, indirect and relative addressing modes
- Power-on Reset (POR)
  - Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code-protection
- Power saving SLEEP mode
- Selectable oscillator options
- Low-power, high-speed CMOS FLASH/EEPROM technology
- Fully static design
- In-Circuit Serial Programming™ (ICSP) via two pins
- Single 5V In-Circuit Serial Programming capability
- In-Circuit Debugging via two pins
- Processor read/write access to program memory
- Wide operating voltage range: 2.0V to 5.5V
- High Sink/Source Current: 25 mA
- Commercial and Industrial temperature ranges
- Low-power consumption:
  - < 2 mA typical @ 5V, 4 MHz
  - 20 µA typical @ 3V, 32 kHz
  - < 1 µA typical standby current

### Pin Diagram



### Peripheral Features:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during sleep via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- 10-bit multi-channel Analog-to-Digital converter
- Synchronous Serial Port (SSP) with SPI™ (Master Mode) and I<sup>2</sup>C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) 8-bits wide, with external RD, WR and CS controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out reset (BOR)

- I) Contiene un bloque llamado (POR) que significa Reset al encendido.
- m) Contiene un bloque llamado (PWRT) y (OST), los cuales tienen la función de temporizar el arranque del microcontrolador, y controlar el tiempo de encendido del oscilador.
- n) Contiene un (WDT) que significa perro guardián, el cual opera con un oscilador interno tipo RC.
- o) Se puede proteger el programa contra lecturas, y esta función es programable.
- p) Posee un modo SLEEP “Dormir” para el ahorro de energía.
- q) Se pueden seleccionar varios tipos de oscilador.
- r) Las tecnologías de las memorias FLASH/EEPROM son de bajo consumo de energía y de alta velocidad.

Con lo cual forman una familia de microcontroladores, casi con las mismas características.

Características del núcleo del microcontrolador.

- a) CPU con tecnología RISC de alto rendimiento.
- b) Solo 35 instrucciones de una sola palabra para aprender.
- c) Todas las instrucciones son de un ciclo, excepto los brincos en el programa los cuales son de dos ciclos.
- d) Velocidad de operación: 20 MHz y 4 MHz.
- e) La memoria de programa tiene una capacidad de 8K x 14 words, para la versión de 40 pines “PIC16F877A”.
- f) La memoria de datos “RAM” tiene una capacidad de 369 x 8 bytes.
- g) La memoria de datos “EEPROM” tiene una capacidad de 256 x 8 bytes.
- h) Función de pines compatibles con los PIC 16C73B/74B/76/77.
- i) Capacidad de interrupción “Mas de 14 fuentes”.
- j) Pila de hardware de ocho niveles de profundidad.
- k) Modos de direccionamiento: Directo, indirecto y relativo.
- s) Diseño totalmente estático.
- t) La programación se realiza de forma serial (ICSP) por dos pines.
- u) Capacidad de programación vía serial con una fuente de 5V.
- v) Contiene un circuito de depuración vía dos pines.
- w) En la memoria de programa del procesador se puede Leer/Escribir.
- x) Amplio rango de voltaje de operación: 2.0 V a 5.5 V.
- y) Alta corriente de entrada/Salida: 25 mA. Máximo por pin.
- z) Rangos de temperatura comercial e industrial.
- aa) Bajo consumo de energía <2 mA. A 5V y 4 MHz.

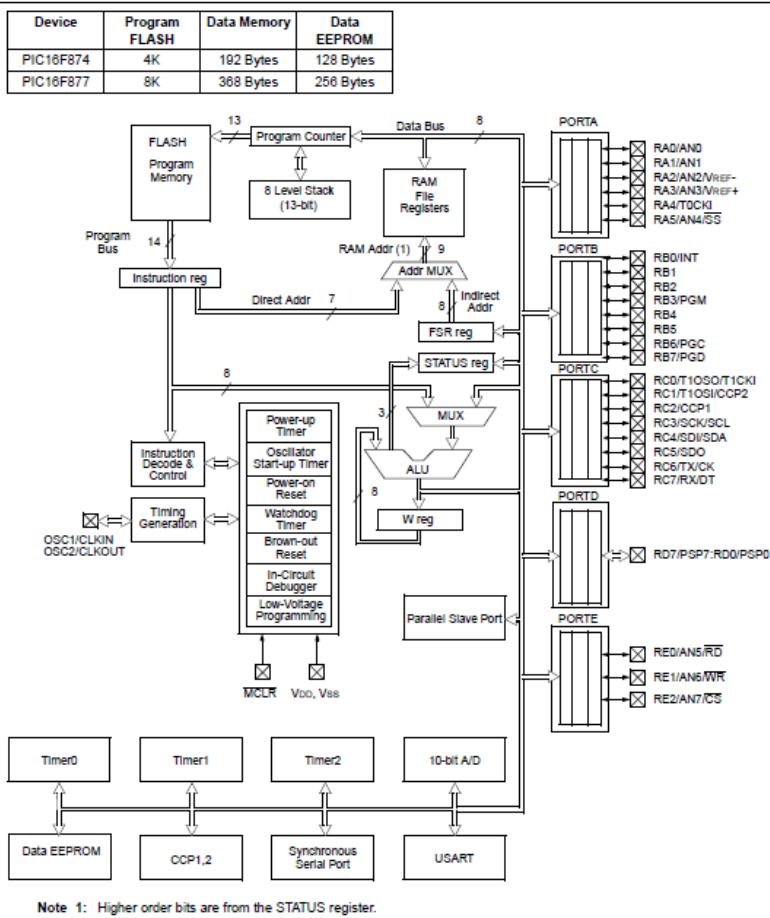
### Funciones de los periféricos:

- a) Timer0: 8 bits temporizador/Contador, con un prescalador de 8 bits.
- b) Timer1: 16 bits temporizador/Contador, con un prescalador, el cual puede ser incrementado durante el modo dormir vía un cristal de reloj externo.

- c) Timer2: 8 bits temporizador/Contador, con un registro generador de periodo, prescalador y postscalador.
- d) Posee dos módulos de captura, comparadores y módulos PWM con resolución de 10 bits.
- e) Convertidor analógico a digital multicanal de 10 bits.
- f) Puerto serial síncrono (SSP) con SPI (Modo maestro) y I<sup>2</sup>C (Maestro/Esclavo).
- g) Transmisor receptor síncrono asíncrono (USART/SCI) con detección de dirección de 9 bits.
- h) Puerto paralelo esclavo (PSP) de amplitud de 8 bits, con señales de control externas RD, WR y CS todas estas negadas solo para la versión de 40 pines.
- i) Circuito de detección de falla a tierra para el circuito de Reset por falla a tierra.

Diagrama de bloques del microcontrolador PIC16F877.

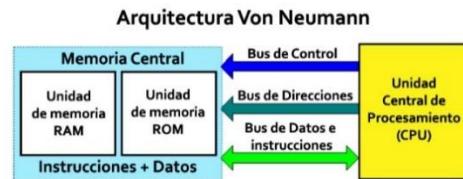
FIGURE 1-2: PIC16F874 AND PIC16F877 BLOCK DIAGRAM



La arquitectura interna de este microcontrolador es de tipo “Harvard”.



En los inicios de los microcontroladores el tipo de arquitectura interna era de tipo “Von Neumann”



Como podemos observar la arquitectura de Von Neumann se caracterizaba por tener una sola memoria principal donde los datos y las instrucciones se almacenaban indistintamente. Esto significaba que sólo había un sistema de bus de datos para direcciones, datos y control de la memoria.

En nuestro caso, estaremos trabajando con el de arquitectura Harvard, en cual se observan dos memorias independientes una de la otra, una que contiene sólo instrucciones y otra que datos. Cada una tiene su bus de acceso y eso da paso a que puedan realizarse operaciones de W/R al mismo tiempo en las dos memorias.

## Set de instrucciones del microcontrolador.

### Operaciones entre registros orientados a byte.

Mnemónico, Operandos	Ejemplo
ADDWF f, d Sumar W con Registro.	Addwf contador,1;
ANDWF f, d Función lógica and entre w y registro.	Andwf contador,1;
CLRF f Borra registro.	Clrf contador;
CLRW Borra W.	Clrw;
COMF f, d Complemento a 1 del registro.	Comf contador,1;
DECFSZ f, d Decrementa en uno registro.	Decfsz contador,1;
DECFSZ f, d Decrementa y brinca si resultado es 0.	Decfsz contador,1;
INCF f, d Incrementa en uno registro.	Incfsz contador,1;
INCF f, d Incrementa en 1 el registro y brinca si 0.	Incfsz contador,1;
IORWF f, d Función lógica or entre w y registro.	Iorwf contador,1;
MOVF f, d Carga W con registro.	Movf contador,1;
MOVWF f Carga registro con W.	Movwf contador;
NOP No operación solo pierde tiempo.	Nop;
RLF f, d Rota un bit a la izquierda por carry	Rlf contador,1;
RRF f, d Rota un bit a la derecha por carry	Rrf contador,1;
SUBWF f, d Resta el contenido de W con registro.	Subwf contador,1;
SWAPF f, d Intercambia nibbles en registro	Swapf contador,1;
XORWF f, d Función lógica or exclusiva entre w y registro.	Xorwf contador,1;

### Operaciones entre registros orientados a bit.

Mnemónico, Operandos	Ejemplo
BCF f, b Pon en cero el bit del registro	Bcf porta,.3;
BSF f, b Pon en uno el bit del registro	Bsf porta,.3;
BTFSZ f, b Prueba bit y brinca si 0.	Btfsz porte,.0;
BTFSZ f, b Prueba bit y brinca si 1.	Btfss porte,.0;

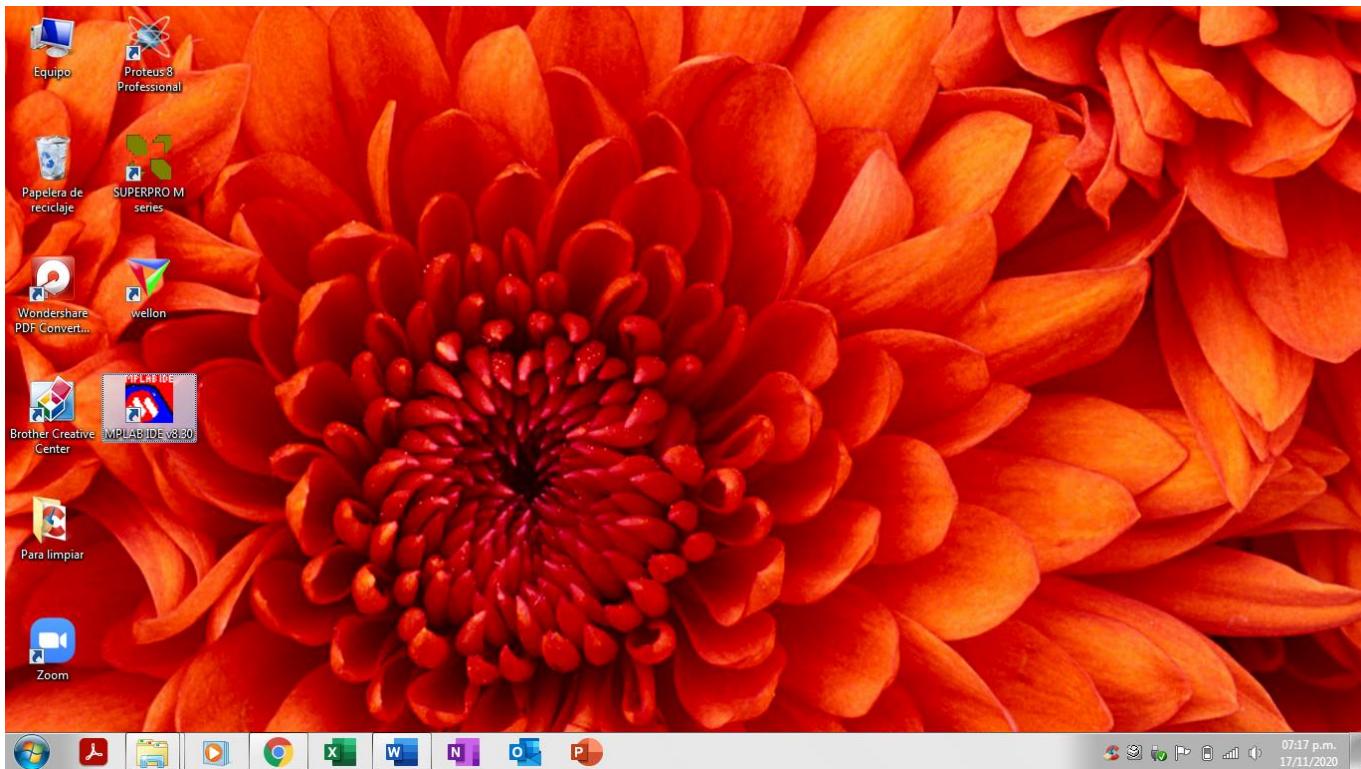
### Operaciones con Literales y Control.

Mnemónico, Operandos	Ejemplo
ADDLW k Suma la literal con el contenido de W.	Addlw 25h;
ANDLW k Función lógica and entre W y literal.	Andlw 0fh;
CALL k Llamada a subrutina.	Call retardo,
CLRWD T Reinicializa el perro guardián.	Clrwdt;
GOTO k Brinca a etiqueta "Dirección".	Goto prog_prin;
IORLW k Función lógica or entre W y literal.	Iorlw 0fh;
MOVLW k Carga w con literal.	Movlw 25h;
RETFIE - Regresa de la subrutina de interrupción.	Retfie;
RETLW k Regresa literal en W.	Retlw 34h;
RETURN - Regresa de la subrutina.	Return;
SLEEP - Coloca al microcontrolador en modo dormir.	Sleep;
SUBLW k Resta la literal con el contenido de W.	Sublw 10h;
XORLW k Función lógica or exclusiva entre W y literal.	Xorlw 34h;

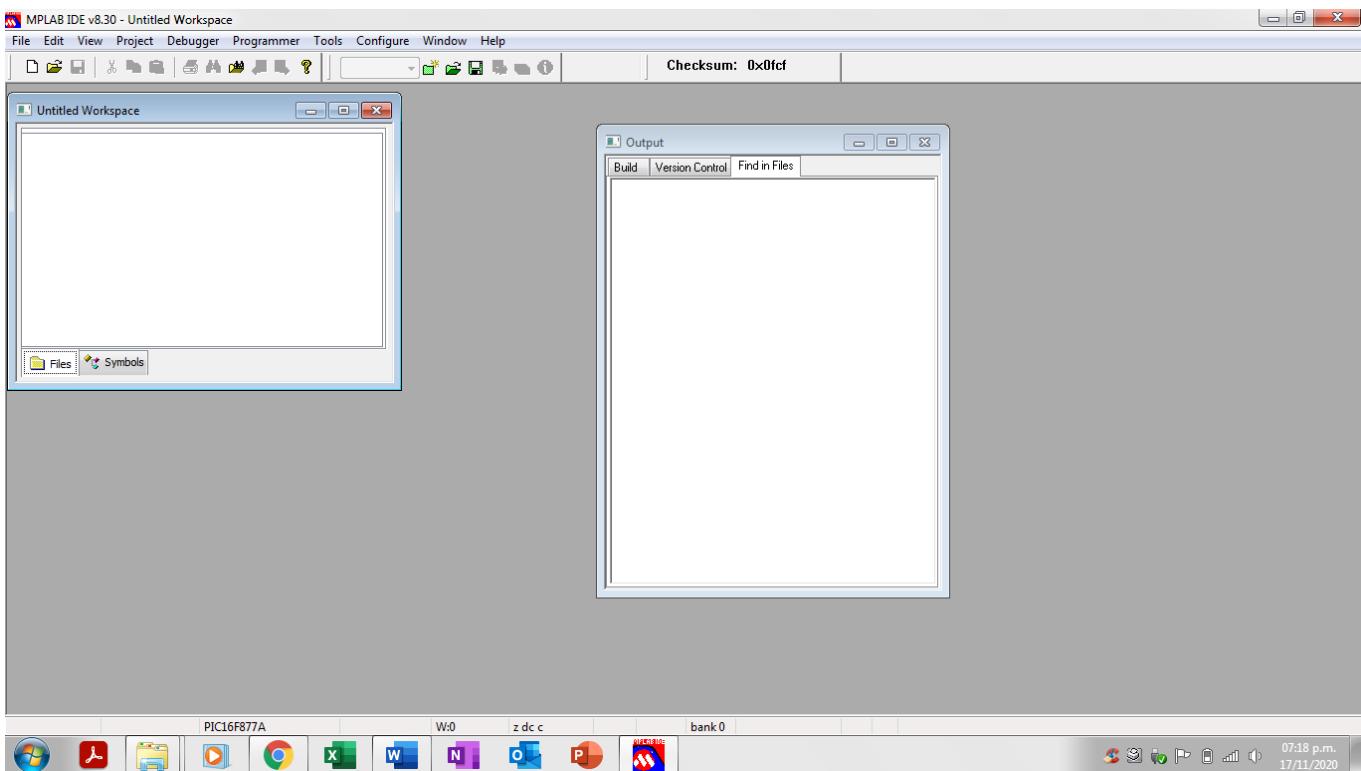
## MPLAB:

En el proceso de elaboración de esta práctica hemos de usar el software de MPLAB para la realización del programa que controlará los registros y que a su vez controlarán los displays. A continuación, explicaremos los pasos para la correcta ejecución del programa y la creación de un proyecto en el caso de esta práctica.

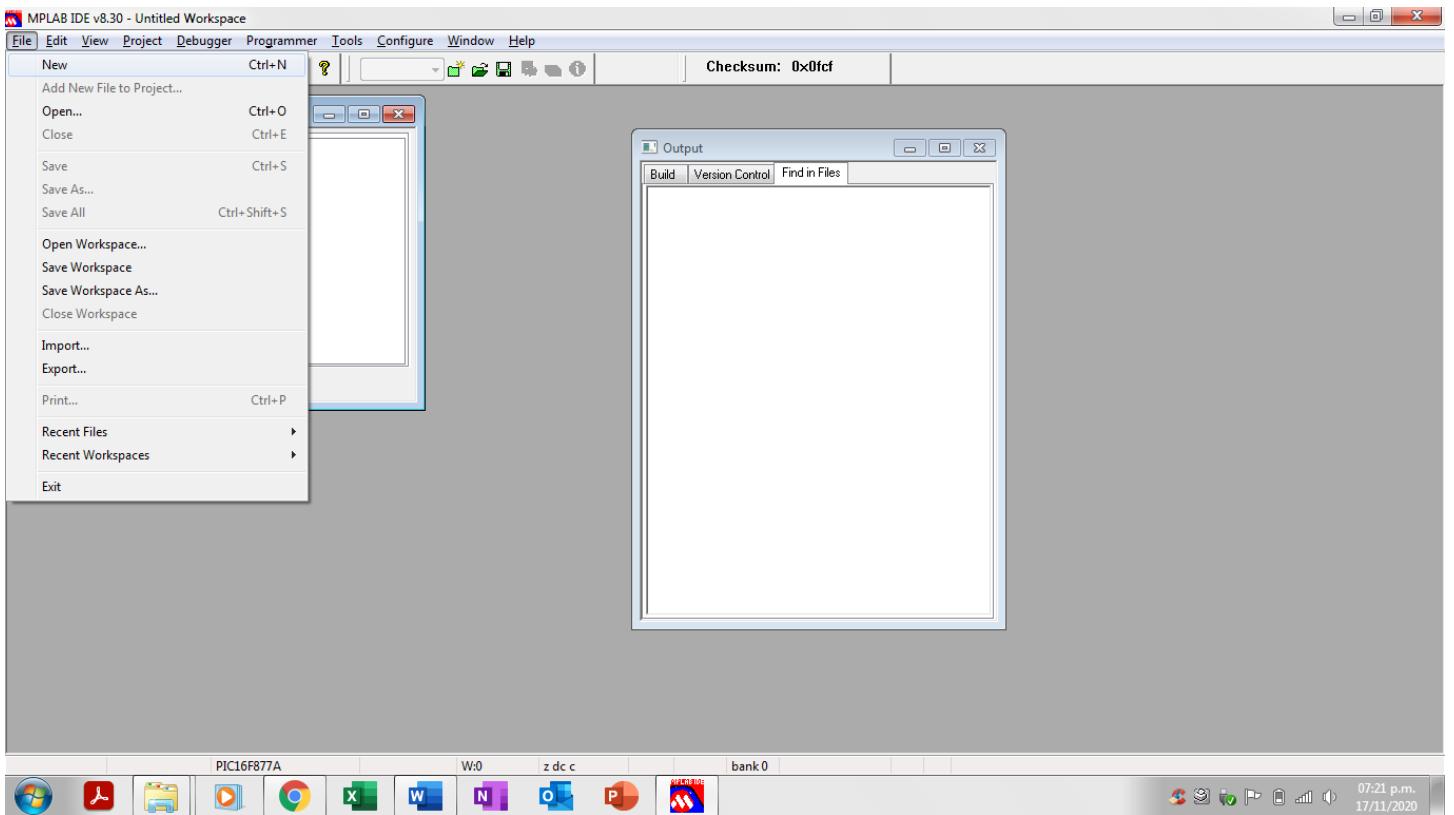
Comenzaremos con abrir la aplicación en el acceso directo que se ubica en el escritorio de la computadora



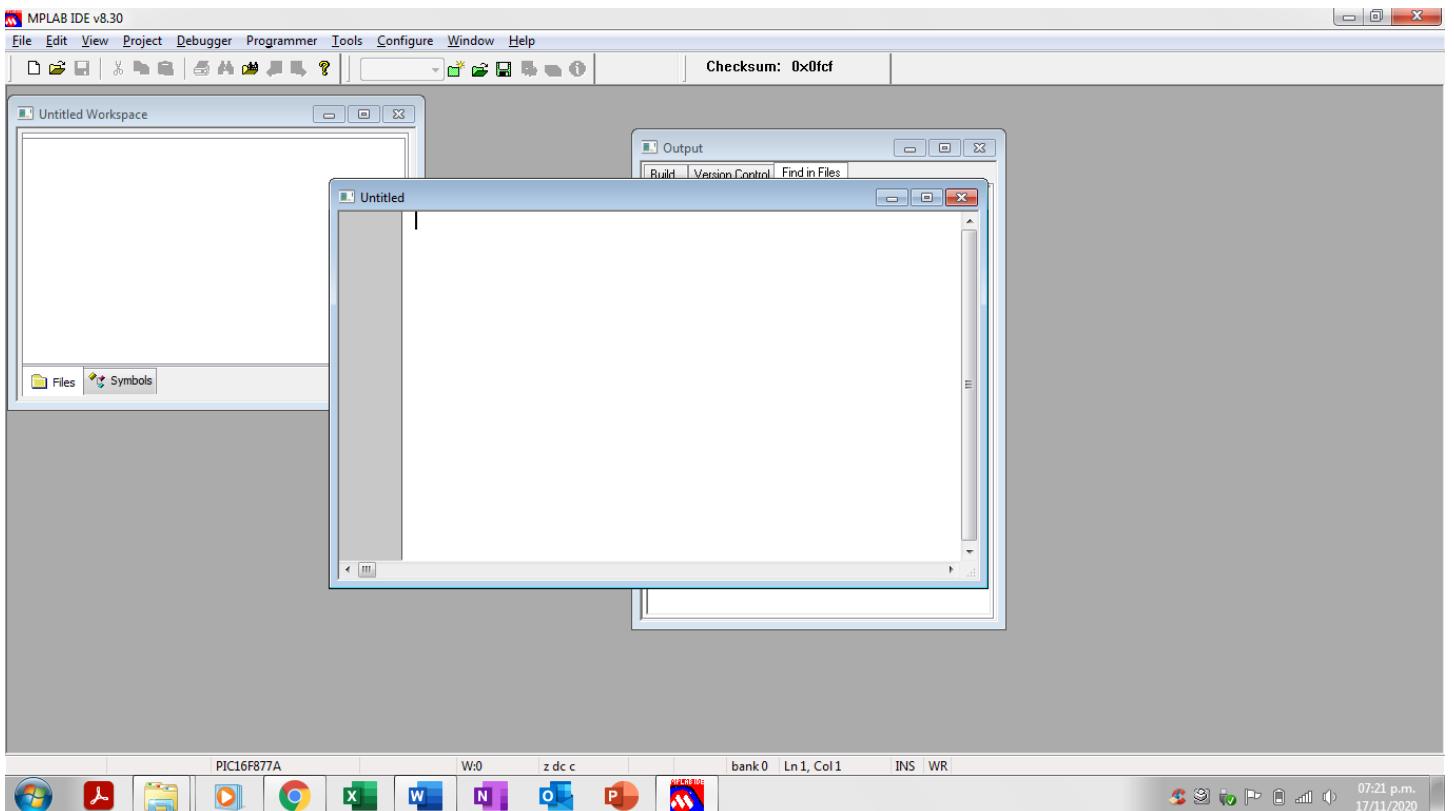
Nos mostrará una ventana como la siguiente:



En la parte superior se encuentra la barra de herramientas. Ahí seleccionamos la opción de “File” y damos click en “New”.



Se abrirá una ventana.



Este es un editor de texto y es donde escribiremos el programa con todas sus instrucciones.

En tanto hayamos transcritto el código, como aún no lo hemos guardado las letras están en un estado monocromático. Por lo que iremos a la herramienta de File y seleccionaremos la opción Save As...

MPLAB IDE v8.30 - [Untitled]

**File**

- New Ctrl+N
- Add New File to Project...
- Open... Ctrl+O
- Close Ctrl+E
- Save Ctrl+S
- Save As...**
- Save All Ctrl+Shift+S
- Open Workspace...
- Save Workspace
- Save Workspace As...
- Close Workspace
- Import...
- Export...
- Print... Ctrl+P
- Recent Files
- Recent Workspaces
- Exit

```

PROGRAMA.

;usará;
;ador de Windows dentro de MPLAB, en la que se encuentran alojadas las características del PIC que elegimos";

; inicialización para los parámetros del PIC; ALL.

del PIC.
/fosc)*coeficiente de fosc= "x"ps.

al Banco 0 de memoria RAM
ura del programa
//
//
//



;

;Asignación de los bits de los puertos de I/O.
;Puerto A.
;Sin_UsoRA0 equ .0; // Sin Uso RA0.
;Data_RA0 equ .0; // Bit 0 de datos.
;Colocamos los valores de inicialización del puerto A para cada una de sus terminales RA0-RAS.
;proga equ b'velores de las terminales en binario'; // Programación inicial del puerto A.
;Puerto B.
;Data_RB0 equ .0; // Bit 0 de datos.
;Sin_UsoRB0 equ .0; // Bit 0 de datos.
;Colocamos los valores de inicialización del puerto B para cada una de sus terminales RB0-RB7.
;prob equ b'velores de las terminales en binario'; // Programación inicial del puerto B.
;Puerto C.

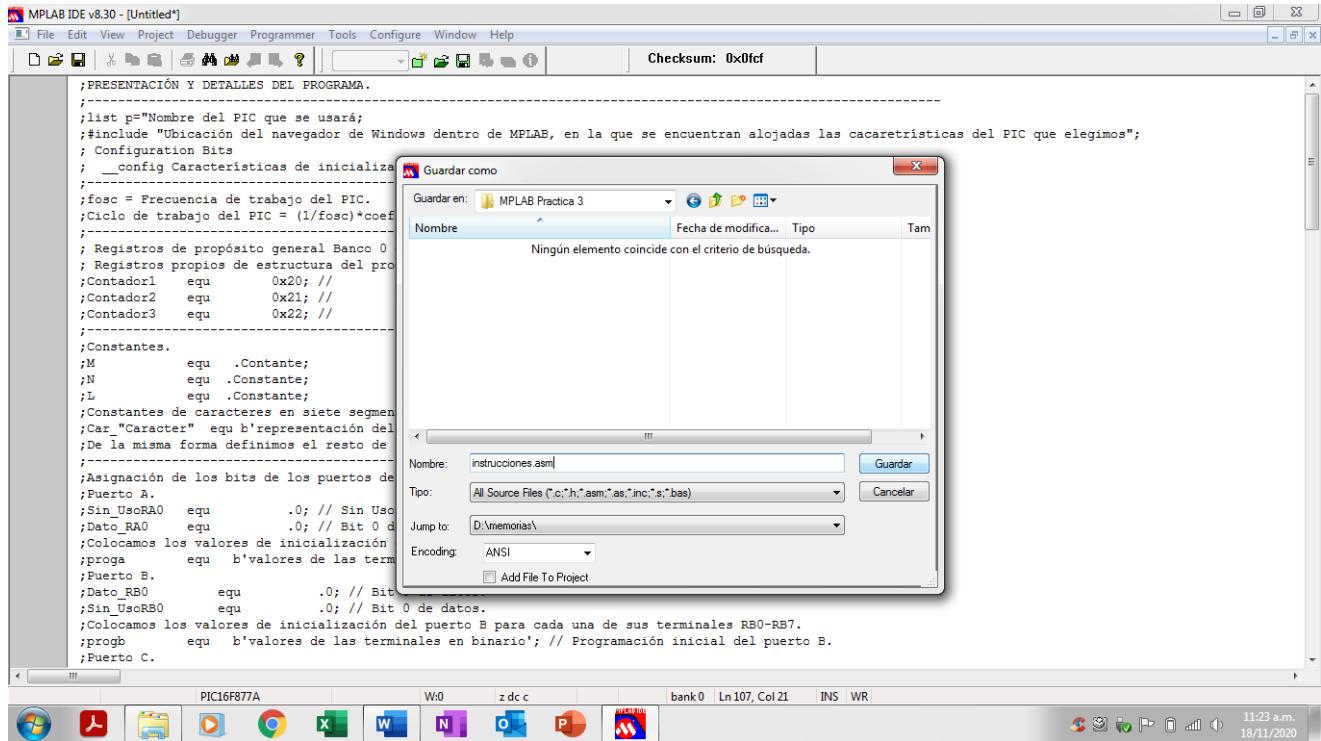
;

```

PIC16F877A W:0 z ddc bank 0 Ln 107, Col 21 INS WR

11:20 a.m. 18/11/2020

En esta opción, nos aparecerá un cuadro, en el que especificaremos el nombre del archivo anexando la extensión ".asm" que significa que es un archivo en lenguaje ensamblador y lo guardamos en este caso con el nombre de instrucciones.



El cuadro desaparecerá y el código se verá distinto, las letras se tornarán de diferentes colores según sea su función. En este caso todo el código se ve de color verde porque todos están especificados como comentarios.

MPLAB IDE v8.30 - [E:\10008000\Semestre 5\A.M.M\MPLAB Practica 3\instrucciones.asm]

File Edit View Project Debugger Programmer Tools Configure Window Help

```

1 ;PRESENTACIÓN Y DETALLES DEL PROGRAMA.
2 ;-----
3 ;list p="Nombre del PIC que se usará;
4 ;include "Ubicación del navegador de Windows dentro de MPLAB, en la que se encuentran alojadas las características del PIC que elegimos";
5 ; Configuration Bits
6 ; __config Características de inicialización para los parámetros del PIC; ALL.
7 ;-----
8 ;fosc = Frecuencia de trabajo del PIC.
9 ;Ciclo de trabajo del PIC = (1/fosc)*coeficiente de fosc= "x"μs.
10 ;-----
11 ; Registros de propósito general Banco 0 de memoria RAM
12 ; Registros propios de estructura del programa
13 ;Contador1 equ 0x20; //
14 ;Contador2 equ 0x21; //
15 ;Contador3 equ 0x22; //
16 ;-----
17 ;Constantes.
18 ;M equ .Contante;
19 ;N equ .Constante;
20 ;L equ .Constante;
21 ;Constantes de caracteres en siete segmentos.
22 ;Car,"Caracter" equ b'representación del carácter en binario'; Caracter Caracter en siete segmentos.
23 ;De la misma forma definimos el resto de los caracteres que usaremos.
24 ;-----
25 ;Asignación de los bits de los puertos de I/O.
26 ;Puerto A.
27 ;Sin_UsoRA0 equ .0; // Sin Uso RA0.
28 ;Dato_RA0 equ .0; // Bit 0 de datos.
29 ;Colocamos los valores de inicialización del puerto A para cada una de sus terminales RA0-RA5.
30 ;proga equ b'vellos de las terminales en binario'; // Programación inicial del puerto A.
31 ;Puerto B.
32 ;Dato_RB0 equ .0; // Bit 0 de datos.
33 ;Sin_UsoRB0 equ .0; // Bit 0 de datos.
34 ;Colocamos los valores de inicialización del puerto B para cada una de sus terminales RB0-RB7.

```

PIC16F877A W:0 z ddc bank 0 Ln 107, Col 21 INS WR

11:29 a.m. 18/11/2020

El siguiente paso seguiremos creando el proyecto y para eso en la herramienta Project seleccionamos New.

MPLAB IDE v8.30 - [E:\10008000\Semestre 5\A.M.M\MPLAB Practica 3\instrucciones.asm]

File Edit View Project Debugger Programmer Tools Configure Window Help

Project Wizard...

- New...
- Open...
- Close
- Set Active Project
- Quickbuild instrucciones.asm
- Package in .zip
- Clean
- Build Configuration
- Build Options...
- Save Project
- Save Project As...
- Add Files to Project...
- Add New File to Project...
- Remove File From Project
- Select Language Toolsuite...
- Set Language Tool Locations...
- Version Control...

```

1 ;PRES
2 ;-----
3 ;list
4 ;#inc
5 ; Con
6 ; -----
7 ;fosc
8 ;Cicl
9 ;-----
10 ; Reg
11 ; Reg
12 ;Cont
13 ;Cont
14 ;Cont
15 ;Cont
16 ;Cont
17 ;Cons
18 ;M
19 ;N
20 ;L
21 ;Cons
22 ;Car
23 ;De l
24 ;-----
25 ;Asignación de los bits de los puertos de I/O.
26 ;Puerto A.
27 ;Sin_UsoRA0 equ .0; // Sin Uso RA0.
28 ;Dato_RA0 equ .0; // Bit 0 de datos.
29 ;Colocamos los valores de inicialización del puerto A para cada una de sus terminales RA0-RA5.
30 ;proga equ b'vellos de las terminales en binario'; // Programación inicial del puerto A.
31 ;Puerto B.
32 ;Dato_RB0 equ .0; // Bit 0 de datos.
33 ;Sin_UsoRB0 equ .0; // Bit 0 de datos.
34 ;Colocamos los valores de inicialización del puerto B para cada una de sus terminales RB0-RB7.

```

PIC16F877A W:0 z ddc bank 0 Ln 107, Col 21 INS WR

11:33 a.m. 18/11/2020

Nos aparecerá una ventana como la siguiente.

MPLAB IDE v8.30 - [E:\10008000\Semestre 5\A.M.M\MPLAB Practica 3\instrucciones.asm]

File Edit View Project Debugger Programmer Tools Configure Window Help

Checksum: 0x0fcf

```

1 ;PRESENTACIÓN Y DETALLES DEL PROGRAMA.
2 ;-----
3 ;list p="Nombre del PIC que se usará;
4 ;include "Ubicación del navegador de Windows dentro de MPLAB, en la que se encuentran alojadas las características del PIC que elegimos";
5 ; Configuration Bits
6 ; __config Características de inicialización para los parámetros del PIC; ALL.
7 ;-----
8 ;fosc = Frecuencia de trabajo del PIC.
9 ;Ciclo de trabajo del PIC = (1/fosc)*coeficiente de fosc= "x"μs.
10 ;-----
11 ; Registros de propósito general Banco
12 ; Registros propios de estructura del p
13 ;Contador1 equ 0x20; //
14 ;Contador2 equ 0x21; //
15 ;Contador3 equ 0x22; //
16 ;-----
17 ;Constantes.
18 ;M equ .Contante;
19 ;N equ .Contante;
20 ;L equ .Contante;
21 ;Constantes de caracteres en siete segm
22 ;Car."Caracter" equ b'representación d
23 ;De la misma forma definimos el resto de los caracteres que usaremos.
24 ;
25 ;Asignación de los bits de los puertos de I/O.
26 ;Puerto A.
27 ;Sin_UsoRA0 equ .0; // Sin Uso RA0.
28 ;Dato_RA0 equ .0; // Bit 0 de datos.
29 ;Colocamos los valores de inicialización del puerto A para cada una de sus terminales RA0-RA5.
30 ;proga equ b'velues de las terminales en binario'; // Programación inicial del puerto A.
31 ;Puerto B.
32 ;Dato_RB0 equ .0; // Bit 0 de datos.
33 ;Sin_UsoRB0 equ .0; // Bit 0 de datos.
34 ;Colocamos los valores de inicialización del puerto B para cada una de sus terminales RB0-RB7.

```

PIC16F877A W:0 z ddc bank 0 Ln 107, Col 21 INS WR

11:34 a.m. 18/11/2020

En esta ventana, donde dice Project Name, colocamos el nombre del proyecto que en este caso será el mismo con el que nombramos el archivo, pero sin la extensión .asm y para Project Directory seleccionamos la carpeta donde alojaremos el proyecto que será la misma que el archivo .asm que creamos antes.

MPLAB IDE v8.30 - [E:\10008000\Semestre 5\A.M.M\MPLAB Practica 3\instrucciones.asm]

File Edit View Project Debugger Programmer Tools Configure Window Help

Checksum: 0x0fcf

```

1 ;PRESENTACIÓN Y DETALLES DEL PROGRAMA.
2 ;-----
3 ;list p="Nombre del PIC que se usará;
4 ;include "Ubicación del navegador de Windows dentro de MPLAB, en la que se encuentran alojadas las características del PIC que elegimos";
5 ; Configuration Bits
6 ; __config Características de inicialización para los parámetros del PIC; ALL.
7 ;-----
8 ;fosc = Frecuencia de trabajo del PIC.
9 ;Ciclo de trabajo del PIC = (1/fosc)*coeficiente de fosc= "x"μs.
10 ;-----
11 ; Registros de propósito general Banco
12 ; Registros propios de estructura del p
13 ;Contador1 equ 0x20; //
14 ;Contador2 equ 0x21; //
15 ;Contador3 equ 0x22; //
16 ;-----
17 ;Constantes.
18 ;M equ .Contante;
19 ;N equ .Contante;
20 ;L equ .Contante;
21 ;Constantes de caracteres en siete segm
22 ;Car."Caracter" equ b'representación d
23 ;De la misma forma definimos el resto de los caracteres que usaremos.
24 ;
25 ;Asignación de los bits de los puertos de I/O.
26 ;Puerto A.
27 ;Sin_UsoRA0 equ .0; // Sin Uso RA0.
28 ;Dato_RA0 equ .0; // Bit 0 de datos.
29 ;Colocamos los valores de inicialización del puerto A para cada una de sus terminales RA0-RA5.
30 ;proga equ b'velues de las terminales en binario'; // Programación inicial del puerto A.
31 ;Puerto B.
32 ;Dato_RB0 equ .0; // Bit 0 de datos.
33 ;Sin_UsoRB0 equ .0; // Bit 0 de datos.
34 ;Colocamos los valores de inicialización del puerto B para cada una de sus terminales RB0-RB7.

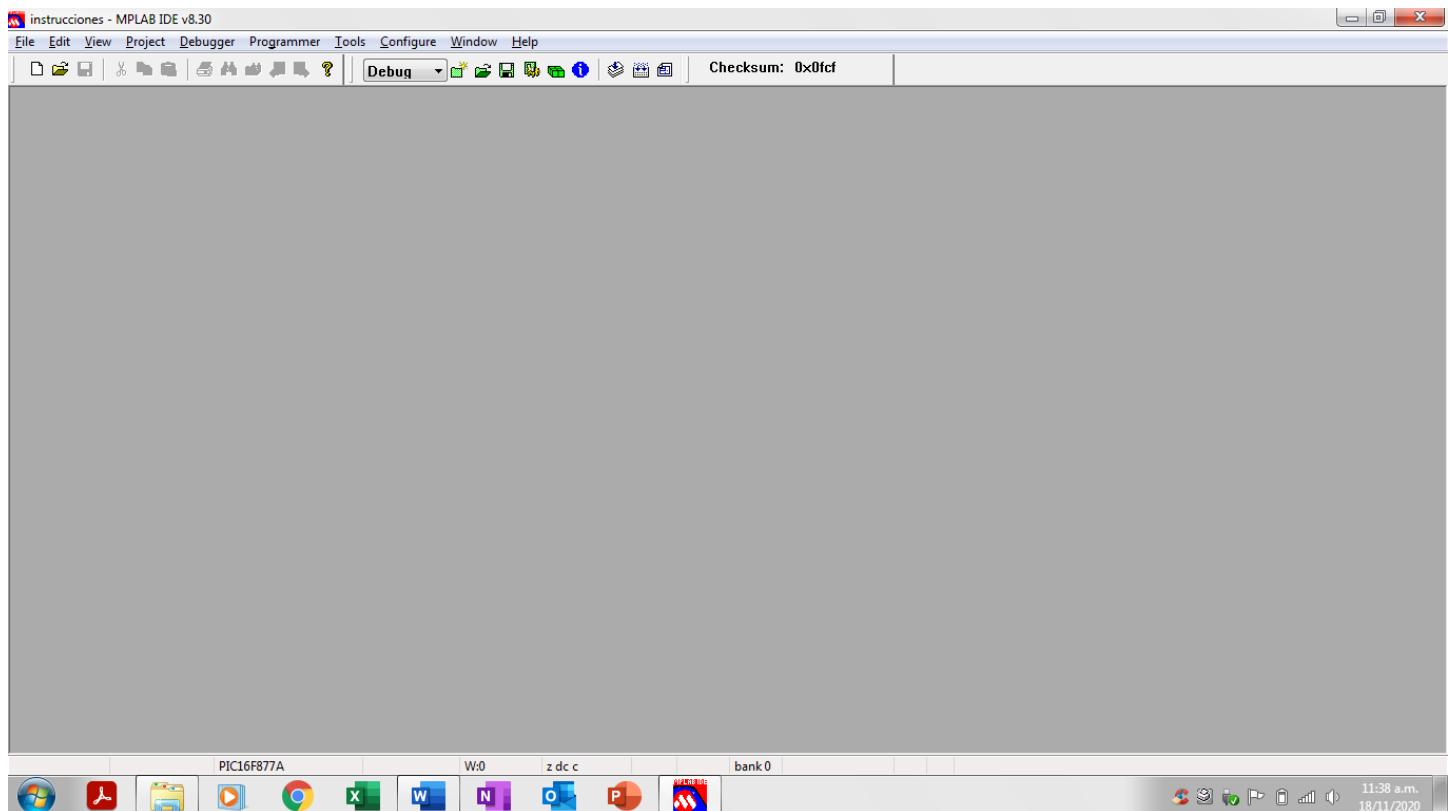
```

PIC16F877A W:0 z ddc bank 0 Ln 107, Col 21 INS WR

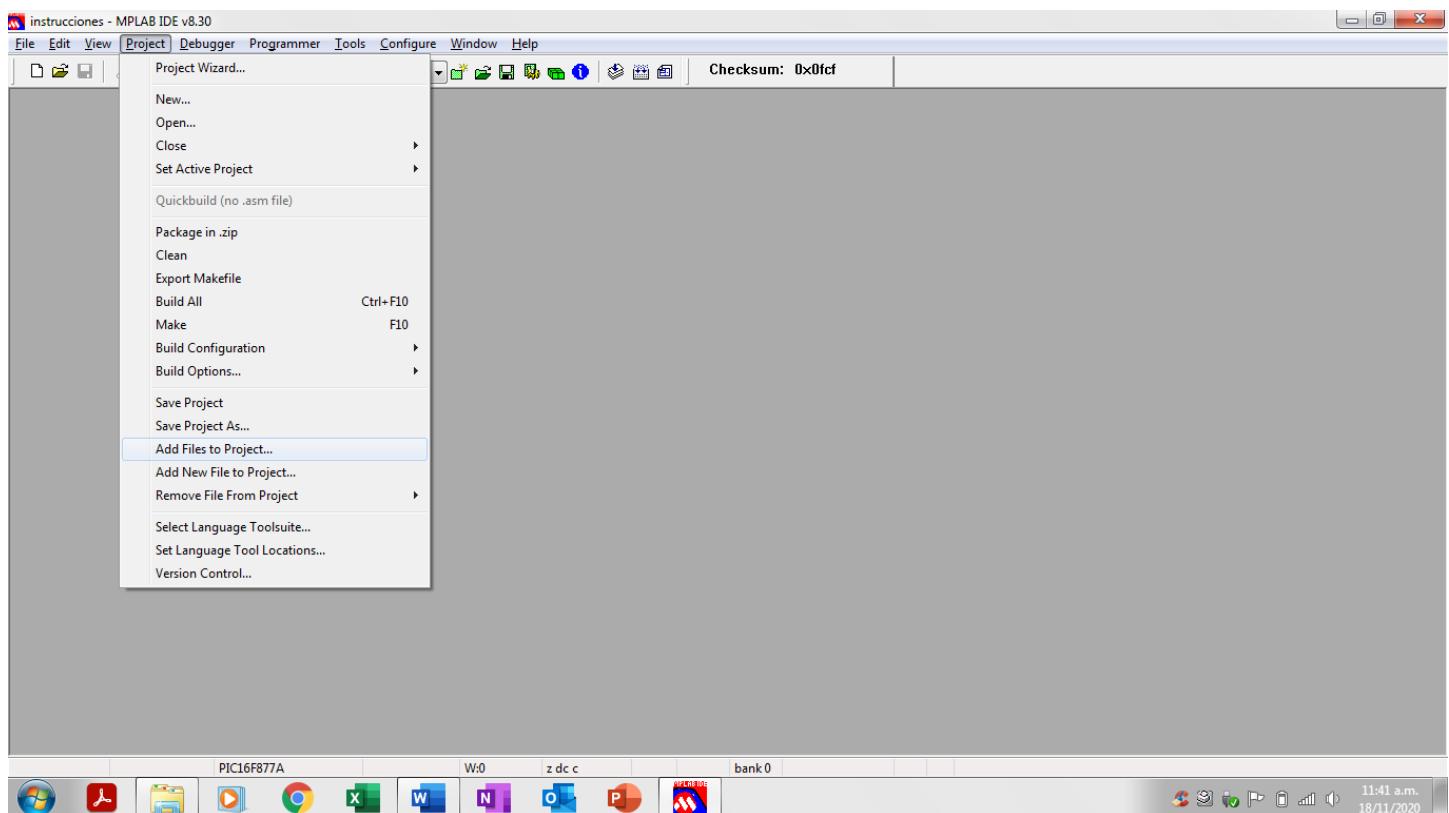
11:38 a.m. 18/11/2020

Seleccionamos OK.

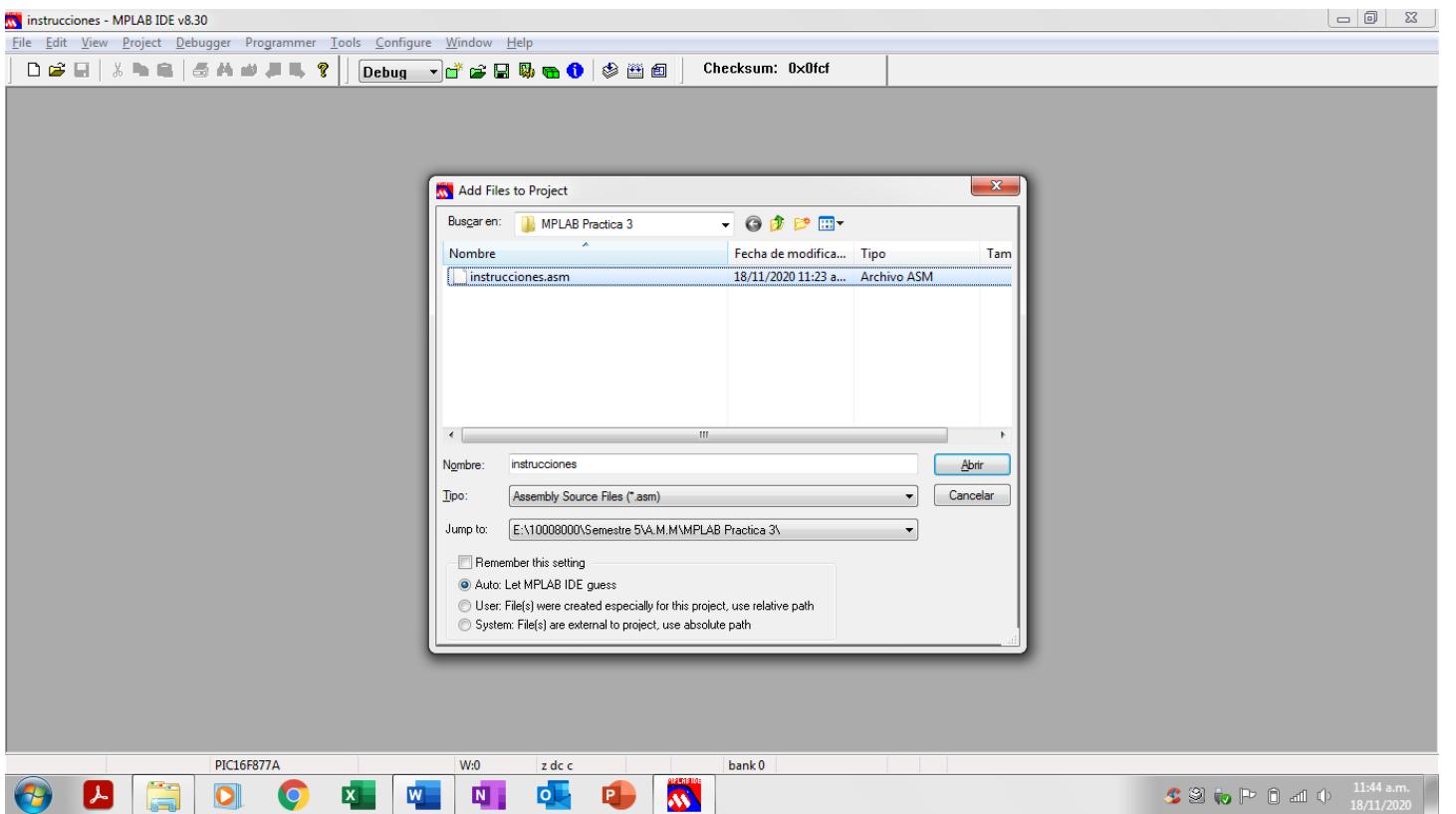
La ventana se mostrará de la siguiente forma.



Como siguiente paso en la herramienta de Project seleccionamos Add Files to Project.

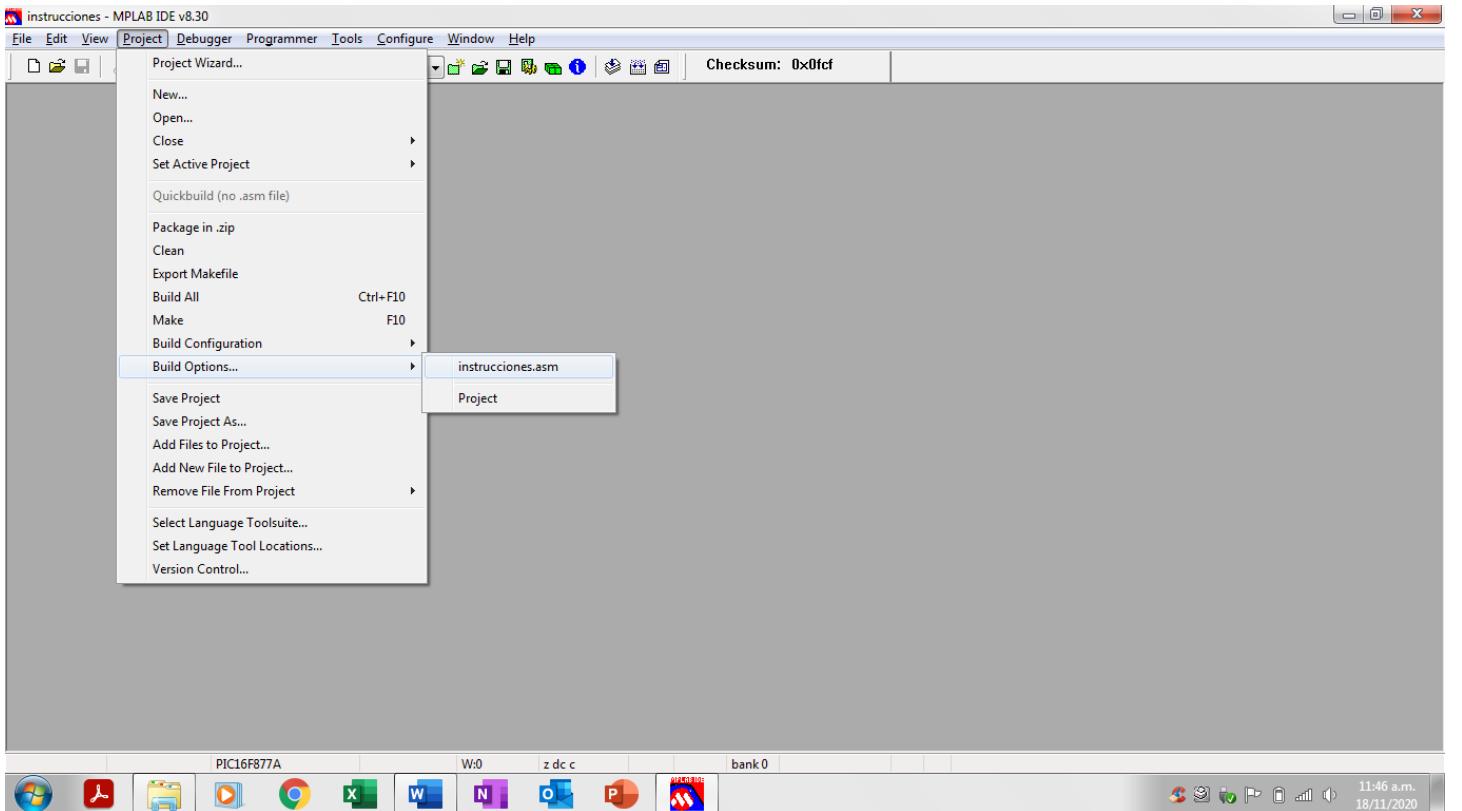


Aquí en la ventana que nos aparecerá seleccionaremos el archivo que creamos con extensión .asm y lo abriremos.

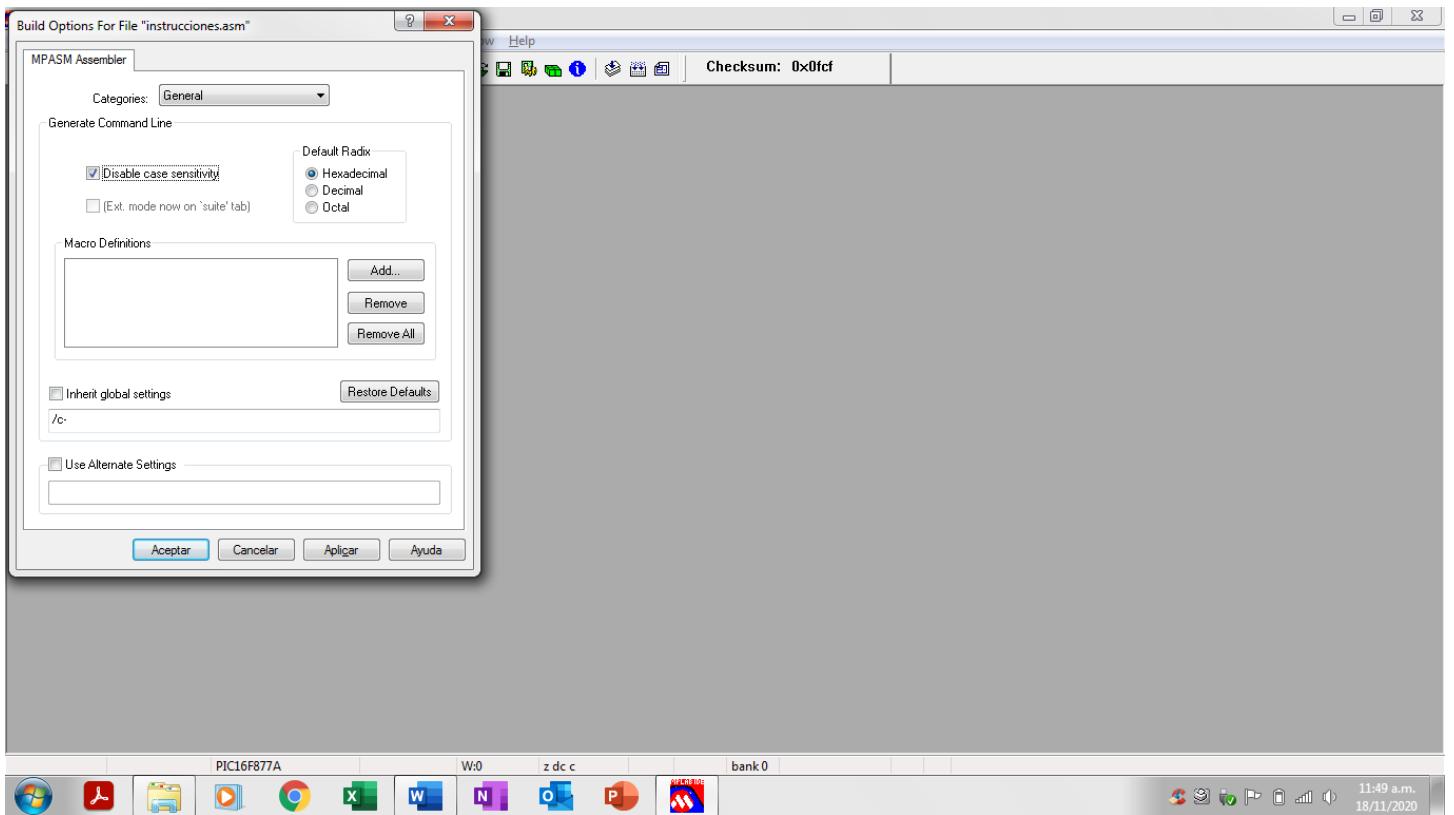


Luego de esto volverá a no haber ventanas activas, el próximo paso será en Project y seleccionamos Build Options.

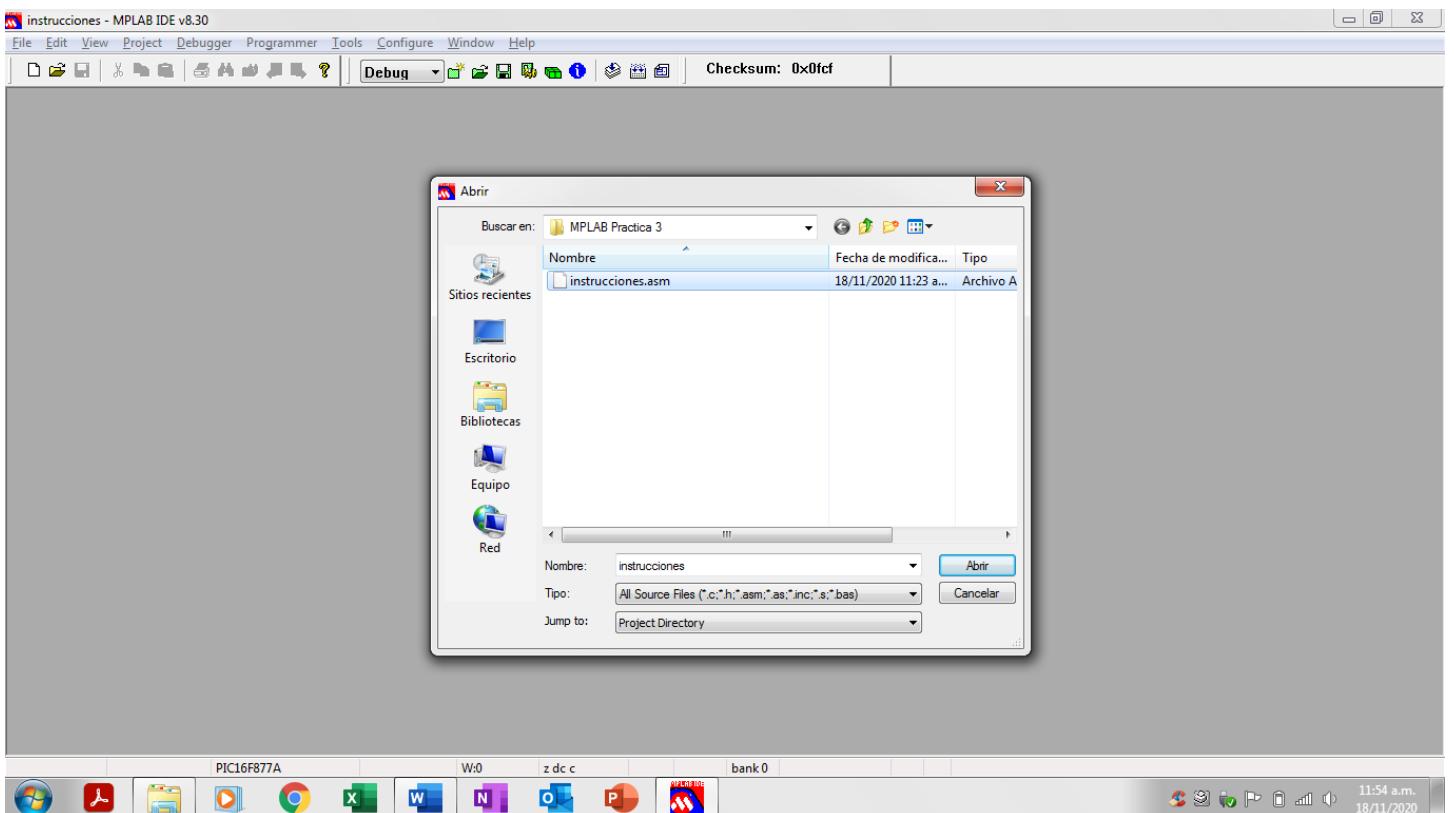
en el menú que aparece abrimos el de nombre del .asm que creamos.



En la ventana que aparecerá en la esquina superior derecha, sólo activaremos la opción Disable case sensitivity y damos click en Aceptar.



Por último, iremos a la herramienta File y seleccionamos la opción Open. Abriremos el archivo que tenga el nombre del proyecto.



Este será el último paso que haremos para guardar el proyecto. A partir de ahora sólo será necesario presionar la opción Save en la herramienta File o seleccionar el ícono que aparece luego de abrir el programa cada vez que lo modificuemos.

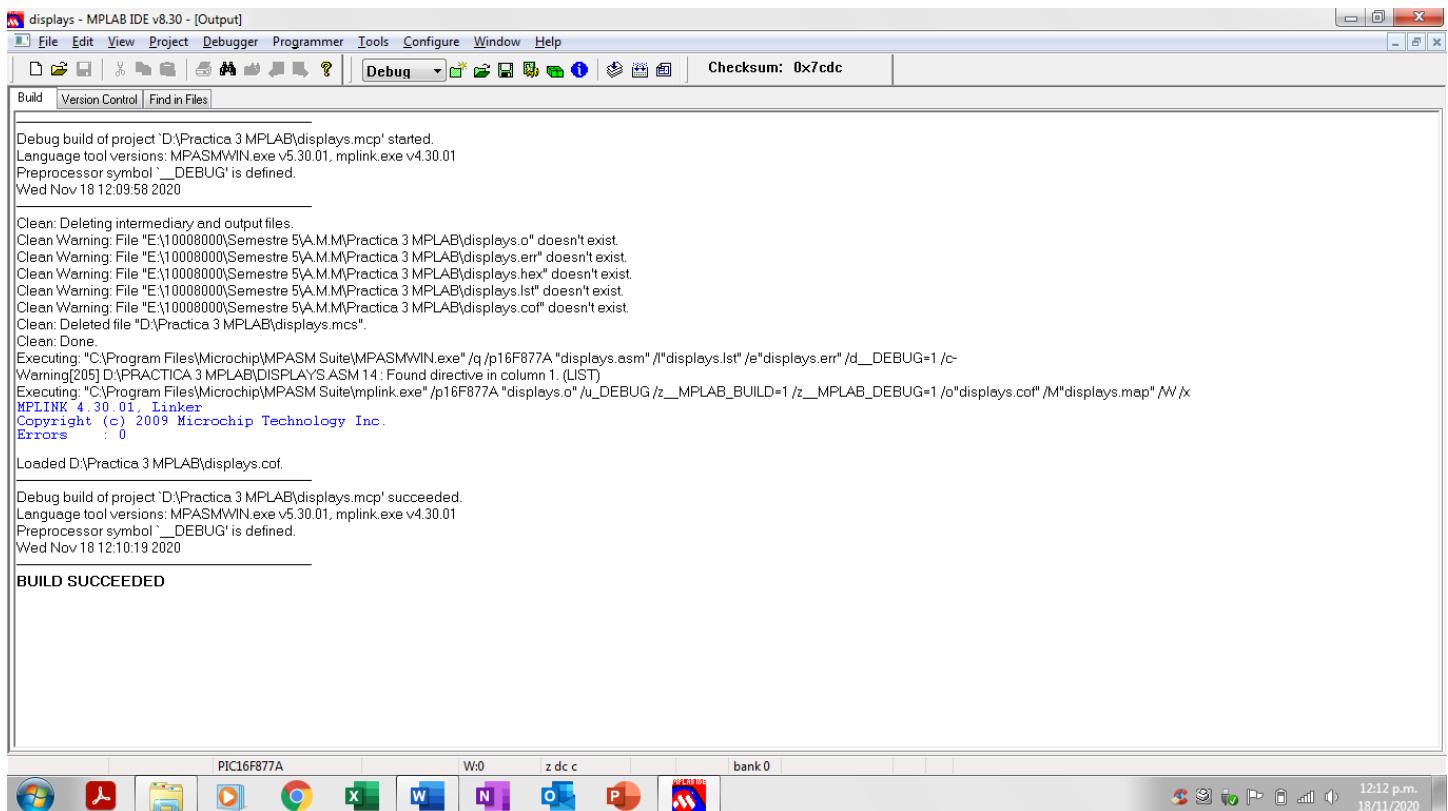
```
1 ;INSTITUTO POLITECNICO NACIONAL.
2 ;CECYT 9 JUAN DE DIOS BATIZ.
3 ;
4 ;PRACTICA 3 DESPLEGADO DE MENSAJES.
5 ;
6 ;GRUPO: 5IMX.
7 ;
8 ;INTEGRANTES:
9 ;1.- Villegas Monroy EMILIO.
10 ;
11 ;COMENTARIO DE LO QUE EL PROGRAMA EJECUTARA.
12 ;-----
13
14 list p=16f877A;
15 #include "c:\program files\microchip\mpasm suite\p16f877a.inc";
16
17 ; Configuration Bits
18 __config _XT_OSC & _WDT_OFF & _PWRTE_ON & _BODEN_OFF & _LVP_OFF & _CP_OFF; ALL
19 ;-----
20 ;
21 ;fosc = 4Mhz.
22 ;Ciclo de trabajo del PIC = (1/fosc)*4 =1 ps.
23 ;-----
24 ;
25 ; Registros de propósito general Banco 0 de memoria RAM.
26 ;
27 ; Registros propios de estructura del programa
28
29 Contador1    equ      0x20; //
30 Contador2    equ      0x21; //
31 Contador3    equ      0x22; //
32 ;
33 ;
34 ;Constantes.
```

Luego de esto, como se trata ya de un proyecto y no de un archivo de edición de texto podemos realizar el proceso de compilado que es para asegurar que no tengamos errores en la programación y para crear un archivo de extensión “.hex”, que nos servirá para cargar la memoria en el simulador Proteus.

Nuestro código tiene el siguiente aspecto.

```
1 ;INSTITUTO POLITECNICO NACIONAL.
2 ;CECYT 9 JUAN DE DIOS BATIZ.
3 ;
4 ;PRACTICA 3 DESPLEGADO DE MENSAJES.
5 ;
6 ;GRUPO: 5IMX.
7 ;
8 ;INTEGRANTES:
9 ;1.- Villegas Monroy EMILIO.
10 ;
11 ;COMENTARIO DE LO QUE EL PROGRAMA EJECUTARA.
12 ;-----
13
14 list p=16f877A;
15 #include "c:\program files\microchip\mpasm suite\p16f877a.inc";
16
17 ; Configuration Bits
18 __config _XT_OSC & _WDT_OFF & _PWRTE_ON & _BODEN_OFF & _LVP_OFF & _CP_OFF; ALL
19 ;-----
20 ;
21 ;fosc = 4Mhz.
22 ;Ciclo de trabajo del PIC = (1/fosc)*4 =1 ps.
23 ;-----
24 ;
25 ; Registros de propósito general Banco 0 de memoria RAM.
26 ;
27 ; Registros propios de estructura del programa
28
29 Contador1    equ      0x20; //
30 Contador2    equ      0x21; //
31 Contador3    equ      0x22; //
32 ;
33 ;
34 ;Constantes.
```

Al compilarlo se debe observar así.



```
displays - MPLAB IDE v8.30 - [Output]
File Edit View Project Debugger Programmer Tools Configure Window Help
Debug | Checksum: 0x7cdc
Build Version Control Find in Files

Debug build of project 'D:\Practica 3 MPLAB\displays.mcp' started.
Language tool versions: MPASMWIN.exe v5.30.01, mplink.exe v4.30.01
Preprocessor symbol '_DEBUG' is defined.
Wed Nov 18 12:09:58 2020

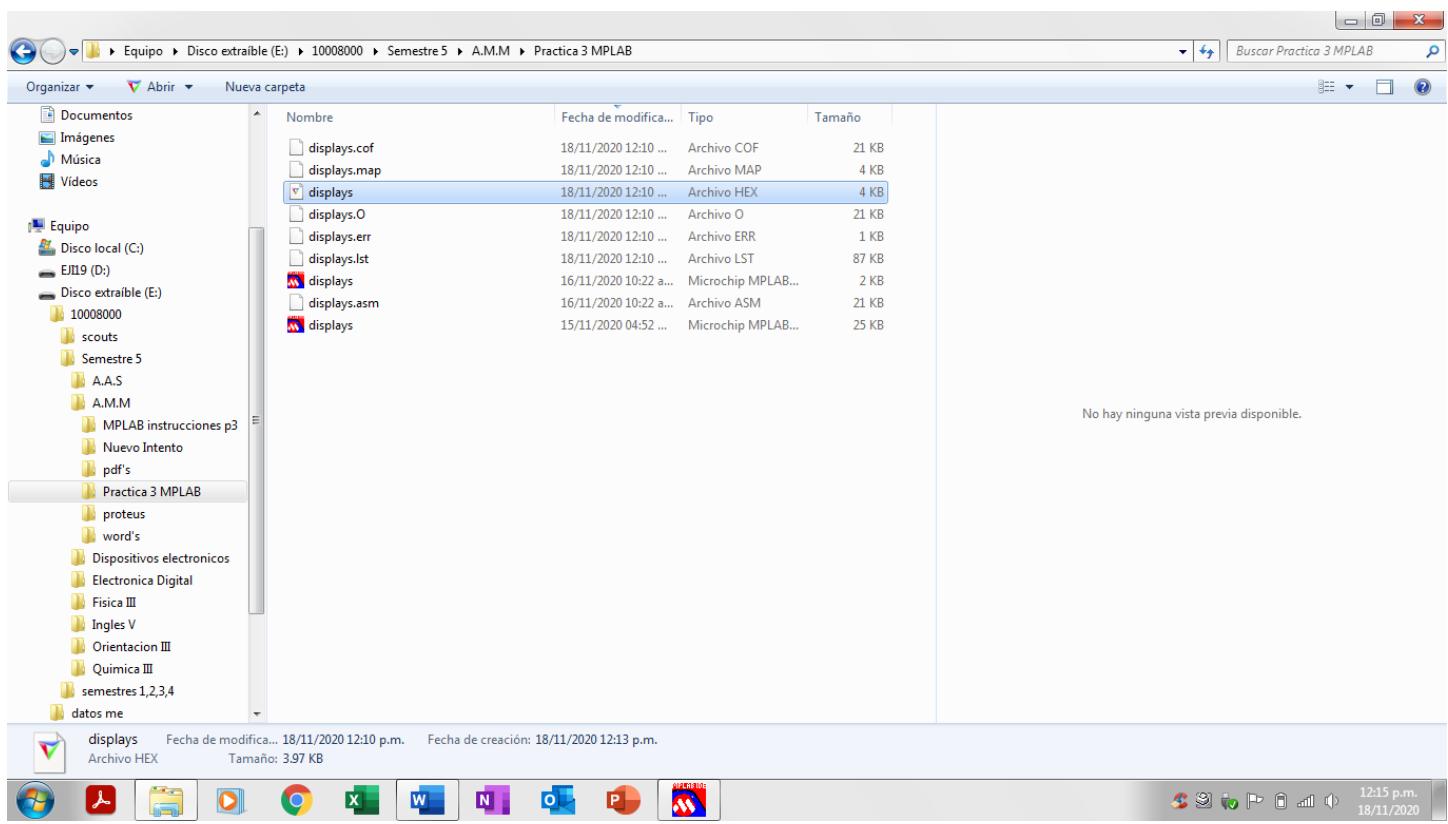
Clean: Deleting intermediary and output files.
Clean Warning: File "E:\10008000\Semestre 5\A.M.M\Practica 3 MPLAB\displays.o" doesn't exist.
Clean Warning: File "E:\10008000\Semestre 5\A.M.M\Practica 3 MPLAB\displays.err" doesn't exist.
Clean Warning: File "E:\10008000\Semestre 5\A.M.M\Practica 3 MPLAB\displays.hex" doesn't exist.
Clean Warning: File "E:\10008000\Semestre 5\A.M.M\Practica 3 MPLAB\displays.lst" doesn't exist.
Clean Warning: File "E:\10008000\Semestre 5\A.M.M\Practica 3 MPLAB\displays.cof" doesn't exist.
Clean: Deleted file "D:\Practica 3 MPLAB\displays.mcs".
Clean: Done.
Executing: "C:\Program Files\Microchip\MPASM Suite\MPASMWIN.exe" /q/p16F877A "displays.asm" /l"displays.lst" /e"displays.err" /d__DEBUG=1 /c
Warning[205] D:\PRACTICA 3 MPLAB\DISPLAYS ASM14 : Found directive in column 1. (LIST)
Executing: "C:\Program Files\Microchip\MPASM Suite\mplink.exe" /p16F877A "displays.o" /u_DEBUG /z__MPLAB_BUILD=1 /z__MPLAB_DEBUG=1 /o"displays.cof" /M"displays.map" /N/x
MPLINK 4.30.01, Linker
Copyright (c) 2009 Microchip Technology Inc.
Errors : 0

Loaded D:\Practica 3 MPLAB\displays.cof.

Debug build of project 'D:\Practica 3 MPLAB\displays.mcp' succeeded.
Language tool versions: MPASMWIN.exe v5.30.01, mplink.exe v4.30.01
Preprocessor symbol '_DEBUG' is defined.
Wed Nov 18 12:10:19 2020

BUILD SUCCEEDED
```

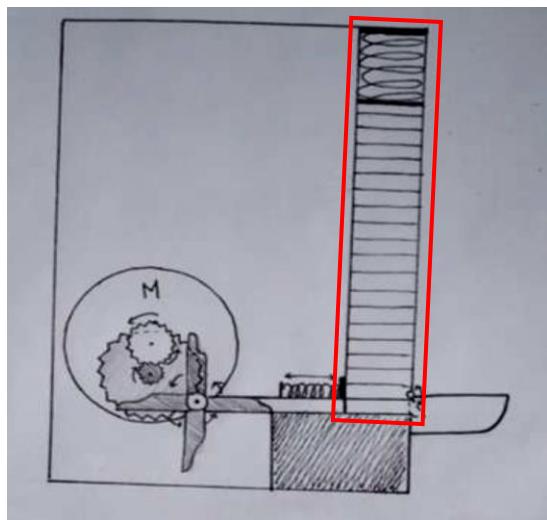
Si vamos a los archivos en la carpeta en la que guardamos los documentos del proyecto veremos que se creó un nuevo archivo de extensión .hex, que es el que usaremos para cargar la memoria en Proteus.



## Funcionamiento de los circuitos

### Control de motores.

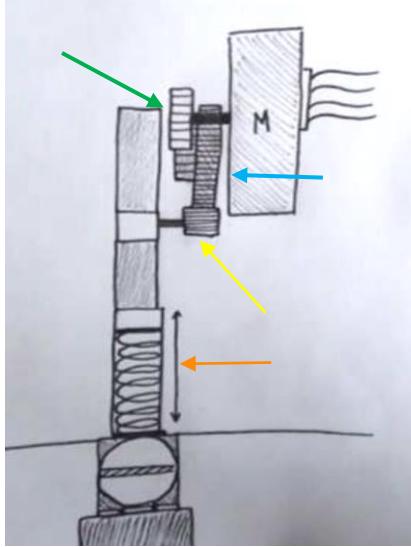
El circuito se basa en el mismo temporizador del sistema de control de horarios para su activación a manera de que cada impulso de interrupción nuevo sirva como PWM para los motores a pasos de 6 hilos que usaremos en el proyecto los cuales controlarán el dispensado de las pastillas. Debido a que únicamente usa piezas mecánicas para su funcionamiento, realmente podemos decir que el trabajo de programación y electrónica es nulo a partir del motor, ya que una vez que un impulso llega a él éste girará 90 grados como máximo y se quedará ahí, sin embargo, mientras va completando el giro, una pieza de piñón compuesto intercepta el movimiento debido a una corona calada que se encuentra en el externo del eje del rotor del motor para que posteriormente la fuerza impartida inicial del motor antes de llegar a su punto máximo sea de 1/4 veces mayor llegando así a una pieza que es fabricada especialmente para este proyecto y se parece a un molino de agua, ya que de un lado con un nuevo piñón 1/3 más grande que el radio menor del piñón compuesto y del otro lado tiene cuatro láminas perpendiculares entre sí las cuales al recibir el impulso del primer piñón se girarán poco a poco en el mismo sentido que el motor pero con 5/4 de la fuerza original pudiendo desplazar una paleta con resorte que sacará la pastilla y regresará debido al resorte y que el motor habrá dado su giro de 90 grados al igual que la pieza que empuja la paleta ya que mencionamos que la forma de ésta en el lado de las laminillas eran diametralmente opuestas y perpendiculares entre sí.



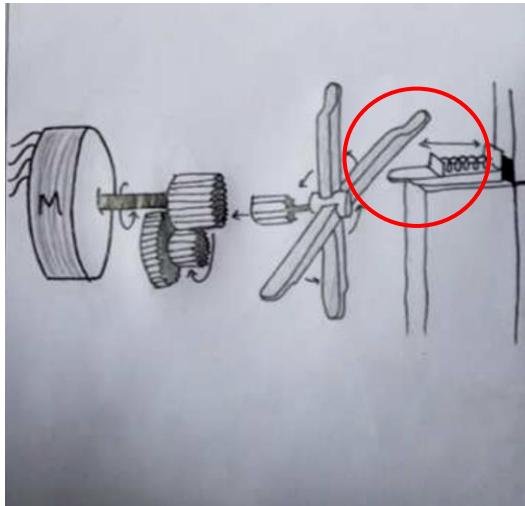
Como podemos ver en el diagrama, el bloque consta de todas las piezas que se mencionaron antes. En este momento nos enfocaremos un poco más en el lado remarcado el cual es el bloque de almacenamiento de las pastillas.

La “casilla” depende su forma de la pastilla, pero suponiendo que sea cilíndrica vemos en la parte superior una tapa removible con un resorte, éste permite almacenar las pastillas de forma que queden ligeramente apretadas ero no en exceso, de esta forma cuando la pieza deslizable entre de izquierda a derecha las pastillas se recorran hacia abajo una posición cayendo la dispensada en una pequeña bandeja.

De manera rápida también podemos observar un poco mejor el sentido de giro de las piezas después del motor a pasos. Al ser un número de piezas impar, el último giro que se dé será en el mismo sentido que el del motor dio originalmente.



En esta vista del diagrama, es más fácilmente apreciable la manera en que cada una de las piezas se junta entre sí. La corona calda del motor señalada en verde, el piñón compuesto señalado en azul con el lado pequeño tocando la corona y el lado grande tocando el piñón de la pieza diseñada señalada en amarillo y las paletas de la pieza empujando desde abajo a la pieza deslizable señalada en naranja la cual sólo se ve el resorte que está encima de ella.



En esta última imagen se puede ver un poco mejor la manera en que los componentes interactúan entre sí y su maneta de ensamblaje debido al tipo de vista. En uno de los extremos podemos observar cómo al girar, las paletas golpean y empujan el lado deslizable que tiene una terminación parecida del mismo lado, pero invertida a modo de espejo para que la fuerza necesaria para mover la pieza sea menor, reduciendo un poco la carga de trabajo del motor.

## Automatización de tiempos.

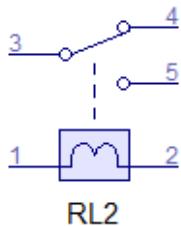
Este sistema se encuentra dominado por un microcontrolador que se encarga de administrar mediante temporizadores los horarios de operación activa del CAD. Usando un menú de interrupciones de tiempo que se puede “editar” podremos hacer que cada cierto rango de horas el microcontrolador genere una salida que activará los sistemas de control de motores y de alarma para avisar que la pastilla está dispensada. Como podemos darnos cuenta en este sistema dependemos únicamente de una herramienta de programación por lo que el software elaborado debe tener algunas consideraciones en caso de saber que serpa utilizado a largo plazo, como la colocación de WDT para aminorar el efecto de algún fallo en el sistema.

```
; INSTITUTO POLITECNICO NACIONAL.  
; CECYT 9 JUAN DE DIOS BATIZ.  
  
; PROYECTO AULA.  
; LA PANDILLA MANTEQUILLA.  
  
; GRUPO: 6IM3.  
  
; INTEGRANTES:  
; ALVARADO RODRÍGUEZ FERNANDO BRYAN  
; AYALA BERNAL ISRAEL  
; BARRERA CASTILLO ERICK ABRAHAM  
; HERNÁNDEZ CABALLERO MAURICIO  
; SÁNCHEZ MARTÍNEZ FELIPE  
; VILLEGRAS MONROY EMILIO  
  
; FECHA DE ENTREGA DEL REPORTE:  
  
; EL PROGRAMA SERÁ EL MENÚ DE NUESTRO PASTILLERO, EL  
; CUAL CON UN DISPOSITIVO DE SALIDA (LCD 20x4) SE PODRÁ  
; VER EL MENÚ, Y CON UN DISPOSITIVO DE ENTRADA (TECLADO  
; MATRICIAL 4x4) EL USUARIO PODRÁ INTERACTUAR CON  
; NUESTRO SISTEMA.  
  
;  
; List p=16f877A;  
#include "c:\Program files\Microchip\Mpasm Suite\p16f877a.inc";  
_CONFIG _CP_OFF & _WDT_OFF & _BODEN_OFF & _PWRTE_ON & _XT_OSC & _WRT_OFF & _LVP_OFF & _CPD_OFF;  
  
;  
; Fosc = 4 MHz.  
; Ciclo de trabajo del PIC = (1/fosc)*4 = 1 µs.  
; T int =(256-tmr0)*(P)*((1/4000000)*4) = 1 ms. // Tiempo de interrupción.  
; tmr0=131, P=8.  
; freq int = 1 / t int = 1 KHz.
```

Posteriormente haremos el anexo del programa fuente y diagrama de flujo completos del programa, la imagen es sólo ilustrativa del trabajo y objetivo de lo que realizó.

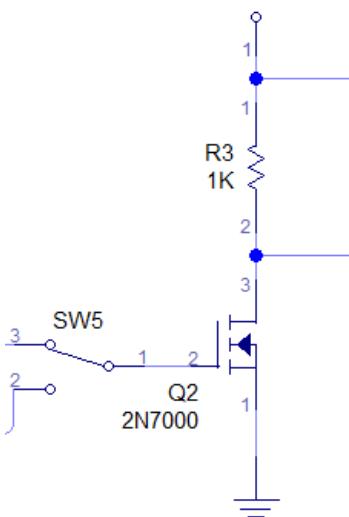
## Sistema de alarma.

Este sistema está directamente relacionado con el sistema de control de horarios. Su funcionamiento inicialmente estaba basado en el cambio de tiros de corriente de los relevadores, pero al ser piezas mecánicas y que tienen un límite de cambios con respecto al tiempo, decidimos cambiarlo por piezas electrónicas como los MOSFET. Decidimos usar transistores por sus propiedades de cerrarse y abrirse como interruptores electrónicos, otra cosa que mantuvimos en mente fue que son elementos con mayor grado de compatibilidad de trabajo con el otro componente de alta potencia que se usa en este bloque que son las bocinas. La potencia es una propiedad que ambos elementos comparten fácilmente sin necesidad de otros componentes debido a que se manejan con voltajes y no con corrientes como otros transistores (BJT o JFET).



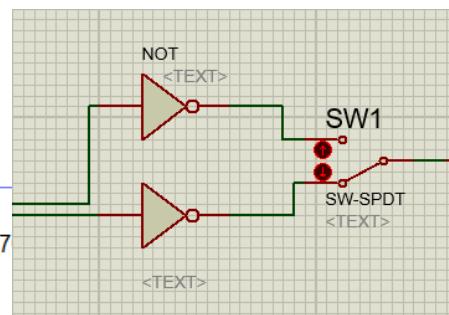
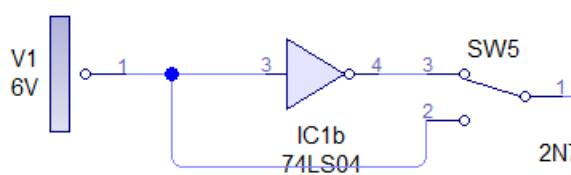
Como sabemos, un relé es un componente mecánico que funciona mediante el uso de campos electromagnéticos pequeños para abrir o cerrar canales de corriente eléctrica. Anteriormente usábamos este componente para desviar la corriente a otro lado una vez la alarma se activará y mediante una NOT y un diodo rectificador pudimos hacer que con la misma señal cambiara a donde queríamos dos veces usando a nuestro favor los dos canales del relé y el común de un interruptor de dos polos un tiro.

Sin embargo, esto no fue suficiente ya que con el tiempo los relés en la realidad se gastan entre más se usen. Tomando en cuenta que se tienen aproximadamente 10,0000 cambios por relé, junto con un horario de alarma cada 8hrs durante un año, la necesidad de cambio de los componentes sería cada 3 años, lo cual no es nada conveniente para nadie. Por esta razón optamos por el uso de los transistores como mencionamos antes de la siguiente forma:



Esta es la forma en la que un MOSFET de tipo E canal N suele conectarse, pero lo que hace que funcione como el interruptor que sustituye al relé de manera tan simple es su capacidad de corte y saturación dependiendo del nivel de voltaje que encuentres en la terminal 2 o Gate, y ya que los pulsos que salen del microcontrolador en las subrutinas de interrupción van de niveles altos a bajos, podemos hacer un control de corte a saturación del transistor con el uso del switch de dos polos un tiro que vemos en la parte derecha. El tener dos entradas una salida nos proporciona el sustituto perfecto para el relé. Sin embargo, aún es un poco pronto para decir que es todo, si tuviéramos dos entradas iguales sería lo mismo si cambiásemos de un polo a otro ya que reciben la misma señal y la salida es la misma por lo que no nos proporciona ese cambio que necesitamos.

Pulso del PIC 0-5V



Esta es la presentación de las dos formas de conectar el sistema de alarma para que haga el cambio y pueda ser desactivado hasta el próximo impulso del microcontrolador. En el ejemplo uno podemos ver que sólo hay una NOT, eso se debe a que está conectada directamente a la salida del microcontrolador que produce las interrupciones cada n horas sin embargo en las simulaciones nos causó algunos problemas debido a que el cambio de polos en el interruptor afectaba en funcionamiento del PIC a pesar de estar configurado como salida ese pin. Por lo que decidimos conectarlo a otras dos señales que no estuvieran directamente conectadas al PIC pero sí recibieran las interrupciones al mismo tiempo que el sistema de alarma y por descontado elegimos al sistema de control de motores. Dado que son 4 podemos elegir sólo dos de ellos y así con dos entradas, dos NOT y el mismo interruptor de tres líneas dos polos un tiro pudimos hacer que el circuito funcionara de la misma manera que el relé sólo que sin hacer interferencia con las señales de salida en el microcontrolador y amplificando el tiempo de vida de los circuitos en general.

Nota: La conexión del segundo circuito al sistema de control de motores es de forma paralela por lo que no afecta en absoluto el funcionamiento de ninguno de los dos a diferencia de la conexión serie del primero entre PIC y alarma.

## Reloj digital en tiempo real.

Para el sistema al que nos referimos pensamos originalmente en el uso del microcontrolador nuevamente para llevar a cabo una subrutina como función de reloj digital, sin embargo, mediante el uso de un microcontrolador especializado en ello (y además recuperado y reparado de un dispositivo electrónico) pudimos reducir la carga de trabajo para el microcontrolador principal, así como reducir el uso de espacio físico en el CAD para el sistema de reloj. La idea de este apartado es únicamente un extra, así como debe haber personas a las que se les dificulte la lectura de un reloj de manecillas seguramente habrá algunos inconvenientes con el paso del tiempo sobre ellos, por lo que un reloj digital en ambas modalidades del proyecto nos parece una buena idea que podemos implementar como un apoyo extra en cuanto a horarios o fecha.

Como ya hemos visto antes, el manejo de una LCD es imprescindible en el desarrollo de este apartado, por lo que haciendo memoria el tiempo de desbordamiento del Timer 0 se calcula según la siguiente ecuación:

$$T_{int} = (256 - \text{Carga TMR0}) * P.T_{CM}$$

Donde  $T_{CM}$  es el ciclo de máquina que se puede calcular mediante la ecuación:

$$T_{CM} = (1/fosc) * 4$$

Y los tiempos máximos y mínimos del TMR0 se calculan de la siguiente forma:

Tiempo mínimo.

Si tmr0 = 255 y P = 2.

$$T_{int} = (256 - 255) * 2 * 0.000001 = 2 \mu s.$$

Tiempo máximo.

Si tmr0 = 0 y P = 256.

$$T_{int} = (256 - 0) * 256 * 0.000001 = 65.5 ms.$$

## INTERRUPCIONES

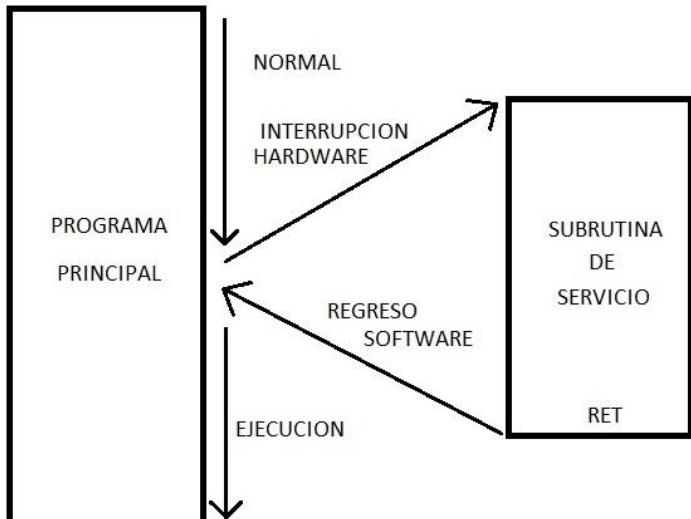
El tener que examinar periódicamente las banderas de estado de los dispositivos de E/S, monopoliza una porción importante del tiempo de operación del microprocesador. Esto reduce la cantidad de información procesada o transmitida por el sistema durante un periodo específico. De ahí que sea mucho más ventajoso, tanto en términos de la cantidad de información procesada como de la complejidad del programa, que el dispositivo periférico solicite directamente el servicio del microprocesador. Las interrupciones dotan al sistema de esta capacidad.

## CONCEPTOS BÁSICOS DE INTERRUPCIONES

Una interrupción se puede definir como una señal proveniente de un dispositivo externo, que llega a una entrada del microprocesador dedicada a este propósito y que le indica al microprocesador que el dispositivo que la originó está solicitando servicio.

Cuando ocurre una interrupción, el microprocesador suspende temporalmente la ejecución del programa principal y transfiere el control a una subrutina especialmente diseñada para atender al dispositivo que provocó la interrupción, a esta subrutina se le denomina subrutina de servicio de la interrupción (*Interrupt Service Routine*) o manejador de interrupción (*Interrupt Handler*). Al terminar el servicio, el microprocesador regresa al programa principal, continuando con sus actividades normales. Desde este punto de vista, una interrupción es esencialmente una llamada a subrutina iniciada por un circuito externo (hardware) y con un retorno controlado por programa (software).

Además, las interrupciones tienen la característica de que son eventos asíncronos, es decir, pueden ocurrir en cualquier momento durante la ejecución del programa principal sin posibilidad de hacer una predicción exacta. Por ello hay que hacer consideraciones especiales y tener cuidados adicionales al manejarlas.



## USOS DE LAS INTERRUPCIONES

Las interrupciones se utilizan:

- En la detección de eventos catastróficos como las fallas súbitas en las fuentes de alimentación. Con la interrupción el microprocesador puede ser alertado de preservar la información y desconectar los dispositivos que puedan dañarse.
- Cuando el microprocesador no necesita información con frecuencia. Como sistemas de alarma. El dispositivo genera una interrupción cuando llega el momento, y mientras tanto el microprocesador puede ejecutar otras tareas.
- En la atención a dispositivos periféricos que son mucho más lentos que el microprocesador: teclados, pantallas, impresoras. En vez de que el microprocesador los espere, es más eficiente que los dispositivos lo interrumpan cuando tengan algo nuevo que enviar o estén listos para recibir.

## **VENTAJAS Y DESVENTAJAS DE LAS INTERRUPCIONES**

### Ventajas:

- 1.- Liberan al microprocesador de la pérdida de tiempo por esperar a que ocurra cierto evento.
- 2.- Permiten la ejecutar un programa principal y un control simultáneo de varios dispositivos externos.
- 3.- Proporcionan servicio prioritario a dispositivos críticos dentro del sistema.
- 4.- Facilitan la detección de eventos en “tiempo real”.

### Desventajas:

- 1.- Los circuitos externos adicionales son necesarios para la señal de interrupción.
- 2.- Es muy difícil probar su funcionamiento y encontrar fallas debido a su naturaleza asíncrona.
- 3.- Pueden requerir instrucciones adicionales además de las estrictamente necesarias para dar servicio al dispositivo. Por ejemplo, para preservar registros o identificar al dispositivo.

## **TIPOS DE INTERRUPCIONES**

Un microprocesador puede tener entradas para responder a dos tipos de interrupciones: inhibibles (*maskable*) y no inhibibles (*no maskable*).

Cuando se activa una entrada de interrupción no inhibible, el microprocesador siempre es interrumpido, es decir, la señal de interrupción es aceptada bajo cualquier condición. Esto hace que las interrupciones no inhibibles sean las más apropiadas para manejar eventos catastróficos tales como la pérdida de energía.

Por otro lado, cuando se activa una entrada de interrupción inhibible, el microprocesador reconoce la interrupción solamente si esa entrada se encuentra habilitada. Las entradas de interrupción inhibibles se habilitan o inhabilitan bajo el control del programa. Si la entrada está inhabilitada, el microprocesador ignora la interrupción.

## **RESPUESTA A UNA INTERRUPCIÓN**

En respuesta a una interrupción el microprocesador realiza las siguientes operaciones:

- 1.- Se completa la instrucción que está en proceso.
- 2.- Se ejecuta un ciclo de máquina especial, durante el cual se almacena el contenido del contador de programa (PC), y se transfiere el control a una dirección apropiada.
- 3.- Se inicia la ejecución de la subrutina de servicio a la interrupción, preservando el estado de la CPU si esto es necesario.
- 4.- Si son varios los dispositivos que pudieron haber causado la interrupción, se identifica aquel que haya solicitado la interrupción primero y tenga mayor prioridad.
- 5.- Se ejecuta la parte de la subrutina de servicio que compete directamente a la atención del dispositivo que interrumpió.
- 6.- Se restaura el estado original del microprocesador.
- 7.- Se devuelve el control a la instrucción siguiente, a aquélla donde ocurrió la interrupción.

## **LAS INTERRUPCIONES Y LOS TEMPORIZADORES**

Las interrupciones permiten a cualquier suceso interior o exterior interrumpir la ejecución del programa principal en cualquier momento. En el momento de producirse la interrupción, el PIC ejecuta un salto a la rutina de atención a la interrupción que se encuentra en la dirección 0x0004, previamente definida por el programador, donde se atenderá a la demanda de la interrupción. Cuando se termina de ejecutar

dicha rutina, el PIC retorna a la ejecución del programa principal en la misma posición de la memoria de programa donde se produjo la interrupción.

El manejo de interrupciones permite realizar programas que no tienen que estar continuamente consultando sucesos internos o externos mediante técnicas de consulta, las cuales provocan retardos o paradas en la ejecución del programa principal.

Los Timer o temporizadores son módulos integrados en el PIC que permite realizar cuentas tanto de eventos internos como externos. Cuando la cuenta es interna se habla de temporización y cuando la cuenta es externa se habla de contador. Los timer están íntimamente ligados al uso de las interrupciones, pero no por ello se utilizan siempre de forma conjunta.

### **Subrutinas de pérdida de tiempo.**

Lazos de pérdida de tiempo.

Para determinar qué tiempo utiliza la CPU para ejecutar una instrucción, tenemos que definir el ciclo de máquina, según microchip el ciclo de maquina está definido por la siguiente formula.

$$CM = (1 / f_o) \times 4$$

Si el oscilador conectado al microcontrolador es de 4 MHz entonces.

$$CM = (1 / 4000000) \times 4 = 1 \text{ ms. Entonces } ts = 1 \text{ ms.}$$

Esto significa que cada instrucción simple la CPU la ejecutara en 1 ms y los saltos en 2 ms.

Subrutina de pérdida de tiempo de un lazo

movlw L;	1 ts	$T_{LAZO} = 1 \text{ ts} + 1 \text{ ts} + (1 \text{ ts} + 2 \text{ ts}) * (L-1) + 2 \text{ ts}$
movwf contador1;	1 ts	$T_{LAZO} = (4 + 3(L-1)) * ts$
Loop1 decfsz contador1;	1 ts o 2 ts	$T_{LAZO} = (1 + 3L) * ts$
goto Loop1;	2 ts	

Fórmula para calcular el tiempo que tarda la subrutina de pérdida de tiempo de 1 lazo

Para el tiempo mínimo (L=1):

$$T_{LAZO} = (1 + 3(1)) * ts = 4 \mu s$$

Para el tiempo máximo (L=255):

$$T_{LAZO} = (1 + 3(255)) * ts = 766 \mu s = 0.766 \text{ ms}$$

Subrutina de pérdida de tiempo de dos lazos

movlw N;	1 ts	Loop1 decfsz contador1;	1 ts o 2 ts
movwf contador2;	1 ts	goto Loop1;	2 ts
Loop2 movlw L;	1 ts	decfsz contador1;	1 ts o 2 ts
movwf contador1;	1 ts	goto Loop2;	2 ts

Si nos damos cuenta, la parte de rojo es la subrutina de pérdida de tiempo de 1 lazo, a esta parte la llamaremos X, y la parte negra la tomaremos como si fuera la subrutina de tiempo de 1 lazo.

Entonces la fórmula para calcular el tiempo de esta subrutina es la de 1 lazo de tiempo, pero sumando N veces el tiempo que tardará la parte X, y cambiando N por L.

$$T_{LAZO} = 1 \text{ ts} + 1 \text{ ts} + (1 \text{ ts} + 2 \text{ ts}) * (N-1) + 2 \text{ ts} + XN$$

$$T_{LAZO} = (4 + 3(N-1) + N(1 + 3L)) * ts$$

$$T_{LAZO} = (1 + 3N + N + 3LN) * ts$$

$$T_{LAZO} = (1 + 4N + 3NL) * ts$$

Fórmula para calcular el tiempo que tarda la subrutina de pérdida de tiempo de 2 lazos

Para el tiempo mínimo (N=L=1):

$$T_{LAZO} = (1 + 4(1) + 3(1)(1)) * t_s = 8 \mu s$$

Para el tiempo máximo (L=255):

$$T_{LAZO} = (1 + 4(255) + 3(255)(255)) * t_s = 196096 \mu s = 0.196096 s$$

Subrutina de pérdida de tiempo de tres lazos

movlw M;	1 ts	Loop1	decfsz contador1; 1 ts o 2 ts
movwf contador3;	1 ts	goto Loop1;	2 ts
Loop3        movlw N; 1 ts		decfsz contador2;	1 ts o 2 ts
movwf contador2;	1 ts	goto Loop2;	2 ts
Loop2        movlw L; 1 ts		decfsz contador3;	1 ts o 2 ts
movwf contador1;	1 ts	goto Loop3;	2 ts

Para este lazo haremos lo mismo que con la anterior, la parte en rojo será X, y la parte en negro será una subrutina de pérdida de tiempo de un lazo pero ahora cambiando L por M, y al final sumaremos M veces X.

$$T_{LAZO} = 1 t_s + 1 t_s + (1 t_s + 2 t_s) * (M-1) + 2 t_s + XM$$

$$T_{LAZO} = (4 + 3(M-1) + M(1 + 4N + 3NL)) * t_s$$

$$T_{LAZO} = (1 + 3M + M + 4MN + 3MNL) * t_s$$

$$T_{LAZO} = (1 + 4M + 4MN + 3MNL) * t_s$$

Fórmula para calcular el tiempo que tarda la subrutina de pérdida de tiempo de 3 lazos

Para el tiempo mínimo (M=N=L=1):

$$T_{LAZO} = (1 + 4(1) + 4(1)(1) + 3(1)(1)(1)) * t_s = 12 \mu s$$

Para el tiempo máximo (M=N=L=255):

$$T_{LAZO} = (1 + 4(255) + 4(255)(255) + 3(255)(255)(255)) * t_s = 50005246 \mu s = 50.005246 s$$

Ahora ya tenemos las fórmulas para poder calcular el tiempo que tarda cada subrutina de pérdida de tiempo, según las constantes L, N y M.

### **Aplicación de las subrutinas de pérdida de tiempo**

“Cronometro Segundero”

Una de las aplicaciones más importantes y demostrativas de las subrutinas de pérdida de tiempo son los cronómetros de los cuales en este aparato estudiaremos una estructura de un segundero.

Prácticamente un segundero es un contador módulo 60 ya que éste cuenta de uno en uno de 0 a 59. Tomando en cuenta esto y para poderlo mostrar en los LCD's tendríamos las siguientes secuencias en estos.

Para poderlo desarrollar utilizaremos los displays de la práctica TDM, el segundero lo mostraremos de forma centrada en los ocho LCD's como se muestra en la siguiente figura, al arrancar el sistema comenzara en 00 segundos y cada que transcurra un segundo este se irá incrementando en uno hasta

llegara 59, y estando en la cuenta máxima y transcurra un segundo más este volverá a contar de nuevo cuenta y así estará indefinidamente.

**Ciclo máquina:** El tiempo que tarda en ejecutarse un programa depende de la frecuencia del oscilador conectado al microcontrolador y del número de ciclos máquina ejecutados. Un ciclo de Máquina es la unidad básica de tiempo que utiliza el microcontrolador para el PIC 16F84, el ciclo máquina equivale a 4 ciclos de reloj, por lo tanto, el tiempo que tarda en producirse un ciclo de máquina es igual a cuatro veces el período del oscilador.

Las instrucciones en el microcontrolador PIC 16F877A necesitan 1 ciclo de máquina para ejecutarse, excepto las de salto (goto, call, btfss, btfsc, return, etc.) que necesitan de dos ciclos máquina. El tiempo que tarda el microcontrolador en ejecutar una tarea se determina por: Tiempo =  $4 \frac{1}{f} cm$  F= frecuencia del oscilador Cm= número de ciclos de máquina que tarda en ejecutar la tarea.

Ejemplo: Calcular la duración de 1 ciclo máquina para un PIC 16F84 que utiliza un cristal de cuarzo de 4 MHz.  $Tiempo = 4 \frac{1}{f} cm = 4 \frac{1}{4MHz} 1 = 1 \mu s$  Calcular el tiempo que tarda en ejecutarse la instrucción "call" si el sistema funciona con un cristal de cuarzo de 4MHz. La instrucción "call" dura 2 ciclos máquina.  $Tiempo = 4 \frac{1}{f} (2)cm = 4 \frac{1}{4MHz} (2) = 2 \mu s$

Ejemplo: En un sistema con microcontrolador PIC 16F877A y cristal de cuarzo de 4 MHz. Se desea generar un retardo de 1,5 ms. Calcular el número de ciclos máquina necesarios.  $Tiempo = 4 \frac{1}{f} cm \rightarrow cm = \text{tiempo} f$   $4 = 1500 \mu s$   $4MHz = 1500$  cm Con el cristal de 4 MHz. El período del oscilador será de  $0,25 \mu s$  Y el ciclo máquina tendrá una duración cuatro veces mayor,  $1 \mu s$ . Por lo tanto, para conseguir 1,5 ms serán necesarios 1500 ciclos máquina.

El MPLAB dispone de una opción de cronómetro denominada Stopwatch que permite medir el tiempo de ejecución de las instrucciones de los programas. El cronómetro Stopwatch calcula el tiempo basándose en la frecuencia de reloj del microcontrolador PIC que se está simulando. Es necesario fijar previamente la frecuencia del oscilador empleado, para eso, se activa desde el menú Debugger > Settings > Stopwatch, con esto se abre la ventana que muestra el tiempo transcurrido y los ciclos máquina empleados en la ejecución de cada instrucción.

**INSTRUCCIÓN “NOP”** La instrucción “nop” (No Operation) no realiza operación alguna. En realidad, consume un ciclo máquina sin hacer nada. Se utiliza para hacer gastar tiempo al microcontrolador sin alterar el estado de los registros ni de los “flags”. Esta instrucción tarda 1 ciclo máquina en ejecutarse.

**RETARDOS MEDIANTE LAZO SIMPLE** En muchas aplicaciones resulta necesario generar tiempos de espera, denominados tiempo de retardo. Estos intervalos pueden conseguirse mediante una subrutina de retardo, basada en un lazo simple de algunas instrucciones que se repiten tantas veces como sea necesario, hasta conseguir el retardo pretendido. Como el tiempo de ejecución de cada instrucción es conocido, lo único que hay que hacer es calcular el valor inicial que debe tener el registro R\_ContA, que actúa como contador del número de interacciones en el lazo para obtener el tiempo de retardo deseado.

Con todo esto en mente, pude llevara a cabo una extensión de lo que vimos anteriormente. O sea, lograr extender el programa para en vez de segundos, ahora llegase hasta horas y al posicionarse en 24 horas, 59 minutos y 59 segundos el reloj de tiempo real se reiniciará completamente para que la cuenta comenzara de nuevo desde cero.

Para que nosotros logremos que nuestro contador sea lo más parecido a la realidad necesitamos que el tiempo que se muestra el dato en el lcd sea de 1 segundo, para esto la subrutina de muestra el tiempo en el lcd debe durar 1 segundo.

En la práctica anterior habíamos dicho que cada carácter debe durar 2 milisegundos en el lcd correspondiente, para que la frecuencia de imagen de los lcds fuera aproximadamente de 60Hz.

```
;=====
;== Subrutina que muestra el tiempo en el lcd ==
;=====

muestra_mensaje    movlw time_corrimiento;
                     movwf Contador1;
barre_imagen       movf buffer8,w;
                     movwf portb;
                     bcf portc,Com_Disp7;
                     call retardo;
                     bsf portc,Com_Disp7;
                     call retardo;
                     movf buffer7,w;
→Así hasta llegar al buffer1...
                     movwf portb;
                     bcf portc,Com_Disp0;
                     call retardo;
                     bsf portc,Com_Disp0;
                     call retardo;
                     call retardo2;
                     decfsz Contador1,f;
                     goto barre_imagen;

return; Regresa de la subrutina.
```

Como podemos ver, esa es la subrutina que muestra el tiempo, cada buffer tiene 2 retardos, uno en el que permanece prendido el lcd y uno que permanece apagado, para que cada buffer tarde aproximadamente 2 milisegundos tenemos que hacer que cada retardo dure 1 milisegundo, y esto se logrará con la subrutina de pérdida de tiempo de dos lazos:

Para comprobar que esta subrutina tarda 1 milisegundo, hay que utilizar la fórmula de pérdida de dos lazos:

```
-----
;=====
;== Subrutina de retardo de 1 milisegundo ==
;=====

retardo            movlw .2; Mueve el valor 2 en decimal al registro W.
                     movwf Contador3; Mueve el contenido del registro W al registro
                     Contador3.
Loop2              movlw .165; Mueve el valor 165 en decimal al registro W.
                     movwf Contador2; Mueve el contenido del registro W al registro
                     Contador2.
Loop1              decfsz Contador2,f;Decremento en 1 al registro Contador2, brinca si es 0.
```

```
        goto Loop1; Ve a donde encuentres la etiqueta Loop1.  
        decfsz Contador3,f;Decremento en 1 al registro Contador3, brinca si es 0.  
        goto Loop2; Ve a donde encuentres la etiqueta Loop2.  
  
        return; Regresa de la subrutina.
```

---

$$T_{LAZO} = (1 + 4N + 3NL) * t_s$$

Si N=10 y L=32

$$T_{LAZO} = (1 + 4(10) + 3(32)(10)) * t_s = 1001 \mu s = 1.001 \text{ milisegundos}$$

Como vemos, nuestros cálculos nos dan un tiempo de 1001 microsegundos, o 1.001 milisegundos para cada retardo. Pero hay que sumarle los 2 microsegundos de la instrucción de return.

Pero si tomamos en cuenta las demás microinstrucciones de cada buffer, obtendremos en total 2014 microsegundos por cada uno de los buffers, y si lo multiplicamos por los 8 buffers, tendremos que la subrutina que muestra el tiempo dura 16,112 microsegundos, o 16.112 milisegundos.

Si queremos que tarde 1 segundo cada vez que se llama a esa subrutina entonces dividimos 1 segundo entre el tiempo que tarda:

$$1000000 \mu s / 16112 \mu s = 62.065$$

El número obtenido es el que le debemos asignar a la constante time\_corrimiento para que nuestra subrutina tarde 1 segundo. Pero al no poder poner valores con decimales tendremos que poner el valor entero al que más se acerca, en este caso 62, y ahora para calcular el tiempo que tardará con este numero haremos la siguiente operación:

$$(62) (16112) = 0.998,944 \text{ segundos}$$

Y el resultado es el tiempo en el que el microprograma mostrará el tiempo en los lcds.

Para mejorar aún más este tiempo, podemos poner una última subrutina de pérdida de tiempo después de todos los buffers, para que tarde un poco más y lleguemos a un mejor resultado.

Para calcular cuánto debe tardar esta subrutina hay que ver primero qué tiempo llevamos acumulado después de mostrar todos los buffers, que como lo calculamos anteriormente es de 16112 microsegundos.

Si repetimos 62 veces la subrutina, para que dure exactamente 1 segundo tendría de tardar un tiempo de:

$$1 \text{ segundo} / 62 = 16129.032 \text{ microsegundos}$$

Entonces necesitamos subir el valor de 16112 lo más cercano a 16129, esto se puede lograr con una subrutina de pérdida de tiempo de 1 lazo:

---

```
;  
=====  
== Subrutina de retardo==  
=====  
RetardoX      movlw .4; Mueve el valor 4 en decimal al registro W.
```

LoopA

movwf Contador2; Mueve el contenido del registro W al registro Contador2.  
decfsz Contador2,f;Decremento en 1 al registro Contador2, brinca si es 0.  
goto LoopA; Ve a donde encuentres la etiqueta Loop3.

return; Regresa de la subrutina.

-----  
Esta subrutina de 1 lazo tarda 13 microsegundos, pero hay que sumarle el tiempo de la instrucción return de 2 microsegundos.

Y para poder llamar a esta subrutina se utiliza una instrucción call, la cual añade otros 2 microsegundos.

Entonces, la implementación de este nuevo retardo añade en total 17 microsegundos, teniendo así un tiempo de 16129 microsegundos por cada vez que se repita la subrutina de mostrar el tiempo.

Si la multiplicamos por 62 que es el número de veces que se repite obtendremos:

$$(16129 \text{ seg}) (62) = 0.999998 \text{ segundos}$$

Este es un tiempo mucho más exacto que el que teníamos anteriormente, el cual se asemejará más a la realidad.

## Circuitos Monitor de pulso, Medición de temperatura y Sistema GPS

```
Archivo Edición Formato Ver Ayuda
#include <16f877a.h>
#fuses hs, nowdt
# use delay (clock=4M)
/*
#define lcd_rs_pin      pin_b0 // esto es en caso de que yo quiero especificar los pines que yo quiero utilizar
#define lcd_rw_pin      pin_b1
#define lcd_enable_pin  pin_b2
#define lcd_data4       pin_b4
#define lcd_data5       pin_b5
#define lcd_data6       pin_b6
#define lcd_data7       pin_b7
*/
#include <lcd.c>
void main () {
    set_tris_d(0); // esto es si en caso que yo quiero poner solo pins D COMO salida
    lcd_init();
    while (true){
        lcd_gotoxy (1,1); //Uvicamos la palabra en la primera fila y la primera columna
        printf(lcd_putc, "AutomaticCapsule"); // manifestamos la palabra en nuestro lcd
        lcd_gotoxy (1,2); //Uvicamos la palabra en la segunda fila y la primera columna
        printf(lcd_putc, "Dispenser");
        delay_ms (8000); // Le damos un retardo de 500 ms
        lcd_putc("\f"); // borramos la palabra para entrar a otra instrucción
        for (int car = 0;car<=16;car++){
            lcd_gotoxy(car,1);
            printf(lcd_putc, "PANDILLA");
            delay_ms (500);
            lcd_putc("\f");
        }
        for (car=16;car>=1;Car--){
            lcd_gotoxy(car,1);
            printf(lcd_putc, "PANDILLA");
            delay_ms (500);
            lcd_putc("\f");
        }
    }
}
```

```

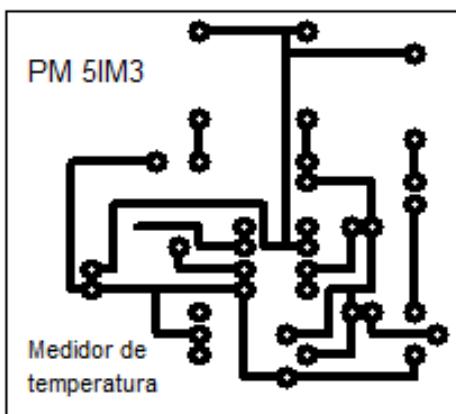
Archivo Edición Formato Ver Ayuda Archivo Edición Formato Ver Ayuda
#include <16F877a.h>
#fuses hs, nowdt
# use delay (clock=4M)
# include <lcd.c>
int A=1;
void main()
{
    lcd_init();
    set_tris_A(0x00);
    set_tris_B(0xFF);
    set_tris_C(0x00);
    output_A(0x00);
    output_C(0x00);
    set_tris_d(0);
    lcd_gotoxy (1,1);
    printf(lcd_putc, "AutomaticCapsule");
    lcd_gotoxy (1,2);
    printf(lcd_putc, "Dispenser");
    delay_ms(4000);
    printf(lcd_putc, "\f");
    do{
        if(!input(pin_B0)==1)
        {
            A--;
            delay_ms(10);
        }
        if(!input(pin_B1)==1)
        {
            A++;
            delay_ms(10);
        }

        if(A>3 && A==0)
        A=1;
        if(A==1)
        {
            printf(lcd_putc, "\f");
            lcd_gotoxy (1,1);
            printf(lcd_putc, "Cual activar");
            lcd_gotoxy (1,2);
            printf(lcd_putc, "1.A");
            lcd_gotoxy (6,2);
            printf(lcd_putc, "2.B");
            lcd_gotoxy (11,2);
            printf(lcd_putc, "3.C");
            delay_ms(500);

            if(!input(pin_B2)==1){
                output_high(pin_A1);
            }
            if(!input(pin_B3)==1){
                output_low(pin_A1);
            }
        }
        if(A==2)
        {
    } if(A==2)
    {
        if(A==2)
    }
}

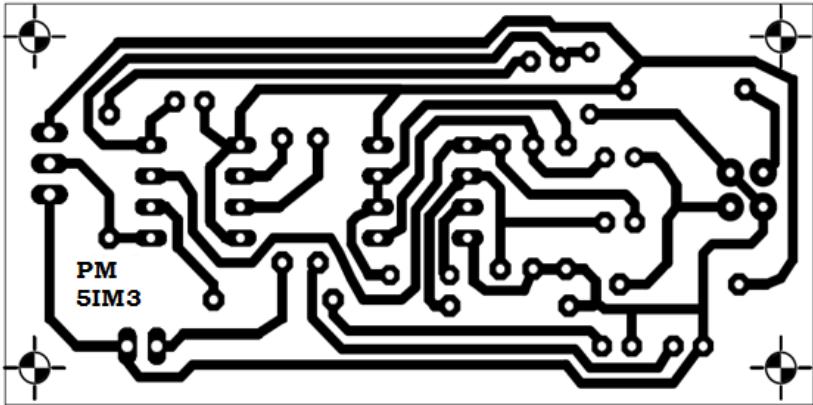
```

Mediante el uso de una SIM800L, un PIC 16F873A, un módulo UBLOX NEO6M y un regulador de voltaje. El regulador es usado para modular el V<sub>i</sub> de la GPS debido a que no se usan los mismos valores entre circuitos. Mediante dos LED's que sirven de testeo y al apretar tres veces el push button en una LCD comienzan a imprimirse la latitud y longitud. Mediante el uso de comandos AT para manejar la SIM800L mandará un mensaje a un número telefónico.



Para el sistema de medición de temperatura, usando un comparador, un regulador de voltaje, un sensor de temperatura, dos transistores y LED's podremos hacer lo siguiente.

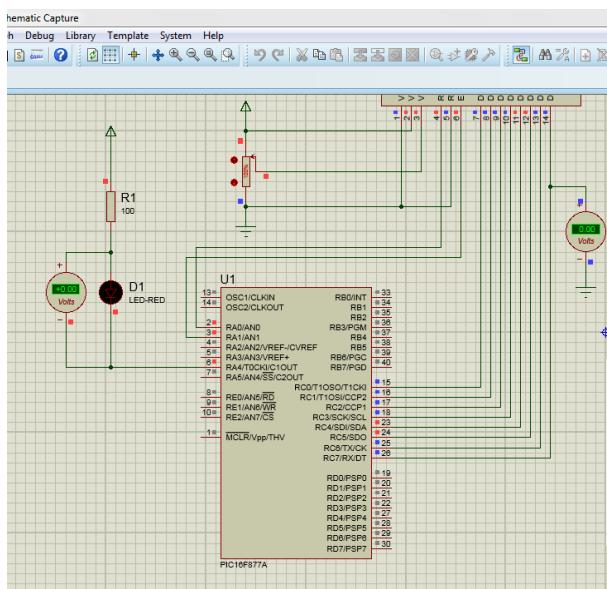
El regulador actuará de Set-Point conectado a la terminal negativa del comparador y a la positiva estará el sensor de temperatura. De esta manera los LED's se iluminarán alternativamente dependiendo de si la temperatura sea menor o mayor a la establecida por el regulador de voltaje



Por último, el sistema de medición de pulso cardíaco, en éste usaremos un componente de optoelectrónica como eje principal el cual es un CNY70, potenciómetro, circuito integrado LM358N y un par de LED's de dos colores. El circuito funciona de manera que los diodos del componente optoelectrónico estén activos constantemente, uno al emitir luz infrarroja y uno al recibirla, cuando algo se atraviesa en el recorrido

de este haz de luz ultravioleta los LED's calibrados con el potenciómetro que se mantienen apagados van a brillar. De esta forma decimos que al colocar un dedo en el circuito el flujo de la sangre producirá una interrupción en la continuidad de los diodos generando una señal al comparador integrado que encenderá los LED's.

## Apartado Práctico



Primero analizaremos los voltajes que nuestro PIC16F877A nos da como 0 lógico y 1 lógico.

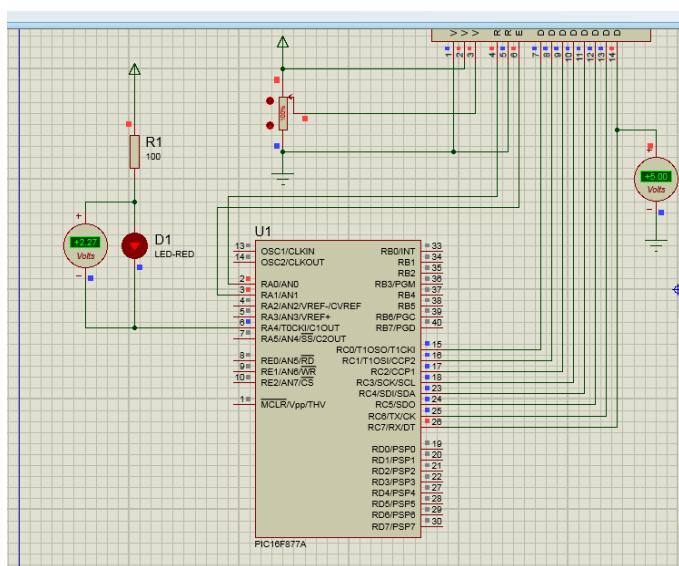
De acuerdo con la imagen el PIC nos da valores exactos de voltaje.

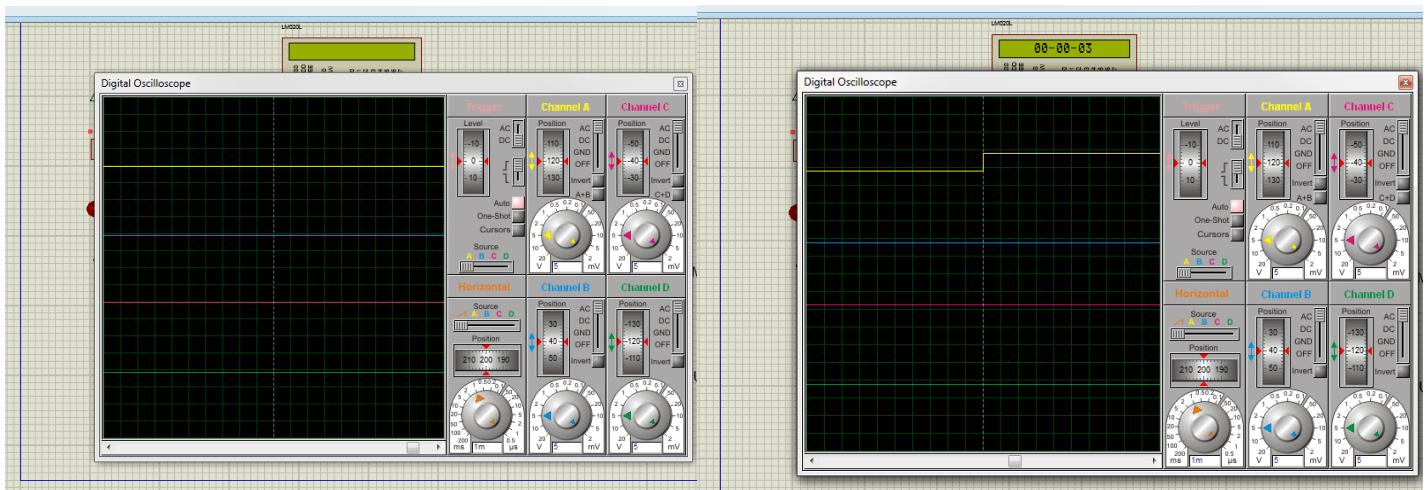
Ya que para un 0 lógico nos da 0.00 Volts.  
Y para un 1 lógico nos da un voltaje de 5 Volts.

Para la segunda parte analizaremos los voltajes que recibe la LCD, que para el caso de esta práctica esperamos que sean los mismos que da como salida el PIC.

Para un 0 lógico nos da 0.00 Volts.  
Para un 1 lógico nos da 5.00 Volts.

En esta parte pondremos el osciloscopio en una de las líneas de las constantes para ver si alcanzamos a ver cuándo el PIC activa y desactiva esa terminal. Para poder explicar lo que sucede en el osciloscopio, usaremos como ejemplo lo que pasa en el segundo 0, justo iniciando la simulación.





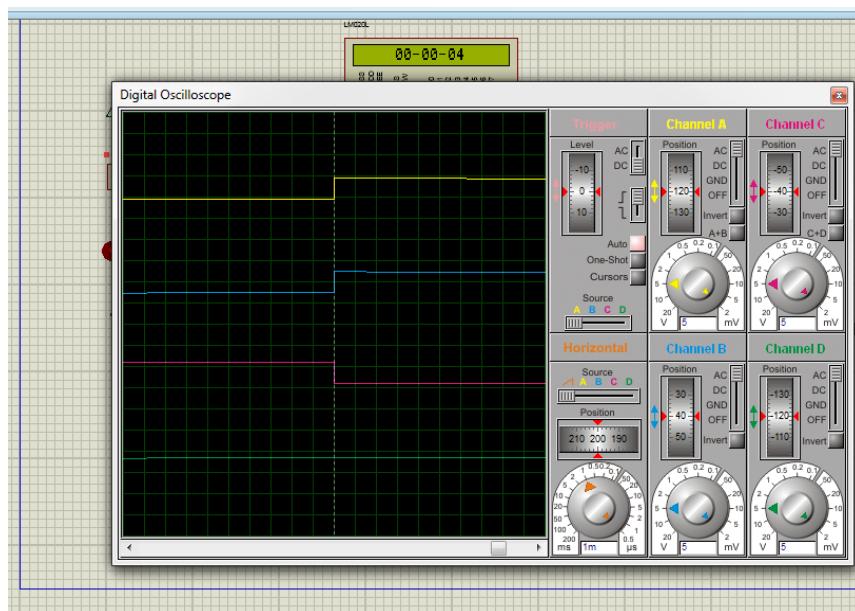
Como podemos ver, las 4 terminales del osciloscopio se conectan a las primeras 4 terminales del puerto b, el cual es el bus de datos.

Las terminales del osciloscopio coinciden con el segmento de lcd que miden, es decir, la terminal del osciloscopio A mide lo que sucede en el segmento a de los lcds, y así hasta la terminal D del osciloscopio.

Para el segundo 0, todos los lcds están en 0, excepto los de los guiones, y en el carácter 0, los segmentos a, b, c, y d de los lcds deben permanecer prendidos, y apagados los 4 en los guiones, es por eso que la gráfica del osciloscopio luce así, ya que primero hay 2 lcds apagados, luego un guion, después otros dos lcds apagados, luego otro guion y al final otros 2 lcds apagados, y se repite.

También podemos ver que cada pulso que manda el PIC, es de 2 milisegundos, ya que cada dato ocupa 2 cuadros de ancho y la perilla TIME/DIV está en 1 milisegundo.

Y el voltaje que manda el PIC es de 5 VOLTS, ya que los pulsos altos ocupan 1 cuadro de altura, y la perilla VOLT/DIV está en 5 Volts.

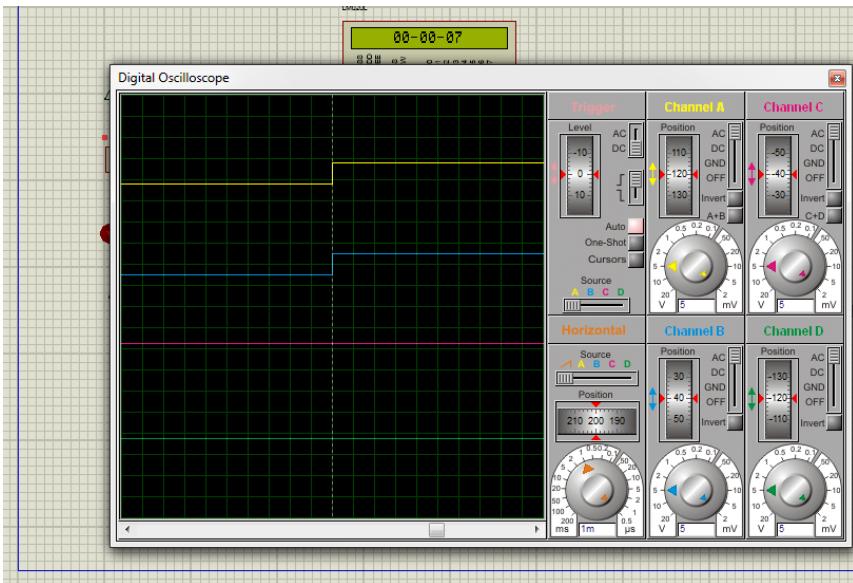


Esta es la última parte que analizaremos del sistema, en la cual pondremos el osciloscopio en algunas terminales del puerto C, para observar como es que el microcontrolador activa y desactiva los lcds.

En este caso no hay tanto conflicto para explicar lo que ocurre en el osciloscopio.

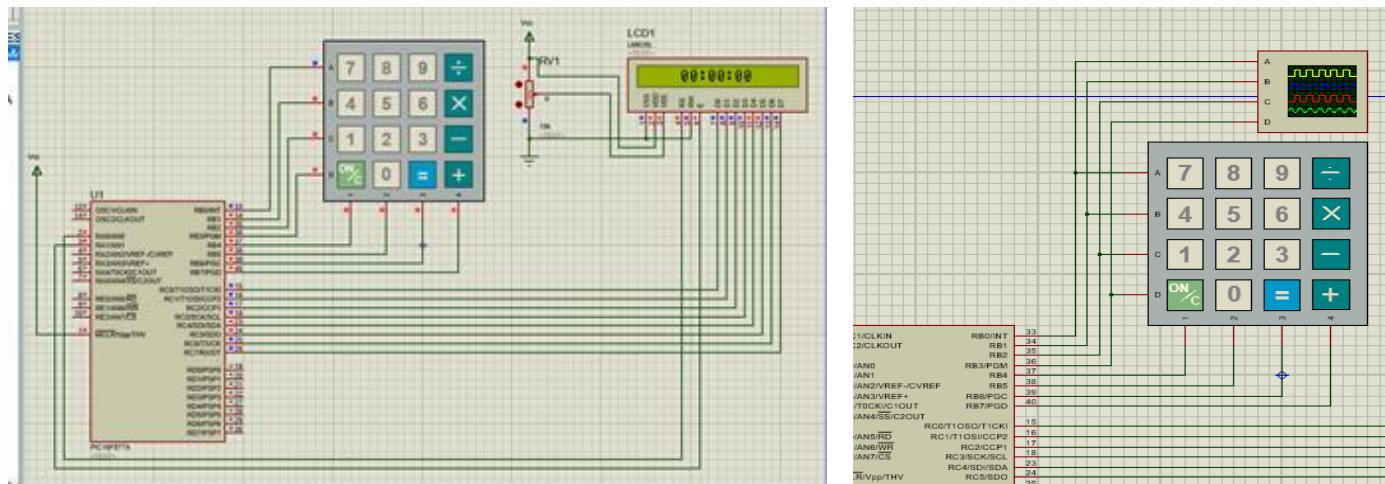
Primero explicaré como están conectadas las terminales del osciloscopio en el puerto c; la

terminal A está conectada al pin del puerto c que controla el común del lcd 8 (el último contando de derecha a izquierda), la terminal B a la del lcd 7, la terminal C a la del lcd 6 y la terminal D a la del lcd 5.



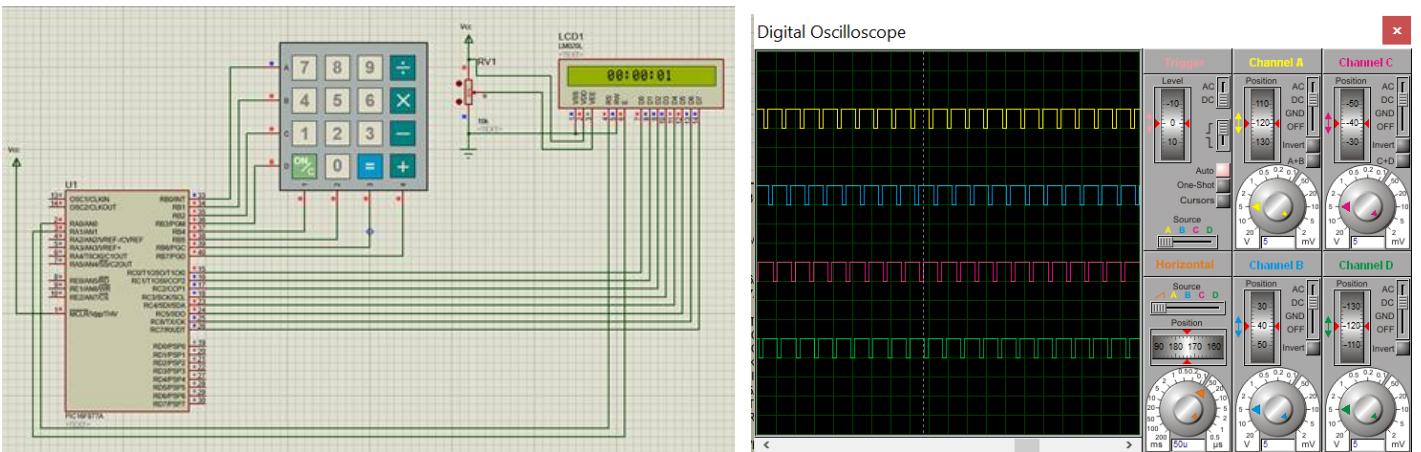
cuadro de ancho y la perilla VOLT/DIV está en 5 Volts, y cada que las activa dura 1 milisegundo, ya que el pulso mide 1 cuadro de ancho, y la perilla de TIME/DIV está en 1 milisegundo.

En la simulación primero empieza como tal el inicio de nuestro reloj de tiempo real, en el cual empieza en 00-00-00, porque aún no se le coloca una hora adecuada entonces inicia ahí. Nótese que las cifras de nuestro reloj están en medio de nuestra pantalla LCD 16x1.

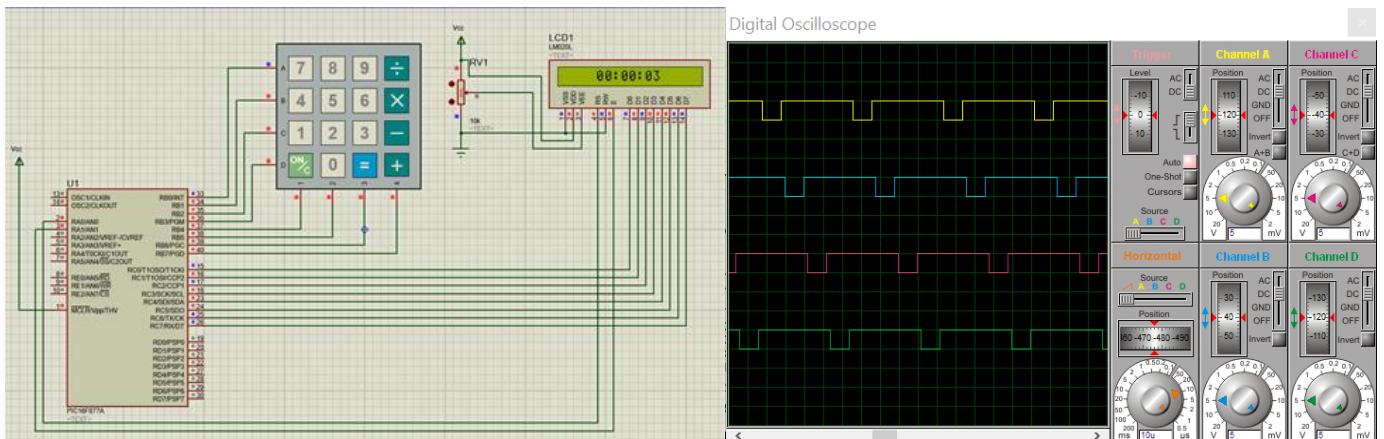
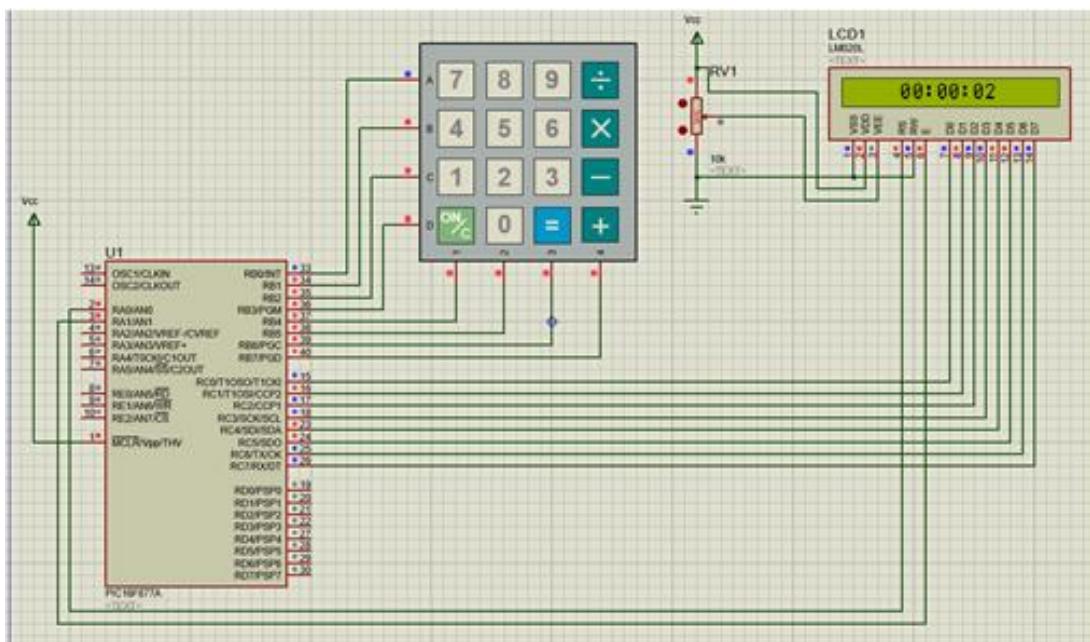


Ya comenzado empieza nuestro programa a contar como si fuera un reloj de verdad, de 00-00-00 pasa a 00-00-01, 00-00-02, 00-00-03, etc. funcionando como un reloj, cabe resaltar que la unidad de segundos, unidad de horas y unidad de minutos solo llegan hasta la cifra 9 mientras que la decena de segundos y decena de minutos llega hasta la cifra 6.

Como vimos en la subrutina de muestra de tiempo, el PIC va activando los lcds empezando por el 8 y terminando con el 1, y la gráfica del osciloscopio eso nos muestra, activa primero la terminal A, luego la B, después la C y al final la D, y ahí acaba ya que no hay más terminales para mostrar. Además, vemos que las activa con un pulso bajo (valor lógico “0”, es decir, 0 volts), ya que al activarlas el pulso mide 1

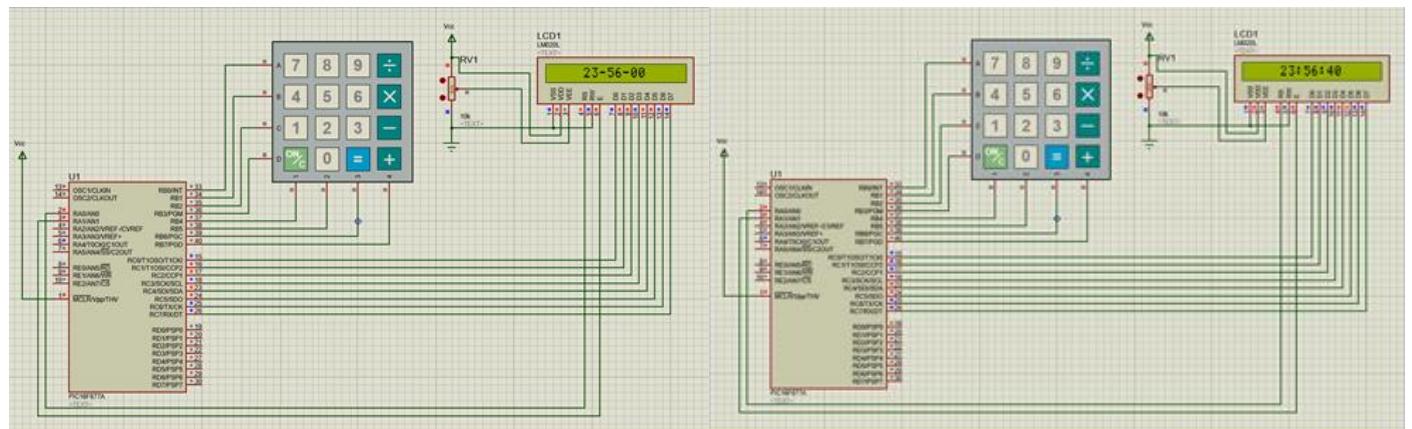
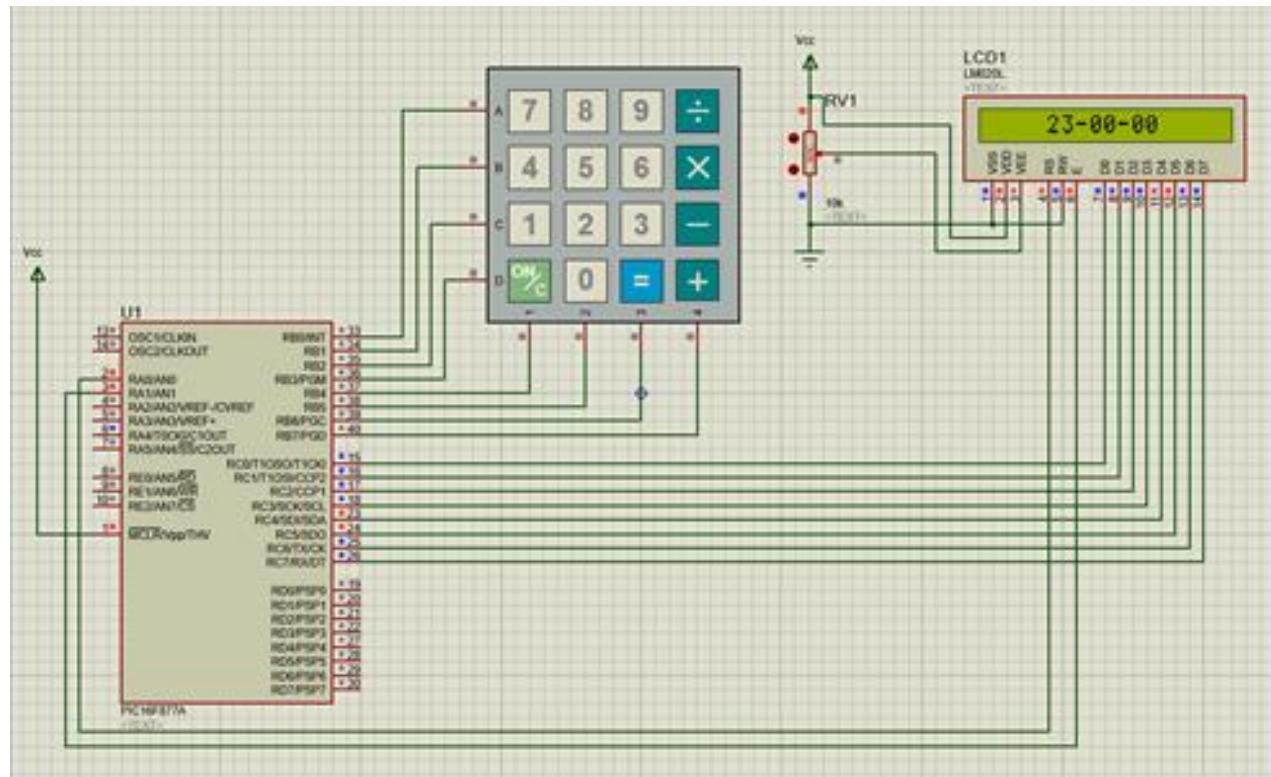


En cuanto la decena de horas esta llega hasta .2 y una vez alcanzado ese valor la unidad de horas llegara máximo a 3, siendo el ultimo dato 23-59-59, para pasar a 00-00-00, y volver a empezar.

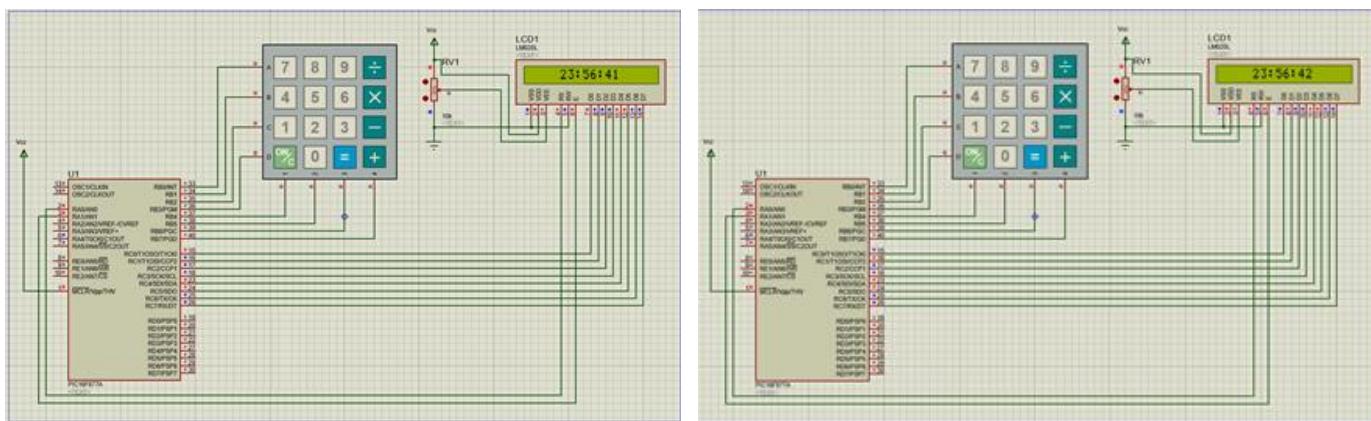


Esto ocurre hasta que apretáramos la tecla A de nuestro teclado, en ese momento el reloj parará y todo el programa principal también, y comenzarán los barridos del teclado, indicando que se está actualizando el tiempo.

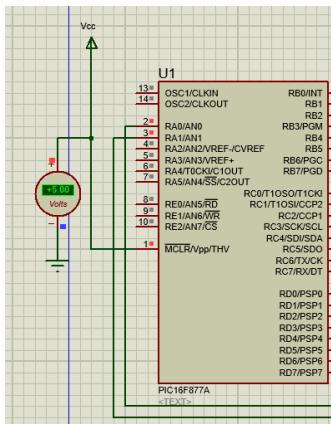
Empezando por decenas de horas, unidades de horas, decenas de minutos, unidades de minutos, decenas de segundos y finalmente unidades de segundos.



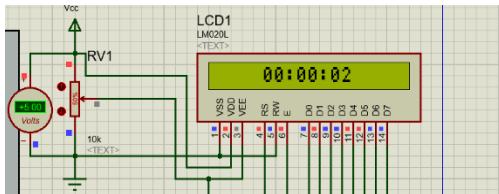
Y al apretar la tecla B del teclado matricial continuara con su conteo mostrando el n mero que sigue en nuestro reloj.



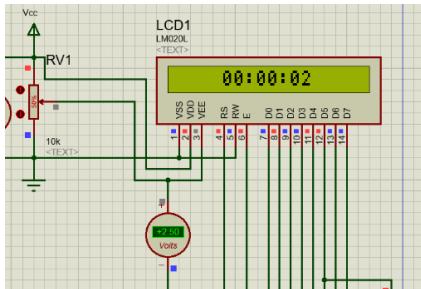
Concluyendo nuestro programa, pero fíjese que si al ingresar un numero no permitido el programa no lo pondrá ese número, por ejemplo, el caso de poner un 7 en la parte de decenas de horas o un 9 en decenas de segundos, en estos casos el programa no pondrá ese numero en la LCD porque no es un numero valido de nuestro programa.



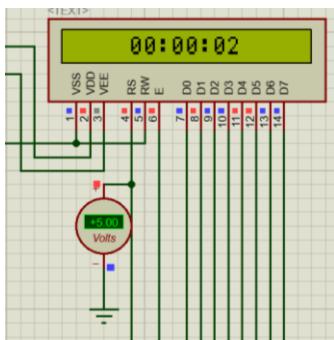
Voltaje de alimentación del circuito al Pic16F877A es de 5V



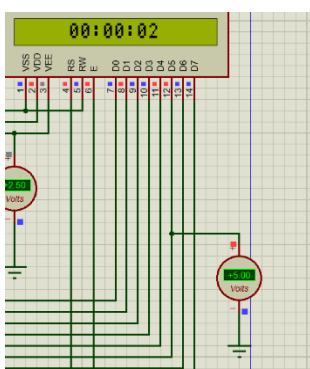
Voltaje de alimentación de la LCD 16x1 es de 5V



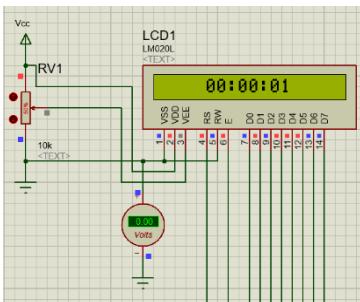
Voltaje de alimentación de la LCD 16x1es de 2.5V



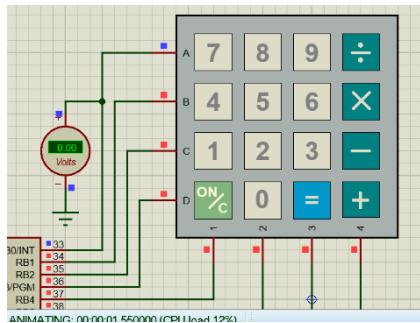
Voltaje RS para activación de comando es de 5V



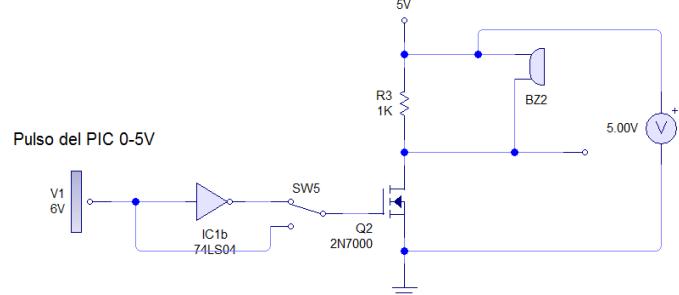
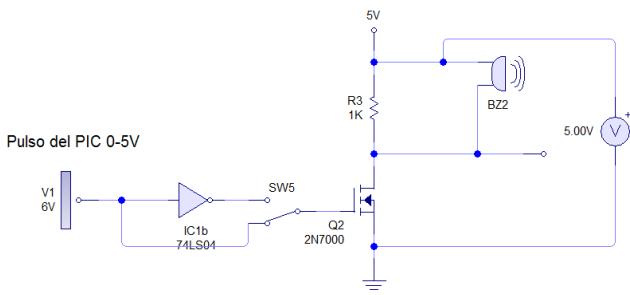
Voltaje D0-D7 de la LCD es de 5V



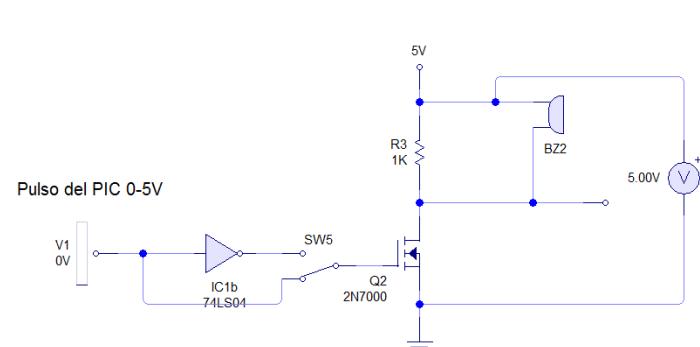
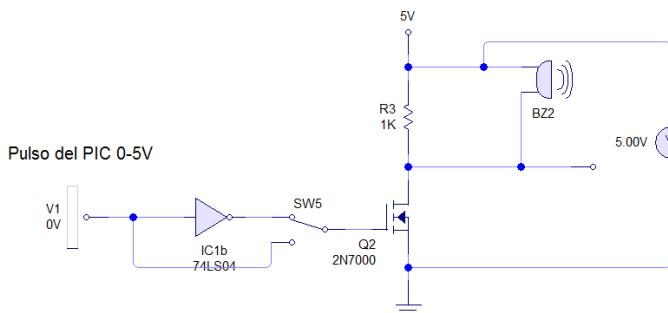
Voltaje pin RW de la LCD es 0V



Voltaje de barrido del teclado matricial es de 0V y de 5V

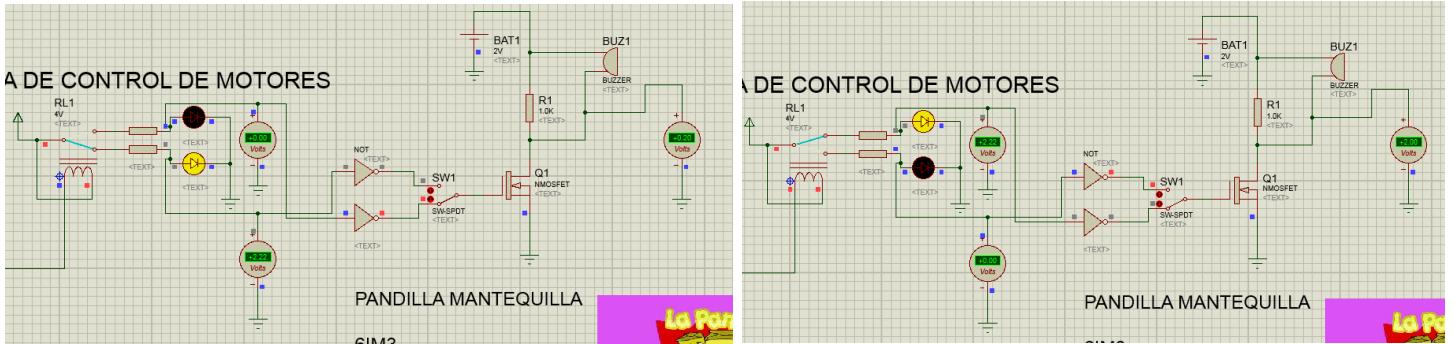


En estas dos capturas de tiempo podemos ver cómo el voltaje en estado alto activa la bocina y al cambiar de polo el switch la bocina se apaga. Todo esto con una sola señal de entrada



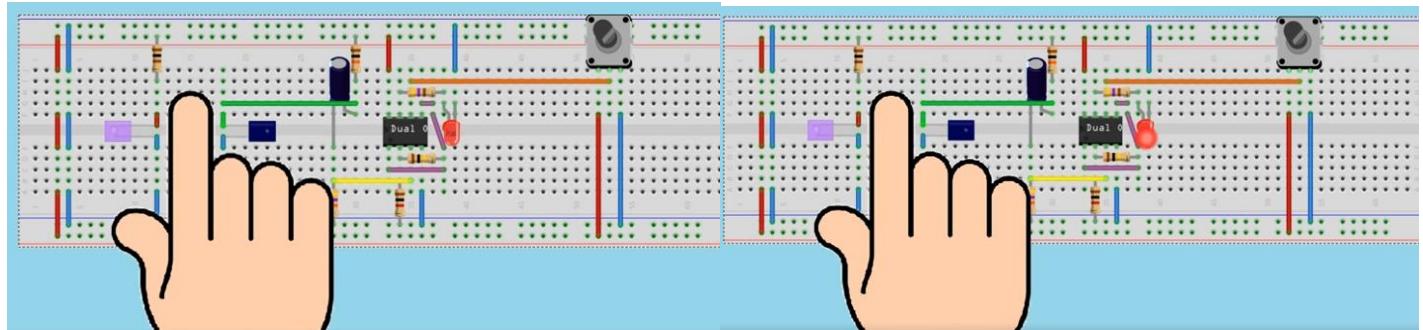
En estas dos figuras lo que cambió es el voltaje de entrada ahora en estado bajo por lo que para desactivar una vez más el circuito cambiamos de nuevo la posición del polo.

Nuevamente podemos ver cómo todo es a base de una sola señal de entrada y la NOT es la que hace posible la multiplexación y el proceso inverso se realiza con el switch.



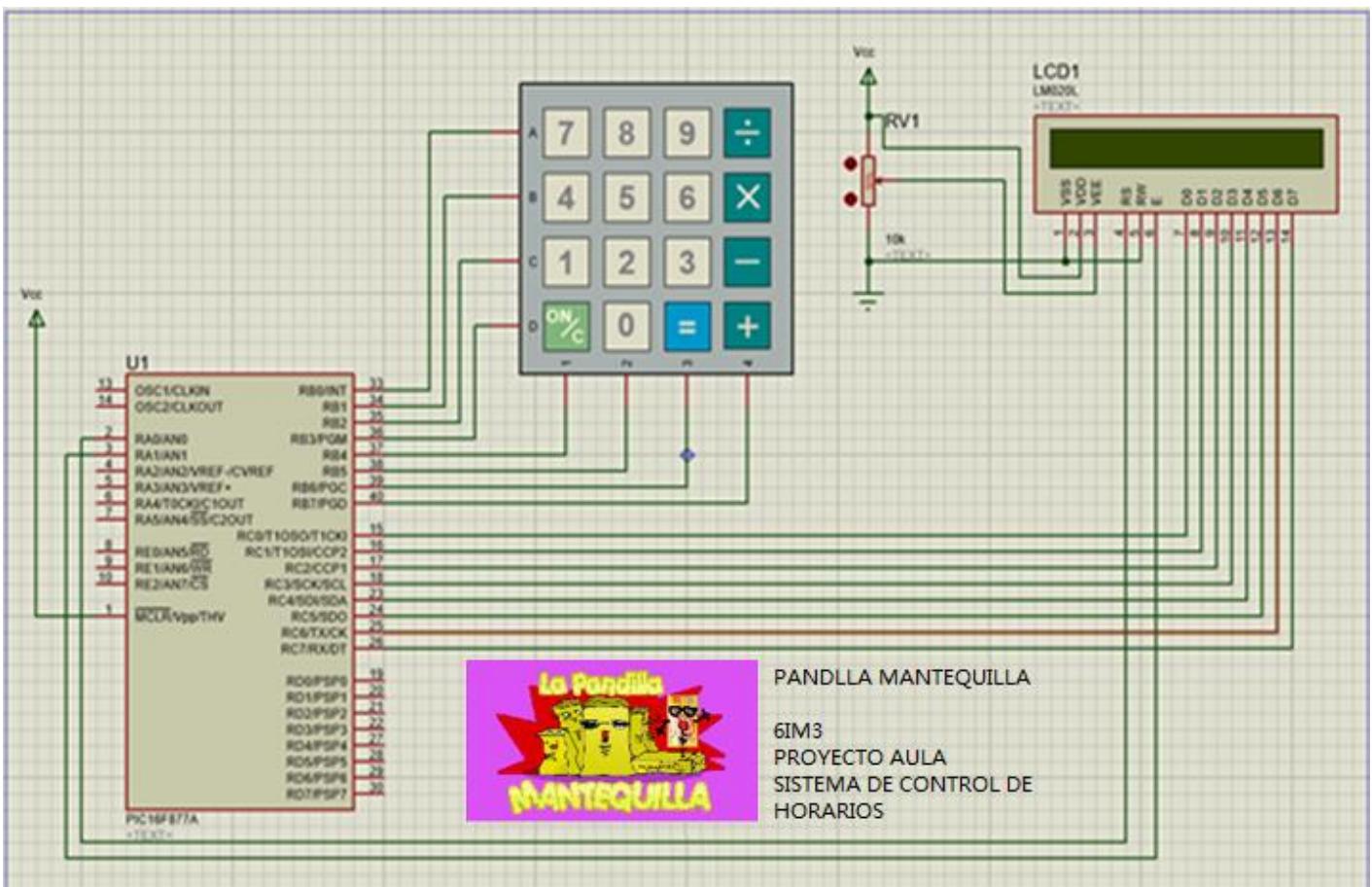
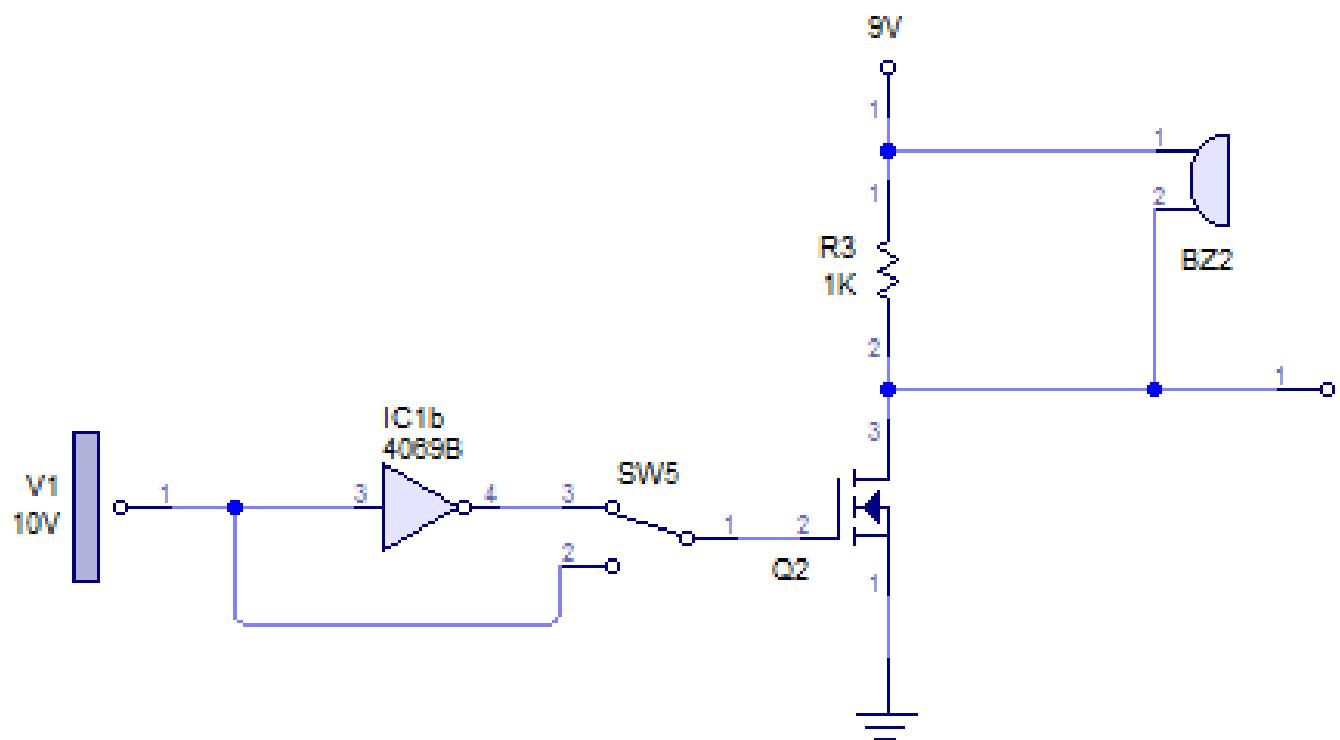
Esta es la presentación de las dos formas de conectar el sistema de alarma para que haga el cambio y pueda ser desactivado hasta el próximo impulso del microcontrolador. En el ejemplo uno podemos ver que sólo hay una NOT, eso se debe a que está conectada directamente a la salida del microcontrolador que produce las interrupciones cada n horas sin embargo en las simulaciones nos causó algunos problemas debido a que el cambio de polos en el interruptor afectaba en funcionamiento del PIC a pesar de estar configurado como salida ese pin. Por lo que decidimos conectarlo a otras dos señales que no estuvieran directamente conectadas al PIC pero sí recibieran las interrupciones al mismo tiempo que el sistema de alarma y por descontado elegimos al sistema de control de motores. Dado que son 4 podemos elegir sólo dos de ellos y así con dos entradas, dos NOT y el mismo interruptor de tres líneas dos polos un tiro pudimos hacer que el circuito funcionara de la misma manera que el relé sólo que sin hacer interferencia con las señales de salida en el microcontrolador y amplificando el tiempo de vida de los circuitos en general.

Nota: La conexión del segundo circuito al sistema de control de motores es de forma paralela por lo que no afecta en absoluto el funcionamiento de ninguno de los dos a diferencia de la conexión serie del primero entre PIC y alarma.

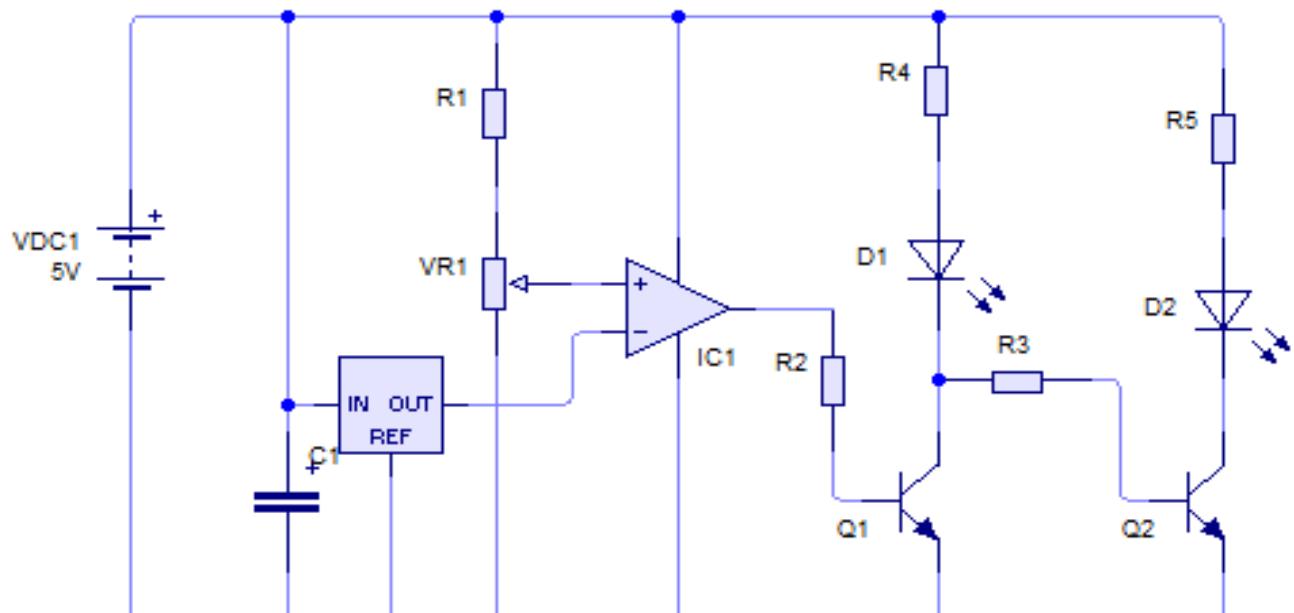


Como dijimos anteriormente, el LED se encenderá cuando haya un obstáculo en la transmisión y recepción de los diodos como el flujo sanguíneo.

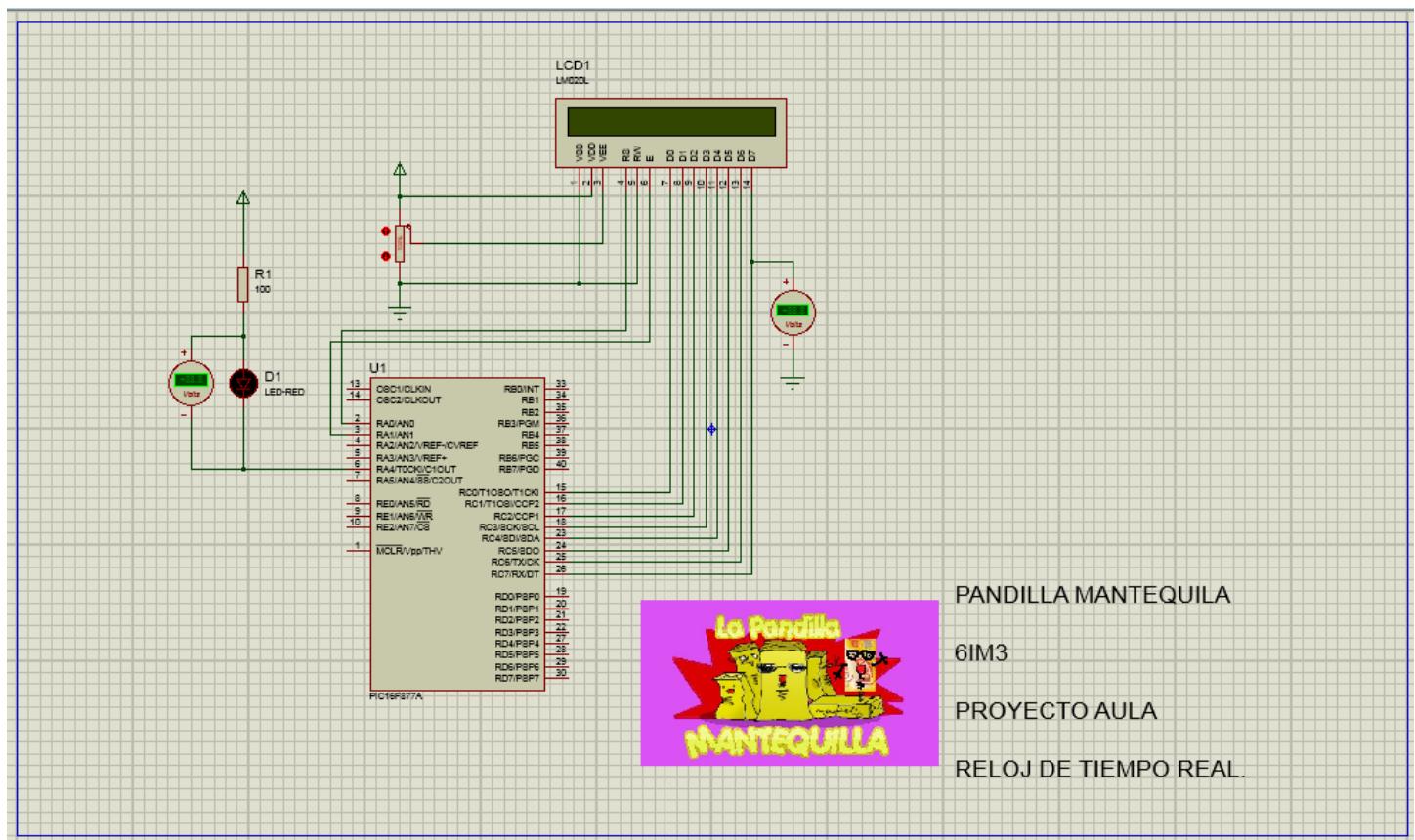
## Diagramas esquemáticos

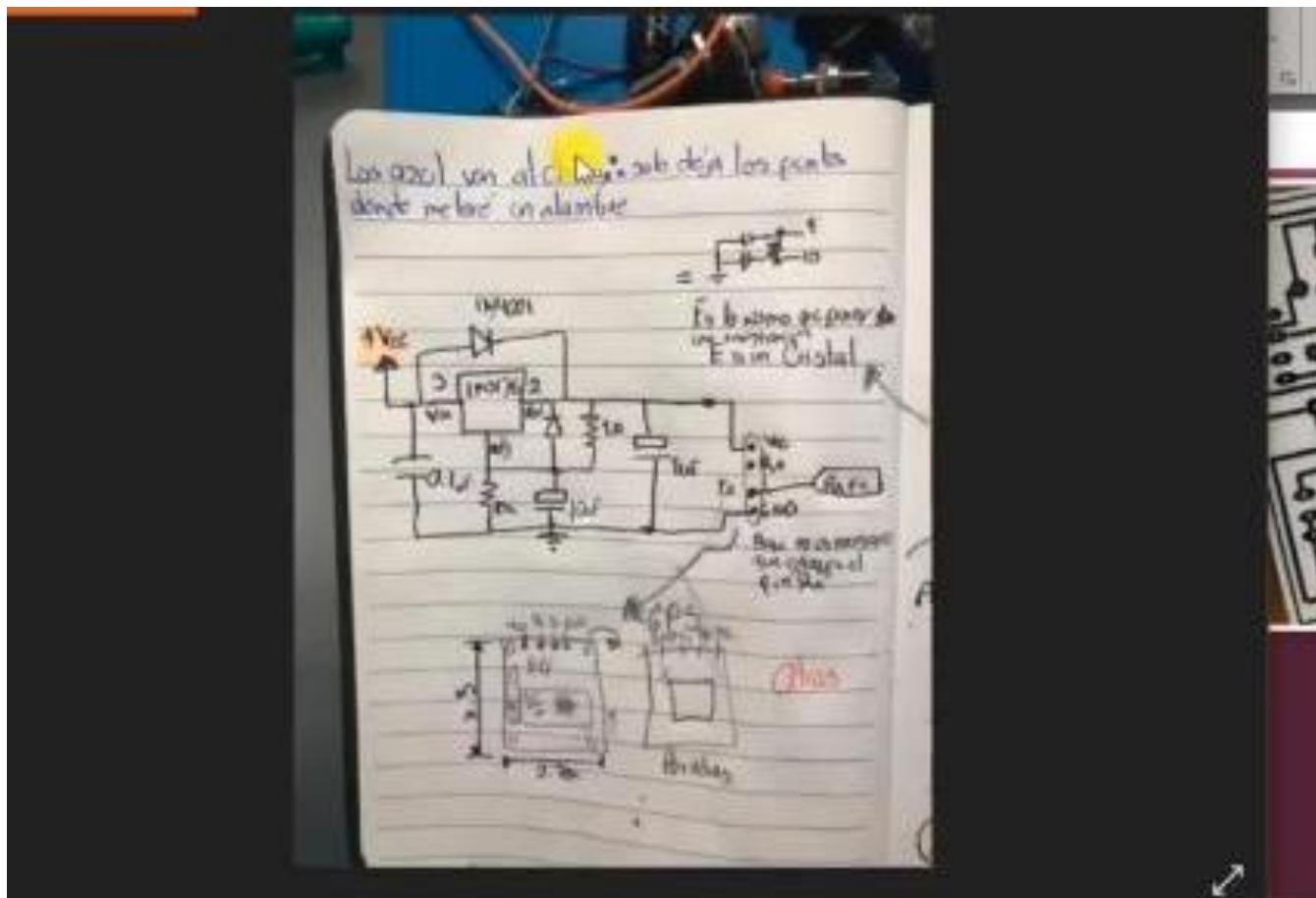
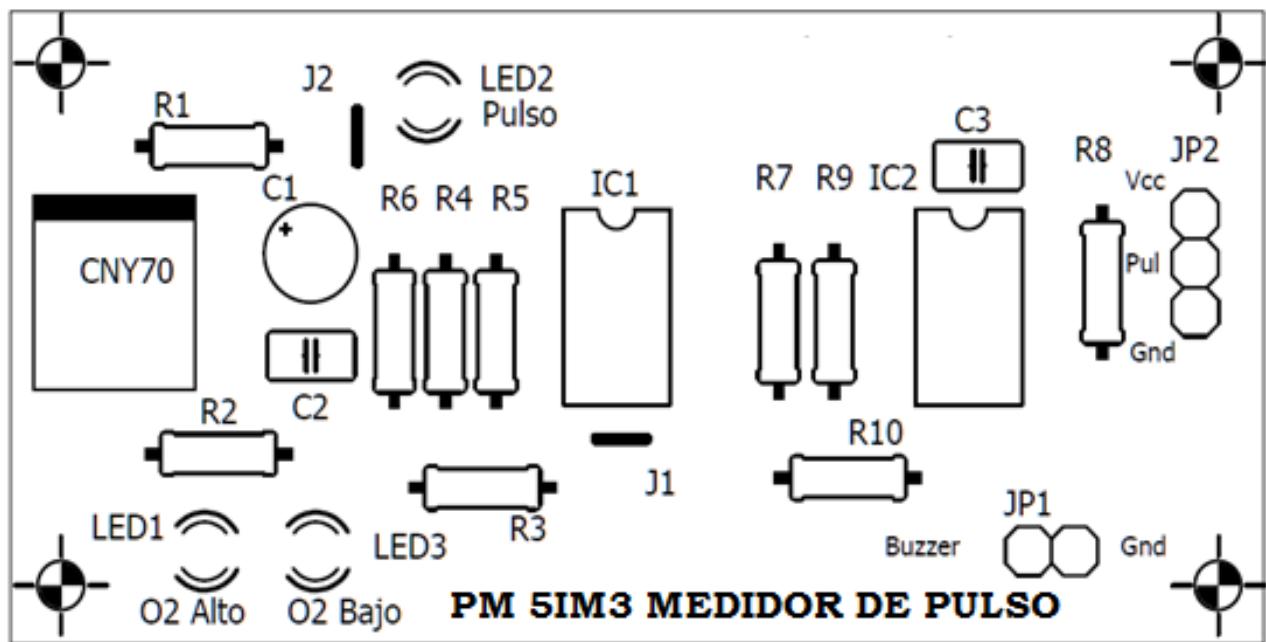


## PANDILLA MANTEQUILLA

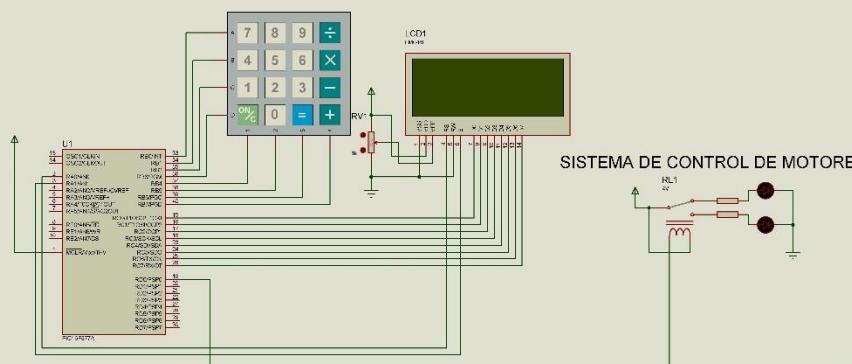


Circuito de medición de temperatura

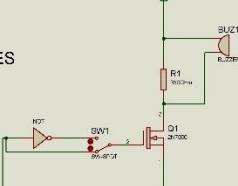




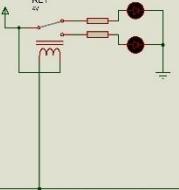
### SISTEMA DE CONTROL DE HORARIO



### SISTEMA DE CONTROL DE ALARMA



### SISTEMA DE CONTROL DE MOTORES



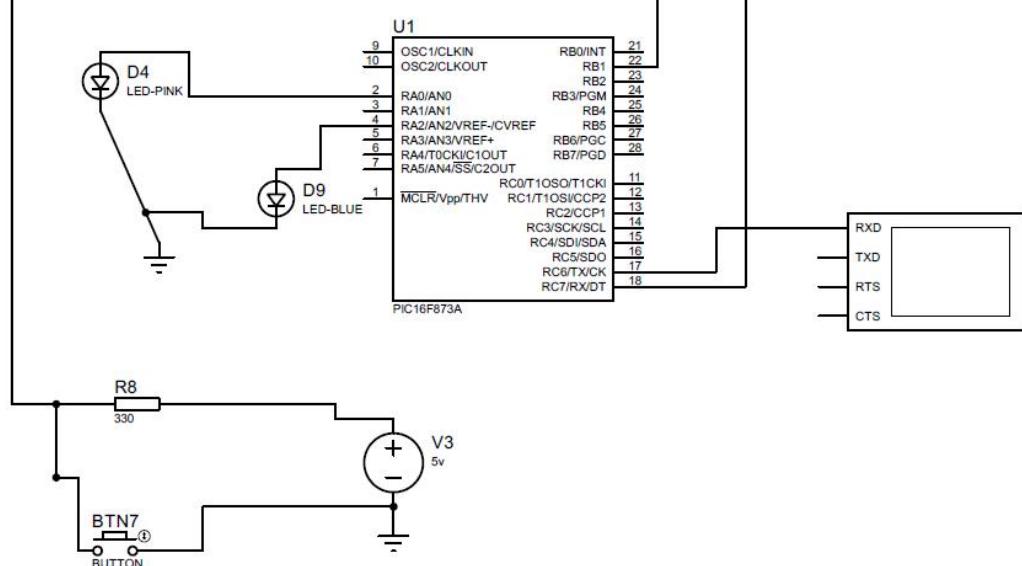
PANDILLA MANTEQUILLA  
6IM3  
PROYECTO AULA  
CIRCUITOS DE CONTROL



### GPS2

TX	54.071125,N
RX	1.995949,W
(I) (I) 9	

VGPS



## Códigos fuente

; INSTITUTO POLITECNICO NACIONAL.

; CECYT 9 JUAN DE DIOS BATIZ.

;

; PROYECTO AULA.

; LA PANDILLA MANTEQUILLA.

;

; GRUPO: 6IM3.

;

; INTEGRANTES:

; ALVARADO RODRÍGUEZ FERNANDO BRYAN

; AYALA BERNAL ISRAEL

; BARRERA CASTILLO ERICK ABRAHAM

; HERNÁNDEZ CABALLERO MAURICIO

; SÁNCHEZ MARTÍNEZ FELIPE

; VILLEGRAS MONROY EMILIO

;

; FECHA DE ENTREGA DEL REPORTE:

;

; EL PROGRAMA SERÁ EL MENÚ DE NUESTRO PASTILLERO, EL

; CUAL CON UN DISPOSITIVO DE SALIDA (LCD 20x4) SE PODRÁ

; VER EL MENÚ, Y CON UN DISPOSITIVO DE ENTRADA (TECLADO

; MATRICIAL 4x4) EL USUARIO PODRÁ INTERACTUAR CON

; NUESTRO SISTEMA.

-----;

;

List p=16f877A;

#include "c:\Program files (x86)\Microchip\Mpasm Suite\p16f877a.inc";

\_CONFIG \_CP\_OFF & \_WDT\_OFF & \_BODEN\_OFF & \_PWRTE\_ON & \_XT\_OSC & \_WRT\_OFF & \_LVP\_OFF  
& \_CPD\_OFF;

-----;

;

```

; Fosc = 4 MHz.

; Ciclo de trabajo del PIC = (1/fosc)*4 = 1 µs.

; T int =(256-tmr0)*(P)*((1/4000000)*4) = 1 ms. // Tiempo de interrupción.

; tmr0=131, P=8.

; frec int = 1/ t int = 1 KHz.

;-----;

;Def. de variables del programa en RAM.

resp_w equ 0x20; Registro de respaldo de w.

resp_status equ 0x21; Registro de respaldo del registro de las banderas de la ALU.

res_pcclath equ 0x22; Registro de respaldo del registro pcclath.

res_fsr equ 0x23; Registro de respaldo del registro fsr.

presc_1 equ 0x24; Registro del prescalador 1. .001 100
10

presc_2 equ 0x25; Registro del prescalador 2. t int = t intb * presc_1 * presc_2

banderas equ 0x26; Registro de las banderas del PIC.

cont_milis equ 0x27; Variable utilizada para el tiempo de espera de la lcd.

cta_uniseg equ 0x28; Variable utilizada para contabilizar las unidades de segundo en binario.

cta_decseg equ 0x29; Variable utilizada para contabilizar las decenas de segundo en binario.

cta_unimin equ 0x30; Variable utilizada para contabilizar las unidades de minuto en binario.

cta_decmin equ 0x31; Variable utilizada para contabilizar las decenas de minuto en binario.

cta_unihor equ 0x32; Variable utilizada para contabilizar las unidades de hora en binario.

cta_dechor equ 0x33; Variable utilizada para contabilizar las decenas de hora en binario.

temporal equ 0x34; Registro utilizado para llevar los valores de la cuenta a la tabla de
conversión a ascii.

buffer8 equ 0x35; Registro utilizado para mostrar el valor de cta_dechor en la lcd.

buffer7 equ 0x36; Registro utilizado para mostrar el valor de cta_unihor en la lcd.

buffer5 equ 0x38; Registro utilizado para mostrar el valor de cta_decmin en la lcd.

buffer4 equ 0x39; Registro utilizado para mostrar el valor de cta_unimin en la lcd.

buffer2 equ 0x41; Registro utilizado para mostrar el valor de cta_decseg en la lcd.

buffer1 equ 0x42; Registro utilizado para mostrar el valor de cta_uniseg en la lcd.

Var_teclado equ 0x43; Variable utilizada para almacenar la tecla oprimida en el puerto b.

```

resp_uniseg usuario.	equ	0x44; Variables para guardar el valor de las unidades de minuto que estableció el
resp_decseg usuario.	equ	0x45; Variables para guardar el valor de las unidades de minuto que estableció el
resp_unimin usuario.	equ	0x46; Variables para guardar el valor de las unidades de minuto que estableció el
resp_decmi nusuario.	equ	0x47; Variables para guardar el valor de las unidades de minuto que estableció el
resp_unihor usuario.	equ	0x48; Variables para guardar el valor de las unidades de minuto que estableció el
resp_dechor usuario.	equ	0x49; Variables para guardar el valor de las unidades de minuto que estableció el

cont\_segundos equ 0x59; Variable que cuenta segundos.

-----

;Constantes

No\_haytecla equ 0xF0; Código que pertenece al no haber oprimido ninguna tecla.

Tec\_7 equ 0xE0; Código de la tecla 1.

Tec\_8 equ 0xD0; Código de la tecla 2.

Tec\_9 equ 0xB0; Código de la tecla 3.

Tec\_A equ 0x70; Código de la tecla A.

Tec\_4 equ 0xE0; Código de la tecla 4.

Tec\_5 equ 0xD0; Código de la tecla 5.

Tec\_6 equ 0xB0; Código de la tecla 6.

Tec\_1 equ 0xE0; Código de la tecla 7.

Tec\_2 equ 0xD0; Código de la tecla 8.

Tec\_3 equ 0xB0; Código de la tecla 9.

Tec\_0 equ 0xD0; Código de la tecla 0.

Tec\_ON equ 0xE0; Código de la tecla ON.

Tec\_igual equ 0xB0; Código de la tecla =.

; banderas del registro banderas.

ban\_int equ .0; Bit0 de la bandera de interrupción.

```

sin_bd1      equ    .1; Bit1 de la bandera de interrupción.
sin_bd2      equ    .2; Bit2 de la bandera de interrupción.
sin_bd3      equ    .3; Bit3 de la bandera de interrupción.
sin_bd4      equ    .4; Bit4 de la bandera de interrupción.
sin_bd5      equ    .5; Bit5 de la bandera de interrupción.
sin_bd6      equ    .6; Bit6 de la bandera de interrupción.
sin_bd7      equ    .7; Bit7 de la bandera de interrupción.

;-----  

; Def. de Ptos. I/O.  

;  

; Puerto A.  

RS_LCD        equ    .0; Bit que controla el modo de operación de la lcd.
Enable_LCD    equ    .1; Bit que controla el Enable de la lcd.
Sin_UsoRA2    equ    .2; Bit2 sin uso del puerto a.
Sin_UsoRA3    equ    .3; Bit3 sin uso del puerto a.
Sin_UsoRA4    equ    .4; Bit4 sin uso del puerto a.
Sin_UsoRA5    equ    .5; Bit5 sin uso del puerto a.

;  

proga        equ B'1111100'; Definición de la configuración de los bits del puerto A.
;  

;Puerto B.  

Act_Ren00     equ    .0; Bit para la activación del renglón 0.
Act_Ren11     equ    .1; Bit para la activación del renglón 1.
Act_Ren22     equ    .2; Bit para la activación del renglón 2.
Act_Ren33     equ    .3; Bit para la activación del renglón 3.
Col_1         equ    .4; Bit de entrada para leer el código de la tecla oprimida de la columna 1.
Col_2         equ    .5; Bit de entrada para leer el código de la tecla oprimida de la columna 2.
Col_3         equ    .6; Bit de entrada para leer el código de la tecla oprimida de la columna 3.
Col_4         equ    .7; Bit de entrada para leer el código de la tecla oprimida de la columna 4.

;  

probgb       equ b'11110000'; Programación inicial del puerto B.
;  

;Puerto C.

```

```

D0_LCD      equ .0; Bit 0 de datos o comandos para la LCD.
D1_LCD      equ .1; Bit 1 de datos o comandos para la LCD.
D2_LCD      equ .2; Bit 2 de datos o comandos para la LCD.
D3_LCD      equ .3; Bit 3 de datos o comandos para la LCD.
D4_LCD      equ .4; Bit 4 de datos o comandos para la LCD.
D5_LCD      equ .5; Bit 5 de datos o comandos para la LCD.
D6_LCD      equ .6; Bit 6 de datos o comandos para la LCD.
D7_LCD      equ .7; Bit 7 de datos o comandos para la LCD.
;

progc      equ b'00000000'; Programación inicial del puerto C como Entrada.
;

;Puerto D.

Activador    equ .0; Bit que activa el dispensamiento.
Sin_UsoRD1   equ .1; Bit1 sin uso del puerto D.
Sin_UsoRD2   equ .2; Bit2 sin uso del puerto D.
Sin_UsoRD3   equ .3; Bit3 sin uso del puerto D.
Sin_UsoRD4   equ .4; Bit4 sin uso del puerto D.
Sin_UsoRD5   equ .5; Bit5 sin uso del puerto D.
Sin_UsoRD6   equ .6; Bit6 sin uso del puerto D.
Sin_UsoRD7   equ .7; Bit7 sin uso del puerto D.
;

progd      equ b'11111110';Definición de la configuración del puerto D.
;

; Puerto E.

Sin_UsoRE0   equ .0; Bit0 sin uso del puerto E.
Sin_UsoRE1   equ .1; Bit1 sin uso del puerto E.
Sin_UsoRE2   equ .2; Bit2 sin uso del puerto E.
;

proge      equ b'111'; Definición de la configuración de los bits del puerto E.
;-----
=====;
== Vector Reset =
=====;

```

org 0000h; Ve a la página 0 de memoria.

vec\_reset clrf pclath; Asegura la página 0 de memoria.  
goto prog\_prin; Ve a prog\_prin.

-----  
;=====

==== Vector de interrupción ==

=====

org 0x0004; Ve a la dirección de memoria donde se encuentra el vector de interrupción.

vec\_int movwf resp\_w; Respalda el estado del registro w.  
movf status,w; Respalda el estado del registro  
movwf resp\_status; de las banderas de la ALU.  
clrf status; Borra el registro status.  
movf pclath,w; Respalda el estado  
movwf res\_pclath; del registro pclath.  
clrf pclath; Borra el registro pclath.  
movf fsr,w; Respalda el estado  
movwf res\_fsr; del registro fsr.

btfsc intcon,t0if; Prueba el bit t0if, del registro intcon, brinca si está en 0.  
call rutina\_int; Llama a la rutina de interrupciones.

sal\_int movlw .131; Mueve el valor 131 en binario al registro w.  
movwf tmr0; Carga el registro tmr0 con w.

movf res\_fsr,w; Restaura el contenido  
movwf fsr; del registro fsr.  
movf res\_pclath,w; Restaura el contenido  
movwf pclath; del registro pclath.  
movf resp\_status,w; Respalda el contenido  
movwf status; del registro status.  
movf resp\_w,w; Restaura el contenido del registro w.

retfie; Regresa de la subrutina de interrupción.

;

---

=====

== Subrutina de Interrupciones ==

=====

rutina\_int        incf cont\_milis,f; Incrementa en 1 el contenido del registro cont\_milis.

                  incf presc\_1,f; Incrementa en 1 el contenido del registro presc\_1.

                  movlw .100; Mueve el valor 100 en binario al registro w.

                  xorwf presc\_1,w; OR-EXCLUSIVO entre los registros w y presc\_1.

                  btfsr status,z; Prueba el bit z del registro status, brinca si esta en 0.

                  goto sig\_int; Ve a sig\_int.

                  goto sal\_rutint; Ve a sal\_rutint.

sig\_int        clrf presc\_1; Borra el contenido del registro presc\_1.

                  incf presc\_2,f; Incrementa en 1 el contenido del registro presc\_2.

                  movlw .10; Mueve el valor 10 en binario al registro w.

                  xorwf presc\_2,w; OR-EXCLUSIVO entre los registros w y presc\_2.

                  btfsr status,z; Prueba el bit z del registro status, brinca si esta en 1.

                  goto sal\_rutint; Ve a sal\_rutint.

                  clrf presc\_1; Borra el contenido del registro presc\_1.

                  clrf presc\_2; Borra el contenido del registro presc\_2.

                  incf cont\_segundos,f; incrementa en 1 el contenido del registro cont\_segundos.

sal\_rutext    bsf banderas,ban\_int; Pon a 1 el bit ban\_int del registro banderas.

sal\_rutint    bcf intcon,t0if; Pon a 0 el bit t0if del registro intcont.

                  return; Regresa de la subrutina.

;

---

=====

== Subrutina de Ini. de Reg. del Pic ==

=====

prog\_ini

bsf status,RP0; Ponte en el banco 1 de ram.

movlw 0x02; Mueve el valor 82 en hexadecimal al registro w.

movwf option\_reg ^0x80; Desabilita pullups y selecciona un prescalador de 8 al TMR0.

movlw proga; Configura los bits del puerto A RS\_LCD y Enable\_LCD

movwf trisa ^0x80; como salidas, y los demás como entradas.

movlw probg; Configura los bits del nibble alto del puerto B

movwf trisb ^0x80; como salidas, y los del nibble bajo como entradas.

movlw progc; Configura los bits del puerto C

movwf trisc ^0x80; como salidas.

movlw progd; Configura el bit Activador del puerto D

movwf trisd ^0x80; como salida y los demás como entradas.

movlw proge; Configura los bits del puerto E

movwf trise ^0x80; como salida.

movlw 0x06; Configura los bits del puerto A y E

movwf adcon1 ^0x80; como entradas o salidas digitales.

bcf status,RP0; Ponte en el banco 0 de ram.

movlw 0xa0; Habilita el sobreflujo del TMR0

movwf intcon; y las interrupciones globales.

movlw .131; Activa la interrupción

movwf tmr0; cada milisecondo.

clrf banderas; Borra el contenido del registro banderas.

clrf portc; Limpia el bus de datos.

movlw 0x03; Desactiva el Enable de la LCD

movwf porta; y ponla en modo datos.

movlw 0x0F; Deshabilita la

movwf portb; lectura del teclado.

clrf resp\_w; Borra el contenido del registro resp\_w.

clrf resp\_status; Borra el contenido del registro resp\_status.

clrf res\_pclath; Borra el contenido del registro resp\_pclath.

clrf res\_fsr; Borra el contenido del registro resp\_fsr.

clrf presc\_1; Borra el contenido del registro presc\_1.

clrf presc\_2; Borra el contenido del registro presc\_2.

clrf banderas; Borra el contenido del registro banderas.

clrf cont\_milis; Borra el contenido del registro cont\_milis.

clrf cta\_uniseg; Borra el contenido del registro cta\_uniseg.

clrf cta\_decseg; Borra el contenido del registro cta\_decseg.

clrf cta\_unimin; Borra el contenido del registro cta\_unimin.

clrf cta\_decmin; Borra el contenido del registro cta\_decmin.

clrf cta\_unihor; Borra el contenido del registro cta\_unihor.

clrf cta\_dechor; Borra el contenido del registro cta\_dechor.

bcf portd,Activador; Desactiva el dispensamiento.

return; Regresa de la subrutina.

-----

=====

== Subrutina que convierte de binario a ASCII ==

=====

convbin\_ascii        movf temporal,w; Convierte el valor del

call tabla\_conv; registro temporal a su codigo

movwf temporal; equivalente en ASCII.

goto sal\_subconvascii; Ve a sal\_subconvascii.

tabla\_conv

addwf pcl,f; Suma el contenido de los registros w y pcl.

retlw '0'; Regresa con el valor 0 en ascii en w.

retlw '1'; Regresa con el valor 1 en ascii en w.

```
        retlw '2'; Regresa con el valor 2 en ascii en w.  
        retlw '3'; Regresa con el valor 3 en ascii en w.  
        retlw '4'; Regresa con el valor 4 en ascii en w.  
        retlw '5'; Regresa con el valor 5 en ascii en w.  
        retlw '6'; Regresa con el valor 6 en ascii en w.  
        retlw '7'; Regresa con el valor 7 en ascii en w.  
        retlw '8'; Regresa con el valor 8 en ascii en w.  
        retlw '9'; Regresa con el valor 9 en ascii en w.
```

sal\_subconvasciii      return; Sal de la subrutina.

;

---

=====

== Programa principal =

=====

prog\_prin      call prog\_ini; Llama a la subrutina de Inicio.

                call ini\_lcd; Llama a la subrutina de inicialización de la lcd.

inicio      bcf porta,RS\_LCD; Pon a la lcd en modo comando.

                movlw 0x81; Pon al cursor de la

                movwf portc; lcd en el digito 2.

                call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

                bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

                movlw 'C'; Manda al bus de datos el

                movwf portc; código del caracter C en ASCII.

                call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

                movlw 'A'; Manda al bus de datos el

                movwf portc; código del caracter A en ASCII.

                call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

                movlw 'P'; Manda al bus de datos el

                movwf portc; código del caracter P en ASCII.

                call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

movlw 'S'; Manda al bus de datos el  
movwf portc; código del carácter S en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'U'; Manda al bus de datos el  
movwf portc; código del carácter U en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'L'; Manda al bus de datos el  
movwf portc; código del carácter L en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'E'; Manda al bus de datos el  
movwf portc; código del carácter E en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw ' '; Manda al bus de datos el  
movwf portc; código del carácter en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'A'; Manda al bus de datos el  
movwf portc; código del carácter A en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'U'; Manda al bus de datos el  
movwf portc; código del carácter U en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'T'; Manda al bus de datos el  
movwf portc; código del carácter T en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'O'; Manda al bus de datos el  
movwf portc; código del carácter O en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'M'; Manda al bus de datos el  
movwf portc; código del carácter M en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'A'; Manda al bus de datos el  
movwf portc; código del carácter A en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

movlw 'T'; Manda al bus de datos el  
movwf portc; código del carácter T en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'I'; Manda al bus de datos el  
movwf portc; código del carácter I en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'C'; Manda al bus de datos el  
movwf portc; código del carácter C en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.  
movlw 0xC5; Pon al cursor de la  
movwf portc; lcd en el digito 26.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

movlw 'D'; Manda al bus de datos el  
movwf portc; código del carácter D en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'T'; Manda al bus de datos el  
movwf portc; código del carácter I en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'S'; Manda al bus de datos el  
movwf portc; código del carácter S en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'P'; Manda al bus de datos el  
movwf portc; código del carácter P en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'E'; Manda al bus de datos el  
movwf portc; código del carácter E en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'N'; Manda al bus de datos el  
movwf portc; código del carácter N en ASCII.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

movlw 'S'; Manda al bus de datos el

movwf portc; código del caracter S en ASCII.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

movlw 'E'; Manda al bus de datos el

movwf portc; código del caracter E en ASCII.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

movlw 'R'; Manda al bus de datos el

movwf portc; código del caracter R en ASCII.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.

movlw 0x95; Pon al cursor de la

movwf portc; lcd en el digito 42.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

movlw 'P'; Manda al bus de datos el

movwf portc; código del caracter P en ASCII.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

movlw 'r'; Manda al bus de datos el

movwf portc; código del caracter r en ASCII.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

movlw 'e'; Manda al bus de datos el

movwf portc; código del caracter e en ASCII.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

movlw 's'; Manda al bus de datos el

movwf portc; código del caracter s en ASCII.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

movlw 'i'; Manda al bus de datos el

movwf portc; código del caracter i en ASCII.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

movlw 'o'; Manda al bus de datos el

movwf portc; código del carácter o en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'n'; Manda al bus de datos el  
movwf portc; código del carácter n en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'e'; Manda al bus de datos el  
movwf portc; código del carácter e en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw ' '; Manda al bus de datos el  
movwf portc; código del carácter en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'O'; Manda al bus de datos el  
movwf portc; código del carácter O en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'N'; Manda al bus de datos el  
movwf portc; código del carácter N en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw '/'; Manda al bus de datos el  
movwf portc; código del carácter / en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'C'; Manda al bus de datos el  
movwf portc; código del carácter C en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw ' '; Manda al bus de datos el  
movwf portc; código del carácter en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'p'; Manda al bus de datos el  
movwf portc; código del carácter p en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'a'; Manda al bus de datos el  
movwf portc; código del carácter a en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'r'; Manda al bus de datos el

movwf portc; código del caracter r en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'a'; Manda al bus de datos el  
movwf portc; código del caracter a en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.  
movlw 0xD4; Pon al cursor de la  
movwf portc; lcd en el digito 61.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

movlw 'i'; Manda al bus de datos el  
movwf portc; código del caracter i en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'n'; Manda al bus de datos el  
movwf portc; código del caracter n en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 't'; Manda al bus de datos el  
movwf portc; código del caracter t en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'r'; Manda al bus de datos el  
movwf portc; código del caracter r en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'o'; Manda al bus de datos el  
movwf portc; código del caracter o en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'd'; Manda al bus de datos el  
movwf portc; código del caracter d en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'u'; Manda al bus de datos el  
movwf portc; código del caracter u en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

movlw 'c'; Manda al bus de datos el  
movwf portc; código del carácter c en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'i'; Manda al bus de datos el  
movwf portc; código del carácter i en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'r'; Manda al bus de datos el  
movwf portc; código del carácter r en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw ' '; Manda al bus de datos el  
movwf portc; código del carácter en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'u'; Manda al bus de datos el  
movwf portc; código del carácter u en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'n'; Manda al bus de datos el  
movwf portc; código del carácter n en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw ' '; Manda al bus de datos el  
movwf portc; código del carácter en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 't'; Manda al bus de datos el  
movwf portc; código del carácter t en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'i'; Manda al bus de datos el  
movwf portc; código del carácter i en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'e'; Manda al bus de datos el  
movwf portc; código del carácter e en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'm'; Manda al bus de datos el  
movwf portc; código del carácter m en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

movlw 'p'; Manda al bus de datos el  
movwf portc; código del carácter p en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'o'; Manda al bus de datos el  
movwf portc; código del carácter o en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

barre\_teclado1 bsf portb,Act\_Ren00; Desactiva el renglón 0.  
nop; Sin operación.  
bsf portb,Act\_Ren11; Desactiva el renglón 1.  
nop; Sin operación.  
bsf portb,Act\_Ren22; Desactiva el renglón 2.  
nop; Sin operación.  
bsf portb,Act\_Ren33; Desactiva el renglón 3.  
nop; Sin operación.  
bcf portb,Act\_Ren33; Activa el renglón 3.  
movf portb,w; Lee el teclado  
movwf Var\_teclado; matricial para  
movlw 0XF0; averiguar si una tecla  
andwf Var\_teclado,f; fué oprimida.  
movlw Tec\_ON; Brinca si  
subwf Var\_teclado,w; la tecla ON  
btfs status,Z; fue oprimida.  
goto barre\_teclado1; Ve a barre teclado1.  
call nuevo\_time; Ve a nuevo\_time.

cuenta\_time  
movf cta\_uniseg,w; Mueve el contenido del registro  
movwf temporal; cta\_uniseg al registro temporal.  
call convbin\_ascii; Llama a la subrutina que convierte de binario a ascii.  
movf temporal,w; Mueve el contenido del registro  
movwf buffer1; temporal al registro buffer1.  
movf cta\_decseg,w; Mueve el contenido del registro  
movwf temporal; cta\_decseg al registro temporal.

call convbin\_ascii; Llama a la subrutina que convierte de binario a ascii.  
movf temporal,w; Mueve el contenido del registro  
movwf buffer2; temporal al registro buffer2.  
movf cta\_unimin,w; Mueve el contenido del registro  
movwf temporal; cta\_unimin al registro temporal.  
call convbin\_ascii; Llama a la subrutina que convierte de binario a ascii.  
movf temporal,w; Mueve el contenido del registro  
movwf buffer4; temporal al registro buffer4.  
movf cta\_decmin,w; Mueve el contenido del registro  
movwf temporal; cta\_decmin al registro temporal.  
call convbin\_ascii; Llama a la subrutina que convierte de binario a ascii.  
movf temporal,w; Mueve el contenido del registro  
movwf buffer5; temporal al registro buffer5.  
movf cta\_unihor,w; Mueve el contenido del registro  
movwf temporal; cta\_unihor al registro temporal.  
call convbin\_ascii; Llama a la subrutina que convierte de binario a ascii.  
movf temporal,w; Mueve el contenido del registro  
movwf buffer7; temporal al registro buffer7.  
movf cta\_dechor,w; Mueve el contenido del registro  
movwf temporal; cta\_dechor al registro temporal.  
call convbin\_ascii; Llama a la subrutina que convierte de binario a ascii.  
movf temporal,w; Mueve el contenido del registro  
movwf buffer8; temporal al registro buffer8.  
call muestra\_time; Llama a la subrutina que muestra el tiempo en la lcd.

barre\_teclado2 bsf portb,Act\_Ren00; Desactiva el renglón 0.

nop; Sin operación.  
bsf portb,Act\_Ren11; Desactiva el renglón 1.  
nop; Sin operación.  
bsf portb,Act\_Ren22; Desactiva el renglón 2.  
nop; Sin operación.  
bsf portb,Act\_Ren33; Desactiva el renglón 3.  
nop; Sin operación.

bcf portb,Act\_Ren33; Activa el renglón 3.  
movf portb,w; Lee el teclado  
movwf Var\_teclado; matricial para  
movlw 0XF0; averiguar si una tecla  
andwf Var\_teclado,f; fue oprimida.  
movlw Tec\_igual; Brinca si  
subwf Var\_teclado,w; la tecla igual  
btfsr status,Z; no fue oprimida.  
goto inicio; Ve a inicio.

call barre\_teclado; Llama a la subrutina que barre el teclado.

decf cta\_uniseg,f; Decrementa en 1 el contenido del registro cta\_uniseg.  
movlw .255; Brinca si el valor  
subwf cta\_uniseg,w; del registro cta\_uniseg  
btfsr status,Z; es 255.  
goto cuenta\_time; Ve a cuenta\_time.  
movlw .9; Carga la cuenta de  
movwf cta\_uniseg; 9 al registro cta\_uniseg.

decf cta\_decseg,f; Decrementa en 1 el contenido del registro cta\_decseg.  
movlw .255; Brinca si el valor  
subwf cta\_decseg,w; del registro cta\_decseg  
btfsr status,Z; es 255.  
goto cuenta\_time; Ve a cuenta\_time.  
movlw .5; Carga la cuenta de  
movwf cta\_decseg; 5 al registro cta\_decseg.  
movlw .9; Carga la cuenta de  
movwf cta\_uniseg; 9 al registro cta\_uniseg.

decf cta\_unimin,f; Incrementa en 1 el contenido del registro cta\_unimin.  
movlw .255; Brinca si el valor  
subwf cta\_unimin,w; del registro cta\_unimin

btfs status,Z; es 255.  
goto cuenta\_time; Ve a cuenta\_time.  
movlw .9; Carga la cuenta de  
movwf cta\_unimin; 9 al registro cta\_unimin.  
movlw .5; Carga la cuenta de  
movwf cta\_decseg; 5 al registro cta\_decseg.  
movlw .9; Carga la cuenta de  
movwf cta\_uniseg; 9 al registro cta\_uniseg.

decf cta\_decmin; Incrementa en 1 el contenido del registro cta\_decmin.  
movlw .255; Brinca si el valor  
subwf cta\_decmin,w; del registro cta\_decmin  
btfs status,Z; es 255.  
goto cuenta\_time; Ve a cuenta\_time.  
movlw .5; Carga la cuenta de  
movwf cta\_decmin; 5 al registro cta\_decmin.  
movlw .9; Carga la cuenta de  
movwf cta\_unimin; 9 al registro cta\_unimin.  
movlw .5; Carga la cuenta de  
movwf cta\_decseg; 5 al registro cta\_decseg.  
movlw .9; Carga la cuenta de  
movwf cta\_uniseg; 9 al registro cta\_uniseg.

decf cta\_unihor,f; Incrementa en 1 el contenido del registro cta\_unihor.  
movlw .255; Brinca si el valor  
subwf cta\_unihor,w; del registro cta\_unihor  
btfs status,Z; es 255.  
goto cuenta\_time;  
movlw .9; Carga la cuenta de  
movwf cta\_unihor; 9 al registro cta\_unihor.  
movlw .5; Carga la cuenta de  
movwf cta\_decmin; 5 al registro cta\_decmin.  
movlw .9; Carga la cuenta de

movwf cta\_unimin; 9 al registro cta\_unimin.  
movlw .5; Carga la cuenta de  
movwf cta\_decseg; 5 al registro cta\_decseg.  
movlw .9; Carga la cuenta de  
movwf cta\_uniseg; 9 al registro cta\_uniseg.

decf cta\_dechor,f; Incrementa en 1 el contenido del registro cta\_dechor.  
movlw .255; Brinca si el valor  
subwf cta\_dechor,w; del registro cta\_dechor  
btfsr status,Z; es 255.  
goto cuenta\_time; Ve a cuenta\_time.

bsf portd,Activador; Activa el dispensamiento.  
call retardo\_dispensacion; Llama a la subrutina del retardo para el dispensamiento.  
bcf portd,Activador; Desactiva el dispensamiento.

movf resp\_uniseg,w; Restaura el contenido  
movwf cta\_uniseg; del registro cta\_uniseg.  
movf resp\_decseg,w; Restaura el contenido  
movwf cta\_decseg; del registro cta\_decseg.  
movf resp\_unimin,w; Restaura el contenido  
movwf cta\_unimin; del registro cta\_unimin.  
movf resp\_decmin,w; Restaura el contenido  
movwf cta\_decmin; del registro cta\_decmin.  
movf resp\_unihor,w; Restaura el contenido  
movwf cta\_unihor; del registro cta\_unihor.  
movf resp\_dechor,w; Restaura el contenido  
movwf cta\_dechor; del registro cta\_dechor.

goto cuenta\_time; Regresa de la subrutina que cuenta el tiempo.

-----

=====

==== Subrutina que barre el teclado ==

=====

barre\_teclado bsf portb,Act\_Ren00; Desactiva el renglón 0.

nop; Sin operación.

bsf portb,Act\_Ren11; Desactiva el renglón 1.

nop; Sin operación.

bsf portb,Act\_Ren22; Desactiva el renglón 2.

nop; Sin operación.

bsf portb,Act\_Ren33; Desactiva el renglón 3.

nop; Sin operación.

bcf portb,Act\_Ren33; Activa el renglón 3.

movf portb,w; Lee el teclado

movwf Var\_teclado; matricial para

movlw 0XF0; averiguar si una tecla

andwf Var\_teclado,f; fué oprimida

movlw Tec\_ON; Brinca si

subwf Var\_teclado,w; la tecla ON

btfsc status,Z; no fue oprimida.

goto nuevo\_time; Ve a nuevo\_time.

nop; Sin operacion.

btfss banderas,ban\_int; Prueba el bit ban\_int del registro banderas, brinca si esta en 1.

goto barre\_teclado; Ve a barre\_teclado.

bcf banderas,ban\_int; Pon a 0 el bit ban\_int del registro banderas.

return; Regresa de la subrutina.

nuevo\_time

bcf porta,RS\_LCD; Pon a la lcd en modo comando.

movlw 0x01; Borra todos los

movwf portc; datos de la LCD.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

movlw 0x84; Pon al cursor de la

movwf portc; lcd en el digito 5.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

bsf porta,RS\_LCD; Pon a la lcd en modo datos.

movlw 'T'; Manda al bus de datos el  
movwf portc; código del caracter T en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'i'; Manda al bus de datos el  
movwf portc; código del caracter i en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'e'; Manda al bus de datos el  
movwf portc; código del caracter e en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'm'; Manda al bus de datos el  
movwf portc; código del caracter m en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'p'; Manda al bus de datos el  
movwf portc; código del caracter p en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'o'; Manda al bus de datos el  
movwf portc; código del caracter o en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw ' '; Manda al bus de datos el  
movwf portc; código del caracter en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'N'; Manda al bus de datos el  
movwf portc; código del caracter N en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'u'; Manda al bus de datos el  
movwf portc; código del caracter u en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'e'; Manda al bus de datos el  
movwf portc; código del caracter e en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'v'; Manda al bus de datos el

movwf portc; código del caracter v en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'o'; Manda al bus de datos el  
movwf portc; código del caracter o en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.  
movlw 0x9A; Pon al cursor de la  
movwf portc; lcd en el digito 47.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

movlw '/'; Manda al bus de datos el  
movwf portc; código del caracter / en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw '/'; Manda al bus de datos el  
movwf portc; código del caracter / en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw ':'; Manda al bus de datos el  
movwf portc; código del caracter : en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw '/'; Manda al bus de datos el  
movwf portc; código del caracter / en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw ':'; Manda al bus de datos el  
movwf portc; código del caracter : en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw '/'; Manda al bus de datos el  
movwf portc; código del caracter / en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

movlw '/'; Manda al bus de datos el  
movwf portc; código del carácter / en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

dechor  
bsf portb,Act\_Ren00; Desactiva el renglón 0.  
nop; Sin operación.  
bsf portb,Act\_Ren11; Desactiva el renglón 1.  
nop; Sin operación.  
bsf portb,Act\_Ren22; Desactiva el renglón 2.  
nop; Sin operación.  
bsf portb,Act\_Ren33; Desactiva el renglón 3.  
nop; Sin operación.  
bcf portb,Act\_Ren33; Activa el renglón 3.  
movf portb,w; Lee el teclado  
movwf Var\_teclado; matricial para  
movlw 0xF0; averiguar si una tecla  
andwf Var\_teclado,f; fué oprimida.  
movlw No\_haytecla; Brinca si hay  
subwf Var\_teclado,w; alguna tecla oprimida  
btfsr status,Z; del renglón 3.  
goto dechor\_1; Ve a dechor\_1.  
movlw Tec\_0; Brinca si  
subwf Var\_teclado,w; la tecla 0  
btfsr status,Z; no fue oprimida.  
goto dec\_hor\_0; Ve a dec\_hor\_0.

dechor\_1  
bsf portb,Act\_Ren33; Desactiva el renglón 3.  
nop; Sin operación.  
bcf portb,Act\_Ren22; Activa el renglón 2.  
movf portb,w; Lee el teclado  
movwf Var\_teclado; matricial para  
movlw 0xF0; averiguar si una tecla  
andwf Var\_teclado,f; fué oprimida.

```
        movlw No_haytecla; Brinca si hay
        subwf Var_teclado,w; alguna tecla oprimida
        btfsc status,Z; del renglón 2.
        goto dechor_2; Ve a dechor_2.
        movlw Tec_1; Brinca si
        subwf Var_teclado,w; la tecla 1
        btfsc status,Z; no fue oprimida.
        goto dec_hor_1; Ve a dec_hor_1.
        movlw Tec_2; Brinca si
        subwf Var_teclado,w; la tecla 2
        btfsc status,Z; no fué oprimida.
        goto dec_hor_2; Ve a dec_hor_2.
        movlw Tec_3; Brinca si
        subwf Var_teclado,w; la tecla 3
        btfsc status,Z; no fué oprimida.
        goto dec_hor_3; Ve a dec_hor_3.
```

```
dechor_2      bsf portb,Act_Ren22; Desactiva el renglón 2.
              nop; Sin operación.
              bcf portb,Act_Ren11; Activa el renglón 1.
              movf portb,w; Lee el teclado
              movwf Var_teclado; matricial para
              movlw 0xF0; averiguar si una tecla
              andwf Var_teclado,f; no fué oprimida.
              movlw No_haytecla; Brinca si hay
              subwf Var_teclado,w; alguna tecla oprimida
              btfsc status,Z; del renglón 1.
              goto dechor_3; Ve a dechor_3.
              movlw Tec_4; Brinca si
              subwf Var_teclado,w; la tecla 4
              btfsc status,Z; no fué oprimda.
              goto dec_hor_4; Ve a dec_hor_4.
              movlw Tec_5; Brinca si
```

```
subwf Var_teclado,w; la tecla 5
btfsc status,Z; no fué oprimida.
goto dec_hor_5; Ve a dec_hor_5.
movlw Tec_6; Brinca si
subwf Var_teclado,w; la tecla 6
btfsc status,Z; no fué oprimida.
goto dec_hor_6; Ve a dec_hor_6.

dechor_3      bsf portb,Act_Ren11; Desactiva el renglón 1.
nop; Sin operación.
bcf portb,Act_Ren00; Activa el renglón 0.
movf portb,w; Lee el teclado
movwf Var_teclado; matricial para
movlw 0xF0; averiguar si una tecla
andwf Var_teclado,f; fué oprimida.
movlw No_haytecla; Brinca si hay
subwf Var_teclado,w; alguna tecla oprimida
btfsc status,Z; del renglón 0.
goto dechor; Ve a dechor.
movlw Tec_7; Brinca si
subwf Var_teclado,w; la tecla 7
btfsc status,Z; no fué oprimida.
goto dec_hor_7; Ve a dec_hor_7.
movlw Tec_8; Brinca si
subwf Var_teclado,w; la tecla 8
btfsc status,Z; no fué oprimida.
goto dec_hor_8; Ve a dec_hor_8.
movlw Tec_9; Brinca si
subwf Var_teclado,w; la tecla 9
btfsc status,Z; no fué oprimida.
goto dec_hor_9; Ve a dec_hor_9.
goto dechor; Ve a dechor;
```

dec\_hor\_0        movlw .0; Carga 0 al  
                  movwf cta\_dechor; registro cta\_dechor.

                  bcf porta,RS\_LCD; Pon a la lcd en modo comando.  
                  movlw 0x9A; Pon al cursor de la lcd  
                  movwf portc; en el digito 7 del 3er renglón.  
                  call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
                  bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

                  movlw '0'; Manda el código del carácter 0  
                  movwf portc; en ASCII al puerto C.  
                  call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

                  goto unihor; Ve a unihor.

dec\_hor\_1        movlw .1; Carga 1 al  
                  movwf cta\_dechor; registro cta\_dechor.

                  bcf porta,RS\_LCD; Pon a la lcd en modo comando.  
                  movlw 0x9A; Pon al cursor de la lcd  
                  movwf portc; en el digito 7 del 3er renglón.  
                  call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
                  bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

                  movlw '1'; Manda el código del carácter 1  
                  movwf portc; en ASCII al puerto C.  
                  call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

                  goto unihor; Ve a unihor.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.  
movlw 0x9A; Pon al cursor de la lcd  
movwf portc; en el digito 7 del 3er renglón.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

movlw '2'; Manda el código del caracter 2  
movwf portc; en ASCII al puerto C.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto unihor; Ve a unihor.

dec\_hor\_3 movlw .3; Carga 3 al  
movwf cta\_dechor; registro cta\_dechor.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.  
movlw 0x9A; Pon al cursor de la lcd  
movwf portc; en el digito 7 del 3er renglón.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

movlw '3'; Manda el código del caracter 3  
movwf portc; en ASCII al puerto C.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto unihor; Ve a unihor.

dec\_hor\_4 movlw .4; Carga 4 al  
movwf cta\_dechor; registro cta\_dechor.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.  
movlw 0x9A; Pon al cursor de la lcd  
movwf portc; en el digito 7 del 3er renglón.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

movlw '4'; Manda el código del caracter 4

movwf portc; en ASCII al puerto C.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto unihor; Ve a unihor.

dec\_hor\_5 movlw .5; Carga 5 al

movwf cta\_dechor; registro cta\_dechor.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.

movlw 0x9A; Pon al cursor de la lcd

movwf portc; en el digito 7 del 3er renglón.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

movlw '5'; Manda el código del caracter 5

movwf portc; en ASCII al puerto C.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto unihor; Ve a unihor.

dec\_hor\_6 movlw .6; Carga 6 al

movwf cta\_dechor; registro cta\_dechor.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.

movlw 0x9A; Pon al cursor de la lcd

movwf portc; en el digito 7 del 3er renglón.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

movlw '6'; Manda el código del carácter 6  
movwf portc; en ASCII al puerto C.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto unihor; Ve a unihor.

dec\_hor\_7        movlw .7; Carga 7 al  
                  movwf cta\_dechor; registro cta\_dechor.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.  
movlw 0x9A; Pon al cursor de la lcd  
movwf portc; en el dígito 7 del 3er renglón.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
bsf porta,RS\_LCD; Pon a la lcd en modo operación.

movlw '7'; Manda el código del carácter 7  
movwf portc; en ASCII al puerto C.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto unihor; Ve a unihor.

dec\_hor\_8        movlw .8; Carga 8 al  
                  movwf cta\_dechor; registro cta\_dechor.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.  
movlw 0x9A; Pon al cursor de la lcd  
movwf portc; en el dígito 7 del 3er renglón.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
bsf porta,RS\_LCD; Pon a la lcd en modo operación.

movlw '8'; Manda el código del carácter 8  
movwf portc; en ASCII al puerto C.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto unihor; Ve a unihor.

dec\_hor\_9            movlw .9; Carga 9 al

                      movwf cta\_dechor; registro cta\_dechor.

                      bcf porta,RS\_LCD; Pon a la lcd en modo comando.

                      movlw 0x9A; Pon al cursor de la lcd

                      movwf portc; en el digito 7 del 3er renglón.

                      call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

                      bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

                      movlw '9'; Manda el código del carácter 9

                      movwf portc; en ASCII al puerto C.

                      call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

                      goto unihor; Ve a unihor.

unihor              call esp\_int; Llama a la subrutina que espera la interrupción.

                      call esp\_int; Llama a la subrutina que espera la interrupción.

unihor\_0            bsf portb,Act\_Ren00; Desactiva el renglón 0.

                      nop; Sin operación.

                      bsf portb,Act\_Ren11; Desactiva el renglón 0.

                      nop; Sin operación.

                      bsf portb,Act\_Ren22; Desactiva el renglón 2.

                      nop; Sin operación.

                      bsf portb,Act\_Ren33; Desactiva el renglón 3.

                      nop; Sin operación.

                      bcf portb,Act\_Ren33; Activa el renglón 3.

                      movf portb,w; Lee el teclado

                      movwf Var\_teclado; matricial para

                      movlw 0xF0; averiguar si una tecla

```
andwf Var_teclado,f; fué oprimida.  
movlw No_haytecla; Brinca si hay  
subwf Var_teclado,w; alguna tecla oprimida  
btfsr status,Z; del renglón 3.  
goto unihor_1; Ve a unihor_1.  
movlw Tec_0; Brinca si  
subwf Var_teclado,w; la tecla 0  
btfsr status,Z; no fué oprimida.  
goto uni_hor_0; Ve a uni_hor_0.
```

unihor\_1      bsf portb,Act\_Ren33; Desactiva el renglón 3.

```
nop; Sin operación.  
bcf portb,Act_Ren22; Activa el renglón 2.  
movf portb,w; Lee el teclado  
movwf Var_teclado; matricial para  
movlw 0xF0; averiguar si una tecla  
andwf Var_teclado,f; fué oprimida.  
movlw No_haytecla; Brinca si hay  
subwf Var_teclado,w; alguna tecla oprimida  
btfsr status,Z; del renglón 2.  
goto unihor_2; Ve a dechor_2.  
movlw Tec_1; Brinca si  
subwf Var_teclado,w; la tecla 1.  
btfsr status,Z; no fué oprimida.  
goto uni_hor_1; Ve a uni_hor_1.  
movlw Tec_2; Brinca si  
subwf Var_teclado,w; la tecla 2.  
btfsr status,Z; no fué oprimida.  
goto uni_hor_2; Ve a uni_hor_2.  
movlw Tec_3; Brinca si  
subwf Var_teclado,w; la tecla 3.  
btfsr status,Z; no fué oprimida.  
goto uni_hor_3; Ve a uni_hor_3.
```

unihor\_2            bsf portb,Act\_Ren22; Desactiva el renglón 2.

                      nop; Sin operación.

                      bcf portb,Act\_Ren11; Activa el renglón 1.

                      movf portb,w; Lee el teclado

                      movwf Var\_teclado; matricial para

                      movlw 0xF0; averiguar si una tecla

                      andwf Var\_teclado,f; fué oprimida.

                      movlw No\_haytecla; Brinca si hay

                      subwf Var\_teclado,w; alguna tecla oprimida

                      btfsr status,Z; en el renglón 1.

                      goto unihor\_3; Ve a unihor\_3.

                      movlw Tec\_4; Brinca si

                      subwf Var\_teclado,w; la tecla 4

                      btfsr status,Z; no fué oprimida.

                      goto uni\_hor\_4; Ve a uni\_hor\_4.

                      movlw Tec\_5; Brinca si

                      subwf Var\_teclado,w; la tecla 5

                      btfsr status,Z; no fué oprimida.

                      goto uni\_hor\_5; Ve a uni\_hor\_5.

                      movlw Tec\_6; Brinca si

                      subwf Var\_teclado,w; la tecla 6

                      btfsr status,Z; no fué oprimida.

                      goto uni\_hor\_6; Ve a uni\_hor\_6.

unihor\_3            bsf portb,Act\_Ren11; Desactiva el renglón 1.

                      nop; Sin operación.

                      bcf portb,Act\_Ren00; Activa el renglón 0.

                      movf portb,w; Lee el teclado

                      movwf Var\_teclado; matricial para

                      movlw 0xF0; averiguar si una tecla

                      andwf Var\_teclado,f; fué oprimida.

                      movlw No\_haytecla; Brinca si hay

```
subwf Var_teclado,w; alguna tecla oprimida
btfsc status,Z; en el renglón 0.
goto unihor_0; Ve a unihor_0.
movlw Tec_7; Brinca si
subwf Var_teclado,w; la tecla 7
btfsc status,Z; no fué oprimida.
goto uni_hor_7; Ve a uni_hor_7.
movlw Tec_8; Brinca si
subwf Var_teclado,w; la tecla 9
btfsc status,Z; no fué oprimida.
goto uni_hor_8; Ve a uni_hor_8.
movlw Tec_9; Brinca si
subwf Var_teclado,w; la tecla 9
btfsc status,Z; no fué oprimida.
goto uni_hor_9; Ve a uni_hor_9.
goto unihor_0; Ve a unihor_0.
```

uni\_hor\_0

```
movlw .0; Carga 0 al
movwf cta_unihor; registro cta_unihor.

bcf porta,RS_LCD; Pon a la lcd en modo comando.
movlw 0x9B; Pon al cursor de la lcd
movwf portc; en el digito 8 del 3er renglón.
call pulso_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.
bsf porta,RS_LCD; Pon a la lcd en modo operacion.

movlw '0'; Manda el código del carácter 0
movwf portc; en ASCII al puerto C.
call pulso_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto decmin; Ve a decmin.
```

uni\_hor\_1

```
movlw .1; Carga 1 al
```

movwf cta\_unihor; registro cta\_unihor.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.

movlw 0x9B; Pon al cursor de la lcd

movwf portc; en el digito 8 del 3er renglón.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

movlw '1'; Manda el código del caracter 1

movwf portc; en ASCII al puerto C.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto decmin; Ve a decmin.

uni\_hor\_2

movlw .2; Carga 2 al

movwf cta\_unihor; registro cta\_unihor.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.

movlw 0x9B; Pon al cursor de la lcd

movwf portc; en el digito 8 del 3er renglón.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

movlw '2'; Manda el código del caracter 2

movwf portc; en ASCII al puerto C.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto decmin; Ve a decmin.

uni\_hor\_3

movlw .3; Carga 3 al

movwf cta\_unihor; registro cta\_unihor.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.

movlw 0x9B; Pon al cursor de la lcd  
movwf portc; en el digito 8 del 3er renglón.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

movlw '3'; Manda el código del caracter 3  
movwf portc; en ASCII al puerto C.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto decmin; Ve a decmin.

uni\_hor\_4  
movlw .4; Carga 4 al  
movwf cta\_unihor; registro cta\_unihor.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.  
movlw 0x9B; Pon al cursor de la lcd  
movwf portc; en el digito 8 del 3er renglón.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

movlw '4'; Manda el código del caracter 4  
movwf portc; en ASCII al puerto C.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto decmin; Ve a decmin.

uni\_hor\_5  
movlw .5; Carga 5 al  
movwf cta\_unihor; registro cta\_unihor.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.  
movlw 0x9B; Pon al cursor de la lcd  
movwf portc; en el digito 8 del 3er renglón.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

movlw '5'; Mueve el valor de 0 en ascii al registro w.

movwf portc; en ASCII al puerto C.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto decmin; Ve a decmin.

uni\_hor\_6 movlw .6; Carga 6 al

movwf cta\_unihor; registro cta\_unihor.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.

movlw 0x9B; Pon al cursor de la lcd

movwf portc; en el digito 8 del 3er renglón.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

movlw '6'; Manda el código del caracter 6

movwf portc; en ASCII al puerto C.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto decmin; Ve a decmin.

uni\_hor\_7 movlw .7; Carga 7 al

movwf cta\_unihor; registro cta\_unihor.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.

movlw 0x9B; Pon al cursor de la lcd

movwf portc; en el digito 8 del 3er renglón.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

movlw '7'; Manda el código del caracter 7

movwf portc; en ASCII al puerto C.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto decmin; Ve a decmin.

uni\_hor\_8            movlw .8; Carga 8 al

                      movwf cta\_unihor; registro cta\_unihor.

                      bcf porta,RS\_LCD; Pon a la lcd en modo comando.

                      movlw 0x9B; Pon al cursor de la lcd

                      movwf portc; en el digito 8 del 3er renglón.

                      call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

                      bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

                      movlw '8'; Manda el código del carácter 8

                      movwf portc; en ASCII al puerto C.

                      call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

                      goto decmin; Ve a decmin.

uni\_hor\_9            movlw .9; Carga 9 al

                      movwf cta\_unihor; registro cta\_unihor.

                      bcf porta,RS\_LCD; Pon a la lcd en modo comando.

                      movlw 0x9B; Pon al cursor de la lcd

                      movwf portc; en el digito 8 del 3er renglón.

                      call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

                      bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

                      movlw '9'; Manda el código del carácter 9

                      movwf portc; en ASCII al puerto C.

                      call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto decmin; Ve a decmin.

decmin call esp\_int; Llama a la subrutina que espera la interrupción.  
call esp\_int; Llama a la subrutina que espera la interrupción.

decmin\_0 bsf portb,Act\_Ren00; Desactiva el renglón 0.  
nop; Sin operación.  
bsf portb,Act\_Ren11; Desactiva el renglón 1.  
nop; Sin operación.  
bsf portb,Act\_Ren22; Desactiva el renglón 2.  
nop; Sin operación.  
bsf portb,Act\_Ren33; Desactiva el renglón 3.  
nop; Sin operación.  
bcf portb,Act\_Ren33; Activa el renglón 3.  
movf portb,w; Lee el teclado  
movwf Var\_teclado; matricial para  
movlw 0xF0; averiguar si una tecla  
andwf Var\_teclado,f; fué oprimida  
movlw No\_haytecla; Brinca si hay  
subwf Var\_teclado,w; alguna tecla oprimida  
btfsr status,Z; del renglón 3.  
goto decmin\_1; Ve a decmin\_1.  
movlw Tec\_0; Brinca si  
subwf Var\_teclado,w; la tecla 0  
btfsr status,Z; no fué oprimida.  
goto dec\_min\_0; Ve a dec\_min\_0.

decmin\_1 bsf portb,Act\_Ren33; Desactiva el renglón 3.  
nop; Sin operación.  
bcf portb,Act\_Ren22; Activa el renglón 2.  
movf portb,w; Lee el teclado  
movwf Var\_teclado; matricial para  
movlw 0xF0; averiguar si una tecla

```
andwf Var_teclado,f; fué oprimida.  
movlw No_haytecla; Brinca si hay  
subwf Var_teclado,w; alguna tecla oprimida  
btfsr status,Z; del renglón 2.  
goto decmin_2; Ve a decmin_2.  
movlw Tec_1; Brinca si  
subwf Var_teclado,w; la tecla 1  
btfsr status,Z; no fué oprimida.  
goto dec_min_1; Ve a dec_min_1.  
movlw Tec_2; Brinca si  
subwf Var_teclado,w; la tecla 2  
btfsr status,Z; no fué oprimida.  
goto dec_min_2; Ve a dec_min_2.  
movlw Tec_3; Brinca si  
subwf Var_teclado,w; la tecla 3  
btfsr status,Z; no fué oprimida.  
goto dec_min_3; Ve a dec_min_3.
```

```
decmin_2  
bsf portb,Act_Ren22; Desactiva el renglón 2.  
nop; Sin operación.  
bcf portb,Act_Ren11; Activa el renglón 1.  
movf portb,w; Lee el teclado  
movwf Var_teclado; matricial para  
movlw 0xF0; averiguar si una tecla  
andwf Var_teclado,f; fué oprimida.  
movlw No_haytecla; Brinca si hay  
subwf Var_teclado,w; alguna tecla oprimida  
btfsr status,Z; del renglón 1.  
goto decmin_3; Ve a decmin_3.  
movlw Tec_4; Brinca si  
subwf Var_teclado,w; la tecla 4  
btfsr status,Z; no fué oprimida.  
goto dec_min_4; Ve a dec_min_4.
```

```
        movlw Tec_5; Brinca si
        subwf Var_teclado,w; la tecla 5
        btfsc status,Z; no fué oprimida.
        goto dec_min_5; Ve a dec_min_5.

        movlw Tec_6; Brinca si
        subwf Var_teclado,w; la tecla 6
        btfsc status,Z; no fué oprimida.
        goto dec_min_6; Ve a dec_min_6.

decmin_3      bsf portb,Act_Ren11; Desactiva el renglón 1.
                nop; Sin operación.
                bcf portb,Act_Ren00; Activa el renglón 0.
                movf portb,w; Lee el teclado
                movwf Var_teclado; matricial para
                movlw 0xF0; averiguar si una tecla
                andwf Var_teclado,f; fué oprimida.
                movlw No_haytecla; Brinca si hay
                subwf Var_teclado,w; alguna tecla oprimida
                btfsc status,Z; del renglón 0.
                goto decmin_0; Ve a decmin_0.

                movlw Tec_7; Brinca si
                subwf Var_teclado,w; la tecla 7
                btfsc status,Z; no fué oprimida.
                goto dec_min_7; Ve a dec_min_7.

                movlw Tec_8; Brinca si
                subwf Var_teclado,w; la tecla 8
                btfsc status,Z; no fué oprimida.
                goto dec_min_8; Ve a dec_min_8.

                movlw Tec_9; Brinca si
                subwf Var_teclado,w; la tecla 9
                btfsc status,Z; no fué oprimida.
                goto dec_min_9; Ve a dec_min_9.

                goto decmin_0; Ve a decmin_0.
```



bcf porta,RS\_LCD; Pon a la lcd en modo comando.  
movlw 0x9D; Pon al cursor de la lcd  
movwf portc; en el digito 10 del 3er renglón.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

movlw '2'; Manda el código del caracter 2  
movwf portc; en ASCII al puerto C.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto unimin; Ve a unimin.

dec\_min\_3        movlw .3; Carga 3 al  
                  movwf cta\_decmmin; registro cta\_decmmin.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.  
movlw 0x9D; Pon al cursor de la lcd  
movwf portc; en el digito 10 del 3er renglón.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

movlw '3'; Manda el código del caracter 3  
movwf portc; en ASCII al puerto C.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto unimin; Ve a unimin.

dec\_min\_4        movlw .4; Carga 4 al  
                  movwf cta\_decmmin; registro cta\_decmmin.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.  
movlw 0x9D; Pon al cursor de la lcd

movwf portc; en el digito 10 del 3er renglón.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

movlw '4'; Manda el código del caracter 4  
movwf portc; en ASCII al puerto C.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto unimin; Ve a unimin.

dec\_min\_5        movlw .5; Carga 5 al  
                  movwf cta\_decmi; registro cta\_decmi.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.  
movlw 0x9D; Pon al cursor de la lcd  
movwf portc; en el digito 10 del 3er renglón.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

movlw '5'; Manda el código del caracter 5  
movwf portc; en ASCII al puerto C.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto unimin; Ve a unimin.

dec\_min\_6        movlw .6; Carga 6 al  
                  movwf cta\_decmi; registro cta\_decmi.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.  
movlw 0x9D; Pon al cursor de la lcd  
movwf portc; en el digito 10 del 3er renglón.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

movlw '6'; Manda el código del carácter 6  
movwf portc; en ASCII al puerto C.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto unimin; Ve a unimin.

dec\_min\_7        movlw .7; Carga 7 al  
                  movwf cta\_decmi; registro cta\_decmi.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.  
movlw 0x9D; Pon al cursor de la lcd  
movwf portc; en el dígito 10 del 3er renglón.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
bsf porta,RS\_LCD; Pon a la lcd en modo operación.

movlw '7'; Manda el código del carácter 7  
movwf portc; en ASCII al puerto C.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto unimin; Ve a unimin.

dec\_min\_8        movlw .8; Carga 8 al  
                  movwf cta\_decmi; registro cta\_decmi.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.  
movlw 0x9D; Pon al cursor de la lcd  
movwf portc; en el dígito 10 del 3er renglón.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
bsf porta,RS\_LCD; Pon a la lcd en modo operación.

movlw '8'; Manda el código del carácter 8  
movwf portc; en ASCII al puerto C.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto unimin; Ve a unimin.

dec\_min\_9

movlw .9; Carga 9 al

movwf cta\_decmin; registro cta\_decmin.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.

movlw 0x9D; Pon al cursor de la lcd

movwf portc; en el digito 10 del 3er renglón.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

`movlw '9'; Manda el código del carácter 9`

movwf portc; en ASCII al puerto C.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto unimin; Ve a unimin.

unimin

call esp\_int; Llama a la subrutina que espera a la interrupción.

call esp\_int; Llama a la subrutina que espera a la interrupción.

## unimin\_0

bsf portb,Act\_Ren00; Desactiva el renglón 0.

nop; Sin operación.

bsf portb,Act\_Ren11; Desactiva el renglón 1.

nop; Sin operación.

bsf portb,Act\_Ren22; Desactiva el renglón 2.

nop; Sin operación.

bsf portb,Act\_Ren33; Desactiva el renglón 3.

nop; Sin operación.

bcf portb,Act Ren33; Activa el renglón 3.

movf portb,w; Lee el teclado

movwf Var\_teclado; matricial para

```
        movlw 0xF0; averiguar si una tecla
        andwf Var_teclado,f; fué oprimida.
        movlw No_haytecla; Brinca si hay
        subwf Var_teclado,w; alguna tecla oprimida
        btfsc status,Z; del renglón 3.
        goto unimin_1; Ve a unimin_1.
        movlw Tec_0; Brinca si
        subwf Var_teclado,w; la tecla 0
        btfsc status,Z; no fué oprimida.
        goto uni_min_0; Ve a uni_min_0.
```

unimin\_1           bsf portb,Act\_Ren33; Desactiva el renglón 3.

```
                nop; Sin operación.
                bcf portb,Act_Ren22; Activa el renglón 2.
                movf portb,w; Lee el teclado
                movwf Var_teclado; matricial para
                movlw 0xF0; averiguar si una tecla
                andwf Var_teclado,f; fué oprimida.
                movlw No_haytecla; Brinca si hay
                subwf Var_teclado,w; alguna tecla oprimida
                btfsc status,Z; del renglón 2.
                goto unimin_2; Ve a unimin_2.
                movlw Tec_1; Brinca si
                subwf Var_teclado,w; la tecla 1
                btfsc status,Z; no fué oprimida.
                goto uni_min_1; Ve a uni_min_1.
                movlw Tec_2; Brinca si
                subwf Var_teclado,w; la tecla 2
                btfsc status,Z; no fué oprimida.
                goto uni_min_2; Ve a uni_min_2.
                movlw Tec_3; Brinca si
                subwf Var_teclado,w; la tecla 3.
                btfsc status,Z; no fué oprimida.
```

goto uni\_min\_3; Ve a uni\_min\_3.

unimin\_2      bsf portb,Act\_Ren22; Desactiva el renglón 2.

              nop; Sin operación.

              bcf portb,Act\_Ren11; Activa el renglón 1.

              movf portb,w; Lee el teclado

              movwf Var\_teclado; matricial para

              movlw 0xF0; averiguar si una tecla

              andwf Var\_teclado,f; fué oprimida.

              movlw No\_haytecla; Brinca si hay

              subwf Var\_teclado,w; alguna tecla oprimida

              btfsr status,Z; del renglón 1.

              goto unimin\_3; Ve a unimin\_3.

              movlw Tec\_4; Brinca si

              subwf Var\_teclado,w; la tecla 4

              btfsr status,Z; no fué oprimida.

              goto uni\_min\_4; Ve a uni\_min\_4.

              movlw Tec\_5; Brinca si

              subwf Var\_teclado,w; la tecla 5

              btfsr status,Z; no fué oprimida.

              goto uni\_min\_5; Ve a uni\_min\_5.

              movlw Tec\_6; Brinca si

              subwf Var\_teclado,w; la tecla 6

              btfsr status,Z; no fué oprimida.

              goto uni\_min\_6; Ve a uni\_min\_6.

unimin\_3      bsf portb,Act\_Ren11; Desactiva el renglón 1.

              nop; Sin operación.

              bcf portb,Act\_Ren00; Activa el renglón 0.

              movf portb,w; Lee el teclado

              movwf Var\_teclado; matricial para

              movlw 0xF0; averiguar si una tecla

              andwf Var\_teclado,f; fué oprimida.

```
        movlw No_haytecla; Brinca si hay
        subwf Var_teclado,w; alguna tecla oprimida
        btfsc status,Z; del renglón 0.
        goto unimin_0; Ve a unimin.
        movlw Tec_7; Brinca si
        subwf Var_teclado,w; la tecla 7
        btfsc status,Z; no fué oprimida.
        goto uni_min_7; Ve a uni_min_7.
        movlw Tec_8; Brinca si
        subwf Var_teclado,w; la tecla 8
        btfsc status,Z; no fué oprimida.
        goto uni_min_8; Ve a uni_min_8.
        movlw Tec_9; Brinca si
        subwf Var_teclado,w; la tecla 9
        btfsc status,Z; no fué oprimida.
        goto uni_min_9; Ve a uni_min_9.
        goto unimin_0;
```

```
uni_min_0      movlw .0; Carga 0 al
                movwf cta_unimin; registro cta_unimin.

                bcf porta,RS_LCD; Pon a la lcd en modo comando.
                movlw 0x9E; Pon al cursor de la lcd
                movwf portc; en el digito 11 del 3er renglón.
                call pulso_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.
                bsf porta,RS_LCD; Pon a la lcd en modo operacion.

                movlw '0'; Manda el código del carácter 0
                movwf portc; en ASCII al puerto C.
                call pulso_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

                goto decseg; Ve a decseg.
```

uni\_min\_1

movlw .1; Carga 1 al  
movwf cta\_unimin; registro cta\_unimin.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.  
movlw 0x9E; Pon al cursor de la lcd  
movwf portc; en el digito 11 del 3er renglón.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

movlw '1'; Manda el código del caracter 1  
movwf portc; en ASCII al puerto C.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto decseg; Ve a decseg.

uni\_min\_2

movlw .2; Carga 2 al  
movwf cta\_unimin; registro cta\_unimin.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.  
movlw 0x9E; Pon al cursor de la lcd  
movwf portc; en el digito 11 del 3er renglón.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

movlw '2'; Manda el código del caracter 2  
movwf portc; en ASCII al puerto C.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto decseg; Ve a decseg.

uni\_min\_3 movlw .3; Carga 3 al  
movwf cta\_unimin; registro cta unimin.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.  
movlw 0x9E; Pon al cursor de la lcd  
movwf portc; en el digito 11 del 3er renglón.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

movlw '3'; Manda el código del carácter 3  
movwf portc; en ASCII al puerto C.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto decseg; Ve a decseg.

uni\_min\_4  
movlw .4; Carga 4 al  
movwf cta\_unimin; registro cta\_unimin.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.  
movlw 0x9E; Pon al cursor de la lcd  
movwf portc; en el digito 11 del 3er renglón.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

movlw '4'; Manda el código del carácter 4  
movwf portc; en ASCII al puerto C.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto decseg; Ve a decseg.

uni\_min\_5  
movlw .5; Carga 5 al  
movwf cta\_unimin; registro cta\_unimin.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.  
movlw 0x9E; Pon al cursor de la lcd  
movwf portc; en el digito 11 del 3er renglón.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

movlw '5'; Manda el código del caracter 5

movwf portc; en ASCII al puerto C.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto decseg; Ve a decseg.

uni\_min\_6

movlw .6; Carga 6 al

movwf cta\_unimin; registro cta\_unimin.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.

movlw 0x9E; Pon al cursor de la lcd

movwf portc; en el digito 11 del 3er renglón.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

movlw '6'; Manda el código del caracter 6

movwf portc; en ASCII al puerto C.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto decseg; Ve a decseg.

uni\_min\_7

movlw .7; Carga 7 al

movwf cta\_unimin; registro cta\_unimin.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.

movlw 0x9E; Pon al cursor de la lcd

movwf portc; en el digito 11 del 3er renglón.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

movlw '7'; Manda el código del carácter 7  
movwf portc; en ASCII al puerto C.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto decseg; Ve a decseg.

uni\_min\_8        movlw .8; Carga 8 al  
                  movwf cta\_unimin; registro cta\_unimin.  
  
                  bcf porta,RS\_LCD; Pon a la lcd en modo comando.  
                  movlw 0x9E; Pon al cursor de la lcd  
                  movwf portc; en el dígito 11 del 3er renglón.  
                  call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
                  bsf porta,RS\_LCD; Pon a la lcd en modo operación.

                  movlw '8'; Manda el código del carácter 8  
                  movwf portc; en ASCII al puerto C.  
                  call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

                  goto decseg; Ve a decseg.

uni\_min\_9        movlw .9; Carga 9 al  
                  movwf cta\_unimin; registro cta\_unimin.  
  
                  bcf porta,RS\_LCD; Pon a la lcd en modo comando.  
                  movlw 0x9E; Pon al cursor de la lcd  
                  movwf portc; en el dígito 11 del 3er renglón.  
                  call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
                  bsf porta,RS\_LCD; Pon a la lcd en modo operación.

                  movlw '9'; Manda el código del carácter 9  
                  movwf portc; en ASCII al puerto C.  
                  call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto decseg; Ve a decseg.

decseg call esp\_int; Llama a la subrutina que espera la interrupción.  
call esp\_int; Llama a la subrutina que espera la interrupción.

decseg\_0 bsf portb,Act\_Ren00; Desactiva el renglón 0.  
nop; Sin operación.  
bsf portb,Act\_Ren11; Desactiva el renglón 1.  
nop; Sin operación.  
bsf portb,Act\_Ren22; Desactiva el renglón 2.  
nop; Sin operación.  
bsf portb,Act\_Ren33; Desactiva el renglón 3.  
nop; Sin operación.  
bcf portb,Act\_Ren33; Activa el renglón 3.  
movf portb,w; Lee el teclado  
movwf Var\_teclado; matricial para  
movlw 0xF0; averiguar si una tecla  
andwf Var\_teclado,f; fué oprimida.  
movlw No\_haytecla; Brinca si hay  
subwf Var\_teclado,w; alguna tecla oprimida  
btfsr status,Z; del renglón 3.  
goto decseg\_1; Ve a decseg\_1.  
movlw Tec\_0; Brinca si  
subwf Var\_teclado,w; la teclea 0  
btfsr status,Z; no fué oprimida.  
goto dec\_seg\_0; Ve a dec\_seg\_0.

decseg\_1 bsf portb,Act\_Ren33; Desactiva el renglón 3.  
nop; Sin operación.  
bcf portb,Act\_Ren22; Activa el renglón 2.  
movf portb,w; Lee el teclado  
movwf Var\_teclado; matricial para

```
        movlw 0xF0; averiguar si una tecla
        andwf Var_teclado,f; fué oprimida.
        movlw No_haytecla; Brinca si hay
        subwf Var_teclado,w; alguna tecla oprimida
        btfsc status,Z; del renglón 2.
        goto decseg_2; Ve a decseg_2.
        movlw Tec_1; Brinca si
        subwf Var_teclado,w; la tecla 1
        btfsc status,Z; no fué oprimida.
        goto dec_seg_1; Ve a dec_seg_1.
        movlw Tec_2; Brinca si
        subwf Var_teclado,w; la tecla 2
        btfsc status,Z; no fué oprimida.
        goto dec_seg_2; Ve a dec_seg_2.
        movlw Tec_3; Brinca si
        subwf Var_teclado,w; la tecla 3
        btfsc status,Z; no fué oprimida.
        goto dec_seg_3; Ve a dec_seg_3.
```

```
decseg_2      bsf portb,Act_Ren22; Desactiva el renglón 2.
              nop; Sin operación.
              bcf portb,Act_Ren11; Activa el renglón 1.
              movf portb,w; Lee el teclado
              movwf Var_teclado; matricial para
              movlw 0xF0; averiguar si una tecla
              andwf Var_teclado,f; fué oprimida.
              movlw No_haytecla; Brinca si hay
              subwf Var_teclado,w; alguna tecla oprimida
              btfsc status,Z; del renglón 1.
              goto decseg_3; Ve a decseg_3.
              movlw Tec_4; Brinca si
              subwf Var_teclado,w; la tecla 4
              btfsc status,Z; no fué oprimida.
```

```
        goto dec_seg_4; Ve a dec_seg_4.  
        movlw Tec_5; Brinca si  
        subwf Var_teclado,w; la tecla 5  
        btfsc status,Z; no fué oprimida.  
        goto dec_seg_5; Ve a dec_seg_5.  
        movlw Tec_6; Brinca si  
        subwf Var_teclado,w; la tecla 6  
        btfsc status,Z; no fué oprimida.  
        goto dec_seg_6; Ve a dec_seg_6.  
  
decseg_3      bsf portb,Act_Ren11; Desactiva el renglón 1.  
              nop; Sin operación.  
              bcf portb,Act_Ren00; Activa el renglón 0.  
              movf portb,w; Lee el teclado  
              movwf Var_teclado; matricial para  
              movlw 0xF0; averiguar si una tecla  
              andwf Var_teclado,f; fué oprimida.  
              movlw No_haytecla; Brinca si hay  
              subwf Var_teclado,w; alguna tecla oprimida  
              btfsc status,Z; del renglón 0.  
              goto decseg_0; Ve a decseg.  
              movlw Tec_7; Brinca si  
              subwf Var_teclado,w; la tecla 7  
              btfsc status,Z; no fué oprimida.  
              goto dec_seg_7; Ve a dec_seg_7.  
              movlw Tec_8; Brinca si  
              subwf Var_teclado,w; la tecla 8  
              btfsc status,Z; no fué oprimida.  
              goto dec_seg_8; Ve a dec_seg_8.  
              movlw Tec_9; Brinca si  
              subwf Var_teclado,w; la tecla 9  
              btfsc status,Z; no fué oprimida.  
              goto dec_seg_9; Ve a dec_seg_9.
```

goto decseg\_0; Ve a decseg\_0.

dec\_seg\_0            movlw .0; Carga 0 al

                      movwf cta\_decseg; registro cta\_decseg.

                      bcf porta,RS\_LCD; Pon a la lcd en modo comando.

                      movlw 0xA0; Pon al cursor de la lcd

                      movwf portc; en el digito 13 del 3er renglón.

                      call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

                      bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

                      movlw '0'; Manda el código del caracter 0

                      movwf portc; en ASCII al puerto C.

                      call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

                      goto uniseg; Ve a uniseg.

dec\_seg\_1            movlw .1; Carga 1 al

                      movwf cta\_decseg; registro cta\_decseg.

                      bcf porta,RS\_LCD; Pon a la lcd en modo comando.

                      movlw 0xA0; Pon al cursor de la lcd

                      movwf portc; en el digito 13 del 3er renglón.

                      call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

                      bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

                      movlw '1'; Manda el código del caracter 1

                      movwf portc; en ASCII al puerto C.

                      call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

                      goto uniseg; Ve a uniseg.

dec\_seg\_2            movlw .2; Carga 2 al

movwf cta\_decseg; registro cta\_decseg.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.

movlw 0xA0; Pon al cursor de la lcd

movwf portc; en el digito 13 del 3er renglón.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

movlw '2'; Manda el código del caracter 2

movwf portc; en ASCII al puerto C.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto uniseg; Ve a uniseg.

dec\_seg\_3 movlw .3; Carga 3 al

movwf cta\_decseg; registro cta\_decseg.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.

movlw 0xA0; Pon al cursor de la lcd

movwf portc; en el digito 13 del 3er renglón.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

movlw '3'; Manda el código del caracter 3

movwf portc; en ASCII al puerto C.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto uniseg; Ve a uniseg.

dec\_seg\_4 movlw .4; Carga 4 al

movwf cta\_decseg; registro cta\_decseg.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.

movlw 0xA0; Pon al cursor de la lcd  
movwf portc; en el digito 13 del 3er renglón.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

movlw '4'; Manda el código del caracter 4  
movwf portc; en ASCII al puerto C.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto uniseg; Ve a uniseg.

dec\_seg\_5        movlw .5; Carga 5 al  
                  movwf cta\_decseg; registro cta\_decseg.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.  
movlw 0xA0; Pon al cursor de la lcd  
movwf portc; en el digito 13 del 3er renglón.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

movlw '5'; Manda el código del caracter 5  
movwf portc; en ASCII al puerto C.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto uniseg; Ve a uniseg.

dec\_seg\_6        movlw .6; Carga 6 al  
                  movwf cta\_decseg; registro cta\_decseg.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.  
movlw 0xA0; Pon al cursor de la lcd  
movwf portc; en el digito 13 del 3er renglón.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

movlw '6'; Manda el código del caracter 6

movwf portc; en ASCII al puerto C.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto uniseg; Ve a uniseg.

dec\_seg\_7 movlw .7; Carga 7 al

movwf cta\_decseg; registro cta\_decseg.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.

movlw 0xA0; Pon al cursor de la lcd

movwf portc; en el digito 13 del 3er renglón.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

movlw '7'; Manda el código del caracter 7

movwf portc; en ASCII al puerto C.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto uniseg; Ve a uniseg.

dec\_seg\_8 movlw .8; Carga 8 al

movwf cta\_decseg; registro cta\_decseg.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.

movlw 0xA0; Pon al cursor de la lcd

movwf portc; en el digito 13 del 3er renglón.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

movlw '8'; Manda el código del caracter 8

movwf portc; en ASCII al puerto C.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto uniseg; Ve a uniseg.

dec\_seg\_9                movlw .9; Carga 9 al  
                          movwf cta\_decseg; registro cta\_decseg.

                          bcf porta,RS\_LCD; Pon a la lcd en modo comando.  
                          movlw 0xA0; Pon al cursor de la lcd  
                          movwf portc; en el digito 13 del 3er renglón.  
                          call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
                          bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

                          movlw '9'; Manda el código del carácter 9  
                          movwf portc; en ASCII al puerto C.  
                          call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

                          goto uniseg; Ve a uniseg.

uniseg                  call esp\_int; Llama a la subrutina que espera la interrupción.  
                          call esp\_int; Llama a la subrutina que espera la interrupción.

uniseg\_0                bsf portb,Act\_Ren00; Desactiva el renglón 0.  
                          nop; Sin operación.  
                          bsf portb,Act\_Ren11; Desactiva el renglón 1.  
                          nop; Sin operación.  
                          bsf portb,Act\_Ren22; Desactiva el renglón 2.  
                          nop; Sin operación.  
                          bsf portb,Act\_Ren33; Desactiva el renglón 3.  
                          nop; Sin operación.  
                          bcf portb,Act\_Ren33; Activa el renglón 3.  
                          movf portb,w; Lee el teclado

```
        movwf Var_teclado; matricial para
        movlw 0xF0; averiguar si una tecla
        andwf Var_teclado,f; fué oprimida.
        movlw No_haytecla; Brinca si hay
        subwf Var_teclado,w; alguna tecla oprimida
        btfsc status,Z; del renglón 3.
        goto uniseg_1; Ve a uniseg_1.
        movlw Tec_0; Brinca si
        subwf Var_teclado,w; la tecla 0
        btfsc status,Z; no fué oprimida.
        goto uni_seg_0; Ve a uni_seg_0.
```

uniseg\_1           bsf portb,Act\_Ren33; Desactiva el renglón 3.

```
                nop; Sin operación.
                bcf portb,Act_Ren22; Activa el renglón 2.
                movf portb,w; Lee el teclado
                movwf Var_teclado; matricial para
                movlw 0xF0; averiguar si una tecla
                andwf Var_teclado,f; fué oprimida.
                movlw No_haytecla; Brinca si hay
                subwf Var_teclado,w; alguna tecla oprimida
                btfsc status,Z; del renglón 2.
                goto uniseg_2; Ve a uniseg_2.
        movlw Tec_1; Brinca si
        subwf Var_teclado,w; la tecla 1
        btfsc status,Z; no fué oprimida.
        goto uni_seg_1; Ve a uni_seg_1.
        movlw Tec_2; Brinca si
        subwf Var_teclado,w; la tecla 2
        btfsc status,Z; no fué oprimida.
        goto uni_seg_2; Ve a uni_seg_2.
        movlw Tec_3; Brinca si
        subwf Var_teclado,w; la tecla 3
```

btfsc status,Z; no fué oprimida.

goto uni\_seg\_3; Ve a uni\_seg\_3.

uniseg\_2            bsf portb,Act\_Ren22; Desactiva el renglón 2.

nop; Sin operación.

bcf portb,Act\_Ren11; Activa el renglón 1.

movf portb,w; Lee el teclado

movwf Var\_teclado; matricial para

movlw 0xF0; averiguar si una tecla

andwf Var\_teclado,f; fué oprimida.

movlw No\_haytecla; Brinca si hay

subwf Var\_teclado,w; alguna tecla oprimida

btfsc status,Z; del renglón 1.

goto uniseg\_3; Ve a uniseg\_3.

movlw Tec\_4; Brinca si

subwf Var\_teclado,w; la tecla 4

btfsc status,Z; no fué oprimida.

goto uni\_seg\_4; Ve a uni\_seg\_4.

movlw Tec\_5; Brinca si

subwf Var\_teclado,w; la tecla 5

btfsc status,Z; no fué oprimida.

goto uni\_seg\_5; Ve a uni\_seg\_5.

movlw Tec\_6; Brinca si

subwf Var\_teclado,w; la tecla 6

btfsc status,Z; no fué oprimida.

goto uni\_seg\_6; Ve a uni\_seg\_6.

uniseg\_3            bsf portb,Act\_Ren11; Desactiva el renglón 1.

nop; Sin operación.

bcf portb,Act\_Ren00; Activa el renglón 0.

movf portb,w; Lee el teclado

movwf Var\_teclado; matricial para

movlw 0xF0; averiguar si una tecla

```
andwf Var_teclado,f; fué oprimida.  
movlw No_haytecla; Brinca si hay  
subwf Var_teclado,w; alguna tecla oprimida  
btfsr status,Z; del renglón 0.  
goto uniseg_0; Ve a uniseg.  
movlw Tec_7; Brinca si  
subwf Var_teclado,w; la tecla 7  
btfsr status,Z; no fué oprimida.  
goto uni_seg_7; Ve a uni_seg_7.  
movlw Tec_8; Brinca si  
subwf Var_teclado,w; la tecla 8  
btfsr status,Z; no fué oprimida.  
goto uni_seg_8; Ve a uni_seg_8.  
movlw Tec_9; Brinca si  
subwf Var_teclado,w; la tecla 9  
btfsr status,Z; no fué oprimida.  
goto uni_seg_9; Ve a uni_seg_9.  
goto uniseg_0; Ve a unidseg_0.
```

uni\_seg\_0

```
movlw .0; Carga 0 al  
movwf cta_uniseg; registro cta_uniseg.  
  
bcf porta,RS_LCD; Pon a la lcd en modo comando.  
movlw 0xA1; Pon al cursor de la lcd  
movwf portc; en el digito 14 del 3er renglón.  
call pulso_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
bsf porta,RS_LCD; Pon a la lcd en modo operacion.  
  
movlw '0'; Manda el código del carácter 0  
movwf portc; en ASCII al puerto C.  
call pulso_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
  
goto regresa; Ve a regresa.
```

uni\_seg\_1

movlw .1; Carga 1 al  
movwf cta\_uniseg; registro cta\_uniseg.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.  
movlw 0xA1; Pon al cursor de la lcd  
movwf portc; en el digito 14 del 3er renglón.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

movlw '1'; Manda el código del caracter 1  
movwf portc; en ASCII al puerto C.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto regresa; Ve a regresa.

uni\_seg\_2

movlw .2; Carga 2 al  
movwf cta\_uniseg; registro cta\_uniseg.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.  
movlw 0xA1; Pon al cursor de la lcd  
movwf portc; en el digito 14 del 3er renglón.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

movlw '2'; Manda el código del caracter 2  
movwf portc; en ASCII al puerto C.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto regresa; Ve a regresa.

uni\_seg\_3 movlw .3; Carga 3 al  
movwf cta\_uniseg; registro cta\_uniseg.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.  
movlw 0xA1; Pon al cursor de la lcd  
movwf portc; en el digito 14 del 3er renglón.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

movlw '3'; Manda el código del carácter 3  
movwf portc; en ASCII al puerto C.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto regresa; Ve a regresa.

uni\_seg\_4        movlw .4; Carga 4 al  
                  movwf cta\_uniseg; registro cta\_uniseg.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.  
movlw 0xA1; Pon al cursor de la lcd  
movwf portc; en el digito 14 del 3er renglón.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

movlw '4'; Manda el código del carácter 4  
movwf portc; en ASCII al puerto C.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto regresa; Ve a regresa.

uni\_seg\_5        movlw .5; Carga 5 al  
                  movwf cta\_uniseg; registro cta\_uniseg.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.  
movlw 0xA1; Pon al cursor de la lcd

movwf portc; en el digito 14 del 3er renglón.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

movlw '5'; Manda el código del carácter 5  
movwf portc; en ASCII al puerto C.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto regresa; Ve a regresa.

uni\_seg\_6        movlw .6; Carga 6 al  
                  movwf cta\_uniseg; registro cta\_uniseg.  
  
                  bcf porta,RS\_LCD; Pon a la lcd en modo comando.  
                  movlw 0xA1; Pon al cursor de la lcd  
                  movwf portc; en el digito 14 del 3er renglón.  
                  call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
                  bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

                  movlw '6'; Manda el código del carácter 6  
                  movwf portc; en ASCII al puerto C.  
                  call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
  
                  goto regresa; Ve a regresa.

uni\_seg\_7        movlw .7; Carga 7 al  
                  movwf cta\_uniseg; registro cta\_uniseg.  
  
                  bcf porta,RS\_LCD; Pon a la lcd en modo comando.  
                  movlw 0xA1; Pon al cursor de la lcd  
                  movwf portc; en el digito 14 del 3er renglón.  
                  call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
                  bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

movlw '7'; Manda el código del carácter 7  
movwf portc; en ASCII al puerto C.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto regresa; Ve a regresa.

uni\_seg\_8        movlw .8; Carga 8 al  
                  movwf cta\_uniseg; registro cta\_uniseg.  
  
                  bcf porta,RS\_LCD; Pon a la lcd en modo comando.  
                  movlw 0xA1; Pon al cursor de la lcd  
                  movwf portc; en el dígito 14 del 3er renglón.  
                  call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
                  bsf porta,RS\_LCD; Pon a la lcd en modo operación.

                  movlw '8'; Manda el código del carácter 8  
                  movwf portc; en ASCII al puerto C.  
                  call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

                  goto regresa; Ve a regresa.

uni\_seg\_9        movlw .9; Carga 9 al  
                  movwf cta\_uniseg; registro cta\_uniseg.  
  
                  bcf porta,RS\_LCD; Pon a la lcd en modo comando.  
                  movlw 0xA1; Pon al cursor de la lcd  
                  movwf portc; en el dígito 14 del 3er renglón.  
                  call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
                  bsf porta,RS\_LCD; Pon a la lcd en modo operación.

                  movlw '9'; Manda el código del carácter 9  
                  movwf portc; en ASCII al puerto C.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

goto regresa; Ve a regresa.

regresa

movf cta\_uniseg,w; Respalda el contenido

movwf resp\_uniseg; del registro cta\_uniseg.

movf cta\_decseg,w; Respalda el contenido

movwf resp\_decseg; del registro cta\_decseg.

movf cta\_unimin,w; Respalda el contenido

movwf resp\_unimin; del registro cta\_unimin.

movf cta\_decmim,w; Respalda el contenido

movwf resp\_decmim; del registro cta\_decmim.

movf cta\_unihor,w; Respalda el contenido

movwf resp\_unihor; del registro cta\_unihor.

movf cta\_dechor,w; Respalda el contenido

movwf resp\_dechor; del registro cta\_dechor.

bcf porta,RS\_LCD; Pon a la lcd en modo comando.

movlw 0x01; Borra todos los

movwf portc; datos de la LCD.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

movlw 0x82; Pon al cursor de la

movwf portc; lcd en el digito 3.

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

movlw 'T'; Manda al bus de datos el

movwf portc; código del caracter T en ASCII

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

movlw 'i'; Manda al bus de datos el

movwf portc; código del caracter i en ASCII

call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.

movlw 'e'; Manda al bus de datos el

movwf portc; código del carácter e en ASCII  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'm'; Manda al bus de datos el  
movwf portc; código del carácter m en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'p'; Manda al bus de datos el  
movwf portc; código del carácter o en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'o'; Manda al bus de datos el  
movwf portc; código del carácter o en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw ' '; Manda al bus de datos el  
movwf portc; código del carácter en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'e'; Manda al bus de datos el  
movwf portc; código del carácter e en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'n'; Manda al bus de datos el  
movwf portc; código del carácter n en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw ' '; Manda al bus de datos el  
movwf portc; código del carácter en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'C'; Manda al bus de datos el  
movwf portc; código del carácter C en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'u'; Manda al bus de datos el  
movwf portc; código del carácter u en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 'r'; Manda al bus de datos el  
movwf portc; código del carácter r en ASCII.  
call pulso\_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw 's'; Manda al bus de datos el

```
    movwf portc; código del caracter s en ASCII.  
    call pulso_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
    movlw 'o'; Manda al bus de datos el  
    movwf portc; código del caracter o en ASCII.  
    call pulso_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.
```

return; Regresa de la subrutina.

```
;
```

```
=====  
;== Subrutina que muestra el tiempo en la lcd ==  
=====
```

```
muestra_time bcf porta,RS_LCD; Pon a la lcd en modo comando.  
    movlw 0x9A; Mueve el valor 84 en hexadecimal al registro w.  
    movwf portc; Pon al cursor de la lcd en el digito 5.  
    call pulso_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
    bsf porta,RS_LCD; Pon a la lcd en modo operacion.
```

```
    movf buffer8,w; Manda al bus de datos el  
    movwf portc; contenido del registro buffer 8.  
    call pulso_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
    movf buffer7,w; Manda al bus de datos el  
    movwf portc; contenido del registro buffer 7.  
    call pulso_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
    movlw ':'; Manda al bus de datos el  
    movwf portc; código en ASCII de :.  
    call pulso_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
    movf buffer5,w; Manda al bus de datos el  
    movwf portc; contenido del registro buffer 5.  
    call pulso_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
    movf buffer4,w; Manda al bus de datos el  
    movwf portc; contenido del registro buffer 4.
```

```
call pulso_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movlw ':'; Manda al bus de datos el  
movwf portc; código en ASCII de :.  
call pulso_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movf buffer2,w; Manda al bus de datos el  
movwf portc; contenido del registro buffer 2.  
call pulso_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
movf buffer1,w; Manda al bus de datos el  
movwf portc; contenido del registro buffer 1.  
call pulso_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
  
return; Regresa de la subrutina.
```

```
;
```

```
=====  
== Subrutina de inicialización de la LCD ==  
=====
```

```
ini_lcd      bcf porta,RS_LCD; Pon a la lcd en modo comando.
```

```
          movlw 0x38; Enciende la pantalla  
          movwf portc; y el cursor.  
          call pulso_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
          movlw 0x0c; Establece las operaciones de 8 bits, y selecciona un  
          movwf portc; display de 2 líneas y fuente de 5 x 7 caracteres de puntos.  
          call pulso_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
          movlw 0x01; Borra los caracteres que aparecen  
          movwf portc; en la LCD y pon el cursor en el dígito 1.  
          call pulso_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
          movlw 0x06; Establece el modo para incrementar la dirección en uno y para mover  
          movwf portc; el cursor a la derecha a la hora de escribir en el 00/CG RAM.  
          call pulso_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.  
          movlw 0x80; Coloca el cursor de la LCD  
          movwf portc; en la posición del dígito 1.  
          call pulso_enable; Llama a la subrutina que ingresa el dato/comando a la lcd.
```

bsf porta,RS\_LCD; Pon a la lcd en modo operacion.

return; Regresa de la subrutina.

;  
=====  
== Subrutina que ingresa el comando/dato a la LCD ==  
=====

pulso\_enable bcf porta,Enable\_LCD; Pon a 1 el bit Enable\_LCD del registro porta.

call retardo\_1ms; Llama a la subrutina de retardo de 1ms.

bsf porta,Enable\_LCD; Pon a 0 el bit Enable\_LCD del registro porta.

call ret\_40ms; Llama a la subrutina de retardo de 40ms.

return; Regresa de la subrutina.

;  
=====  
== Subrutina de retardo de 1ms =  
=====

retardo\_1ms clrf cont\_milis; Borra el contenido del registro cont\_milis.

esp\_int1ms movlw .1; Mueve el valor 1 en decimal al registro w.

subwf cont\_milis,w; Resta el contenido del registro w al contenido del registro cont\_milis,  
y guarda el resultado en w.

btfss status,Z; Prueba el bit z del registro status, brinca si esta en 1.

goto esp\_int1ms; Ve a esp\_int1ms.

return; Regresa de la interrupcion.

;  
=====  
== Subrutina de retardo de 40ms =  
=====

ret\_40ms clrf cont\_milis; Borra el contenido del registro cont\_milis.

```
esp_int40ms      movlw .40; Mueve el valor 40 en decimal al registro w.  
                  subwf cont_milis,w; Resta el contenido del registro w al contenido del registro cont_milis,  
y guarda el resultado en w.  
                  btfss status,Z; Prueba el bit z del registro status, brinca si esta en 1.  
                  goto esp_int40ms; Ve a esp_int40ms.  
  
                  return; Regresa de la subrutina.  
;  
=====  
== Subrutina de espera de int. de 1 segundo ==  
=====  
  
esp_int          nop; Sin operacion.  
                  btfss banderas,ban_int; Prueba el bit ban_int del registro banderas, brinca si esta en 1.  
                  goto esp_int; Ve a esp_int.  
                  bcf banderas,ban_int; Pon a 0 el bit ban_int del registro banderas.  
  
                  return; Regresa de la interrupcion.  
;  
=====  
== Subrutina de retardo de dispensamiento ==  
=====  
  
retardo_dispen  clrf cont_segundos; Borra el contenido del registro cont_milis.  
esp_ret          movlw .3; Mueve el valor 3 en decimal al registro w.  
                  subwf cont_segundos,w; Resta el contenido del registro w al contenido del registro  
cont_milis, y guarda el resultado en w.  
                  btfss status,Z; Prueba el bit z del registro status, brinca si esta en 1.  
                  goto esp_ret; Ve a esp_ret.  
  
                  return; Regresa de la interrupcion.  
;  
end; Termina el programa
```

```

#include <16f877a.h>
#fuses hs, nowdt
# use delay (clock=4M)
/*
#define lcd_rs_pin    pin_b0 // esto es en caso de que yo quiero especificar los pines que yo quiero
utilizar
#define lcd_rw_pin    pin_b1
#define lcd_enable_pin pin_b2
#define lcd_data4     pin_b4
#define lcd_data5     pin_b5
#define lcd_data6     pin_b6
#define lcd_data7     pin_b7
*/
#include <lcd.c>
void main (){
    set_tris_d(0); // esto es si en caso que yo quiero poner solo pines D COMO salida
    lcd_init();
    while (true){
        lcd_gotoxy (1,1);           //Uvicamos la palabra en la primera fila y la primera columna
        printf(lcd_putc, "AutomaticCapsule"); // manifestamos la palabra en nuestro lcd
        lcd_gotoxy (1,2);           //Uvicamos la palabra en la segunda fila y la primera columna
        printf(lcd_putc, "Dispenser");
        delay_ms(8000);            // Le damos un retardo de 500 ms
        lcd_putc("\f");             // borramos la palabra para entrar a otra instruccion
        for (int car = 0;car<=16;car++){
            lcd_gotoxy(car,1);
            printf(lcd_putc, "PANDILLA");
            delay_ms(500);
            lcd_putc("\f");
        }
        for (car=16;car>=1;Car--){
            lcd_gotoxy(car,1);
            printf(lcd_putc, "PANDILLA");
            delay_ms(500);
            lcd_putc("\f");
        }
    }
}
}
}

```

```
#include <16f877a.h>
#fuses hs, nowdt
# use delay (clock=4M)
# include <lcd.c>
int A=1;
void main()
{
    lcd_init();
    set_tris_A(0x00);
    set_tris_B(0xFF);
    set_tris_C(0x00);
    output_A(0x00);
    output_C(0x00);
    set_tris_d(0);
    lcd_gotoxy (1,1);
    printf(lcd_putc, "AutomaticCapsule");
    lcd_gotoxy (1,2);
    printf(lcd_putc, "Dispenser");
    delay_ms(4000);
    printf(lcd_putc,"f");
    do{
        if(!input(pin_B0)==1)
        {
            A--;
            delay_ms(10);
        }
        if(!input(pin_B1)==1)
        {
            A++;
            delay_ms(10);
        }

        if(A>3 && A==0)
        A=1;
        if(A==1)
        {
            printf(lcd_putc,"f");
            lcd_gotoxy (1,1);
            printf(lcd_putc,"Cual activar");
            lcd_gotoxy (1,2);
            printf(lcd_putc,"1.A*");
            lcd_gotoxy (6,2);
            printf(lcd_putc,"2.B");
            lcd_gotoxy (11,2);
            printf(lcd_putc,"3.C");
        }
    }
}
```

```
delay_ms(500);

if(!input(pin_B2)==1){
output_high(pin_A0);
}
if(!input(pin_B3)==1){
output_low(pin_A0);
}

}

if(A==2)
{
printf(lcd_putc,"f");
lcd_gotoxy (1,1);
printf(lcd_putc,"Cual activar");
lcd_gotoxy (1,2);
printf(lcd_putc,"1.A");
lcd_gotoxy (6,2);
printf(lcd_putc,"2.B*");
lcd_gotoxy (11,2);
printf(lcd_putc,"3.C");
delay_ms(500);
if(!input(pin_B2)==1){
output_high(pin_A1);
}
if(!input(pin_B3)==1){
output_low(pin_A1);
}
}

if(A==3){
printf(lcd_putc,"f");
lcd_gotoxy (1,1);
printf(lcd_putc,"Cual activar");
lcd_gotoxy (1,2);
printf(lcd_putc,"1.A");
lcd_gotoxy (6,2);
printf(lcd_putc,"2.B");
lcd_gotoxy (11,2);
printf(lcd_putc,"3.C*");
delay_ms(500);

if(!input(pin_B2)==1){
output_high(pin_A2);
}
if(!input(pin_B2)==1){
```

```
    output_low(pin_A2);
}
}
do{
    output_high(pin_A3);
}while(!input(pin_B4)==1);

do{
    output_high(pin_A4);
}while (!input(pin_B5)==1);

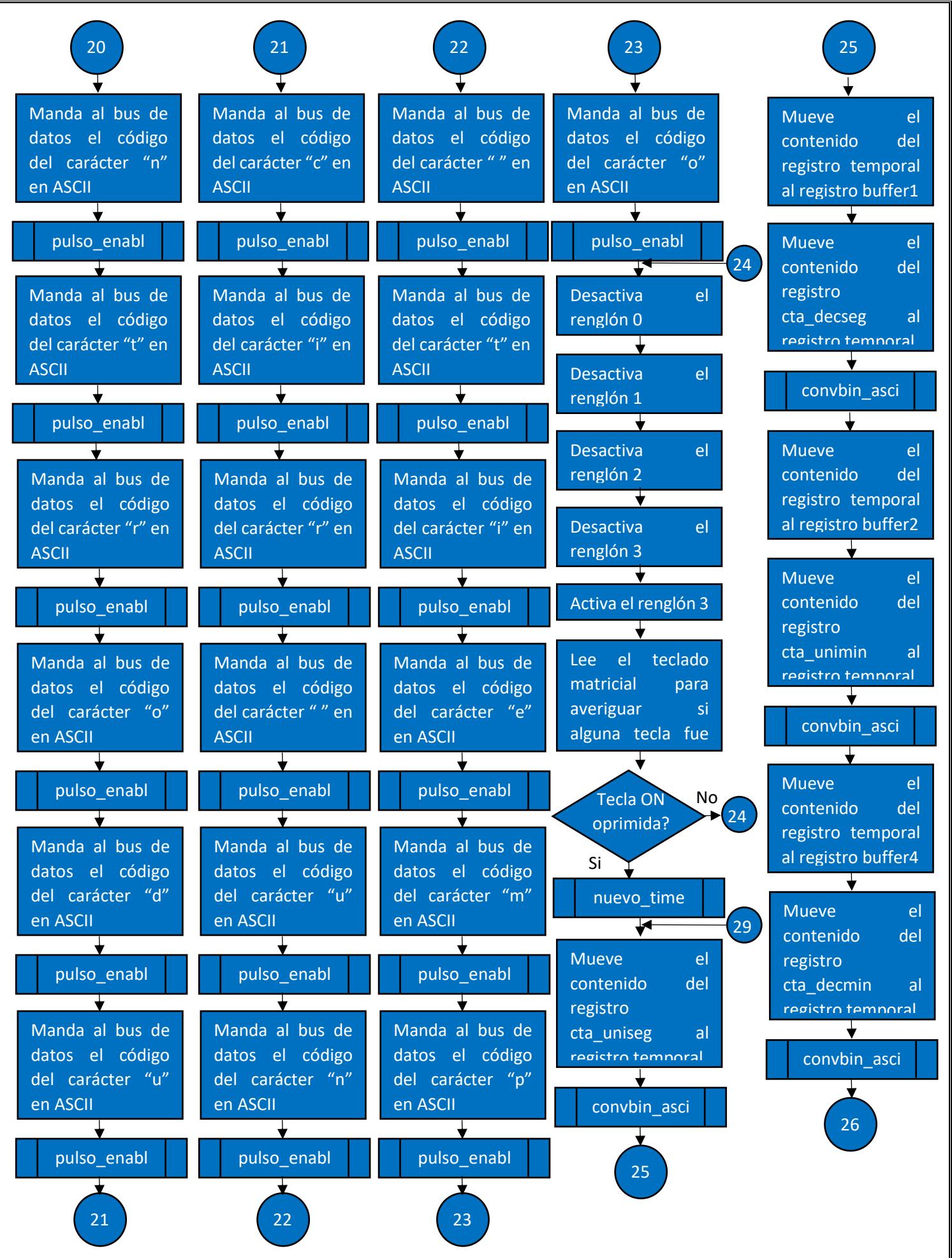
do{
    output_high(pin_A5);
}while(!input(pin_B6)==1);

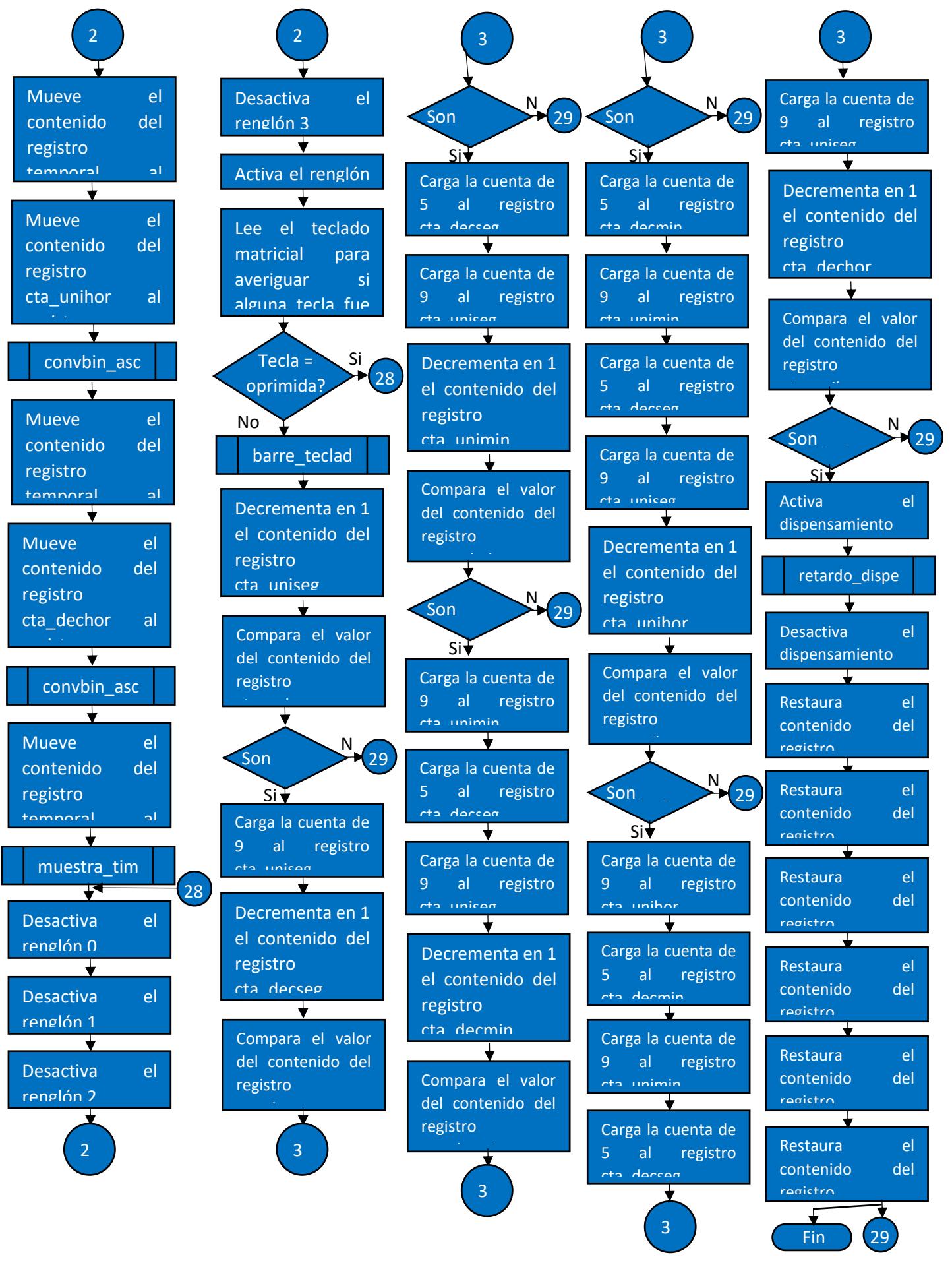
}while(true);
}
```

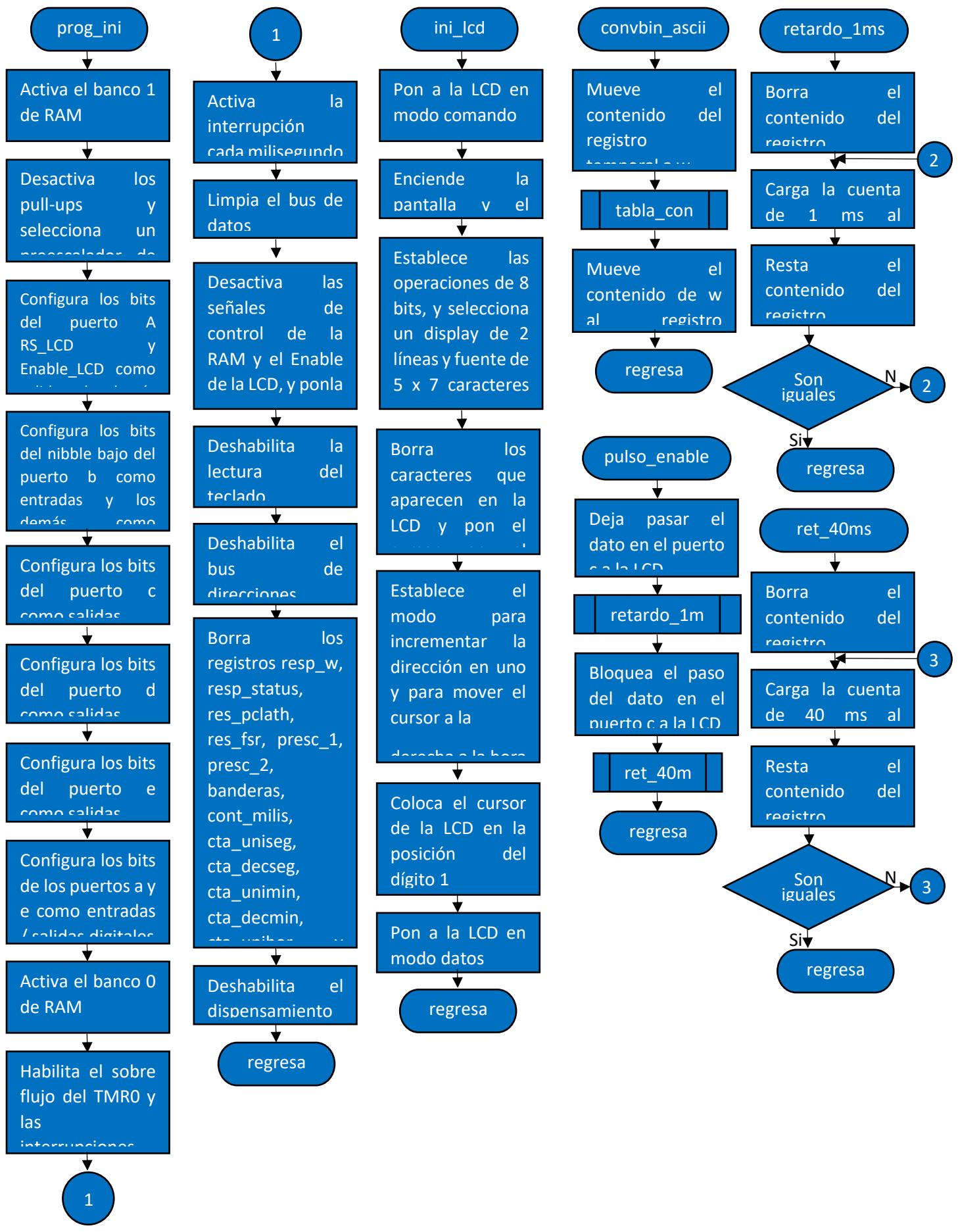
## Diagrama de Flujo











tabla\_conv

Suma el contenido del registro w con el contenido del registro pcl y guarda el resultado en el registro pcl

$pcl_0 = pcl_0 + 4$

Regresa con  
el valor 4 en  
ASCII en w

$pcl_0 = pcl_0 + 3$

Regresa con  
el valor 3 en  
ASCII en w

$pcl_0 = pcl_0 + 2$

Regresa con  
el valor 2 en  
ASCII en w

$pcl_0 = pcl_0 + 1$

Regresa con  
el valor 1 en  
ASCII en w

$pcl_0 = pcl_0 + 0$

Regresa con  
el valor 0 en  
ASCII en w

$pcl_0 = pcl_0 + 5$

Regresa con  
el valor 5 en  
ASCII en w

$pcl_0 = pcl_0 + 6$

Regresa con  
el valor 6 en  
ASCII en w

$pcl_0 = pcl_0 + 7$

Regresa con  
el valor 7 en  
ASCII en w

$pcl_0 = pcl_0 + 8$

Regresa con  
el valor 8 en  
ASCII en w

$pcl_0 = pcl_0 + 9$

Regresa con  
el valor 9 en  
ASCII en w

esp\_int

4

Prueba el bit  
ban\_int del  
registro banderas

Bit en  
1?

No

4

Si

Pon a 0 el bit  
ban\_int del  
registro banderas

regresa

retardo\_dipen

5

Borra el contenido  
del registro  
cont\_segundos

Carga la cuenta de  
3 seg al registro w

Resta el contenido  
del registro  
cont\_segundos de  
w

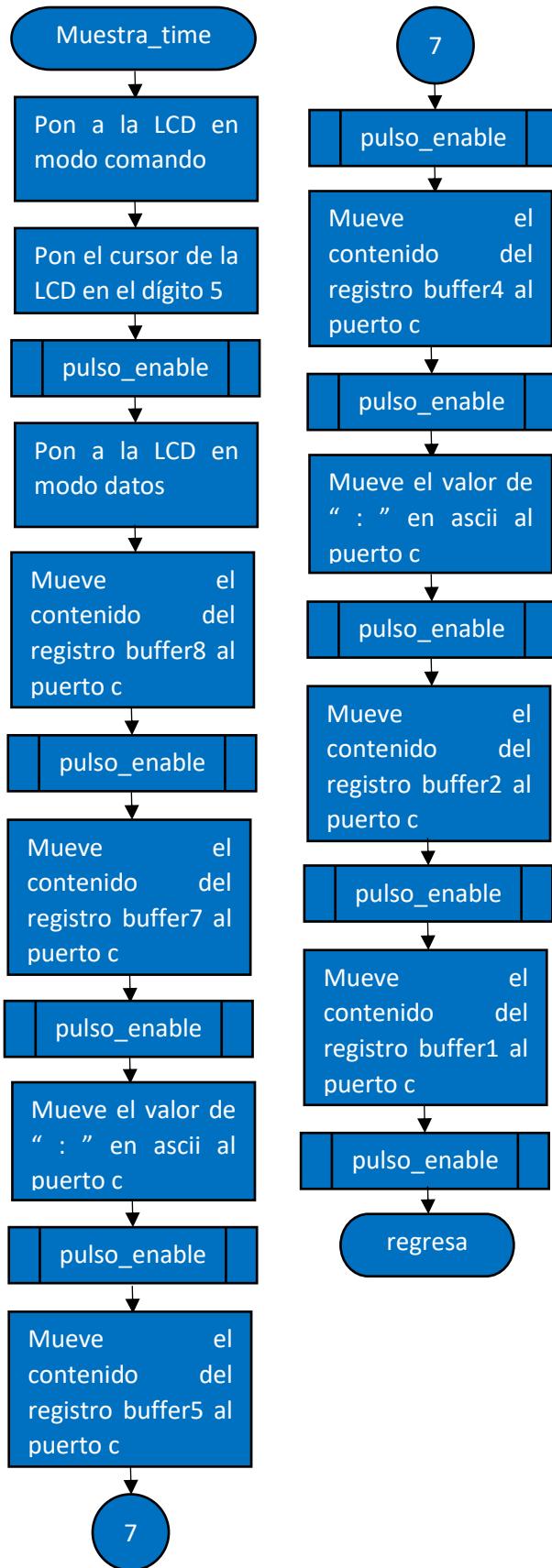
Son iguales?

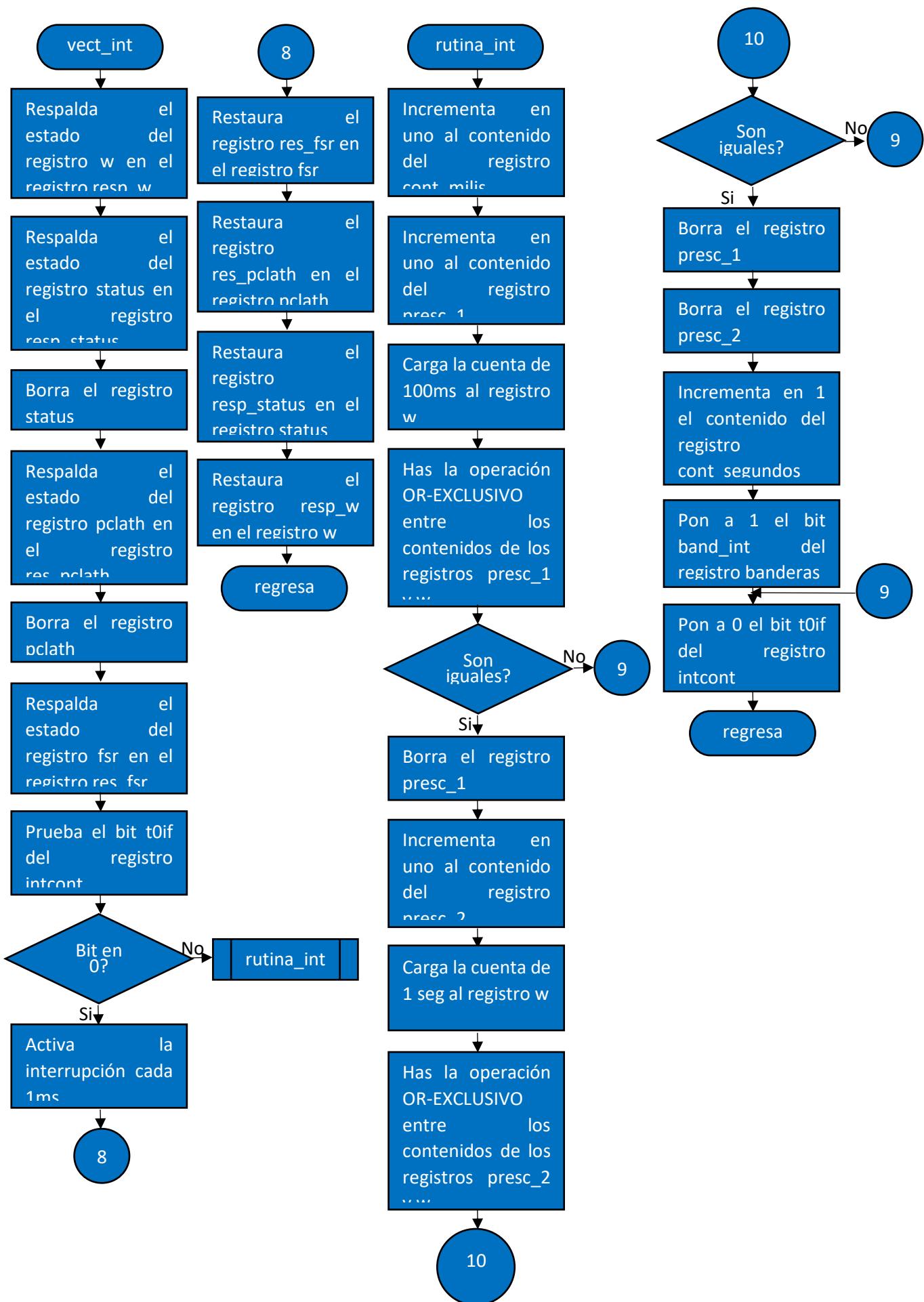
No

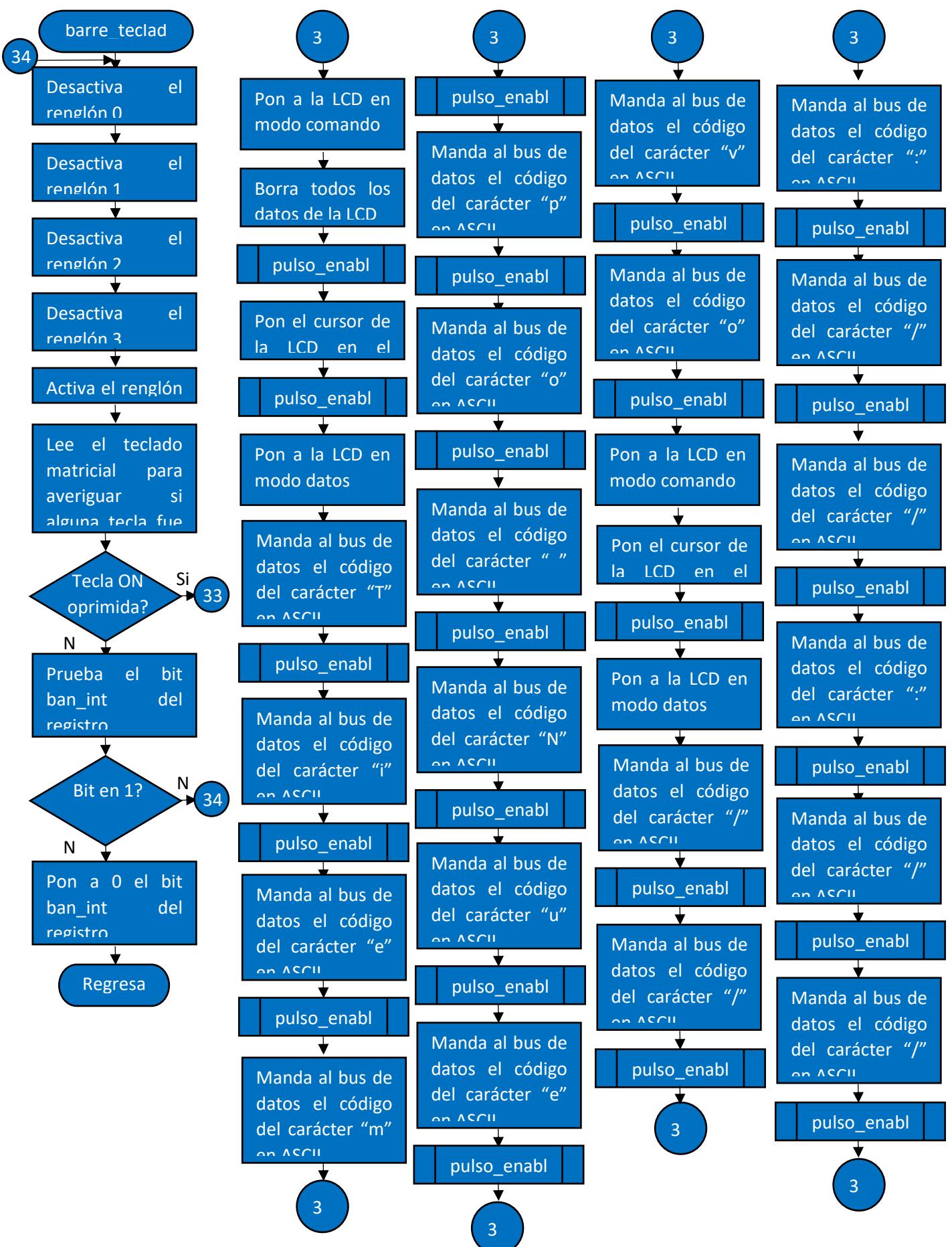
5

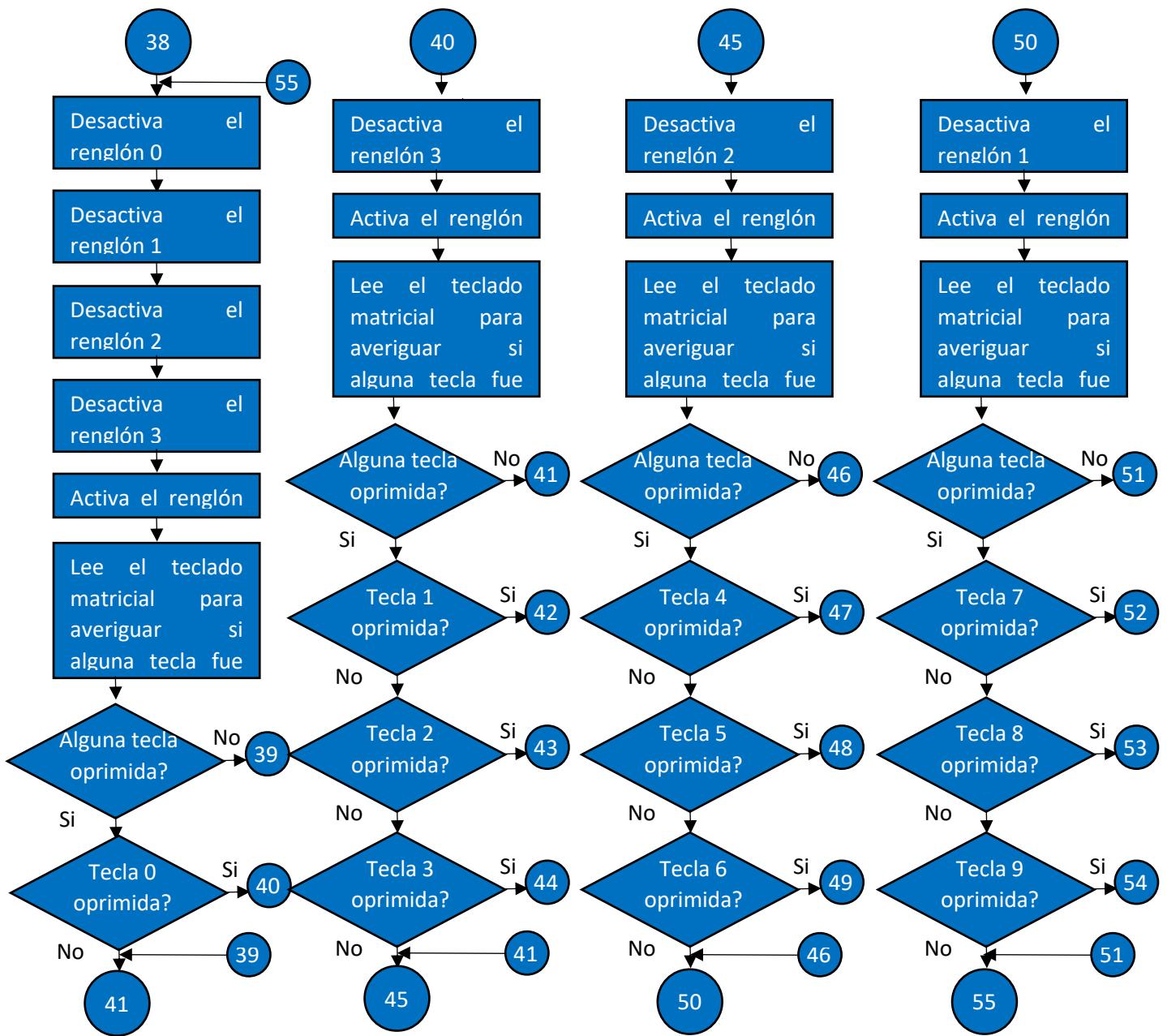
Si

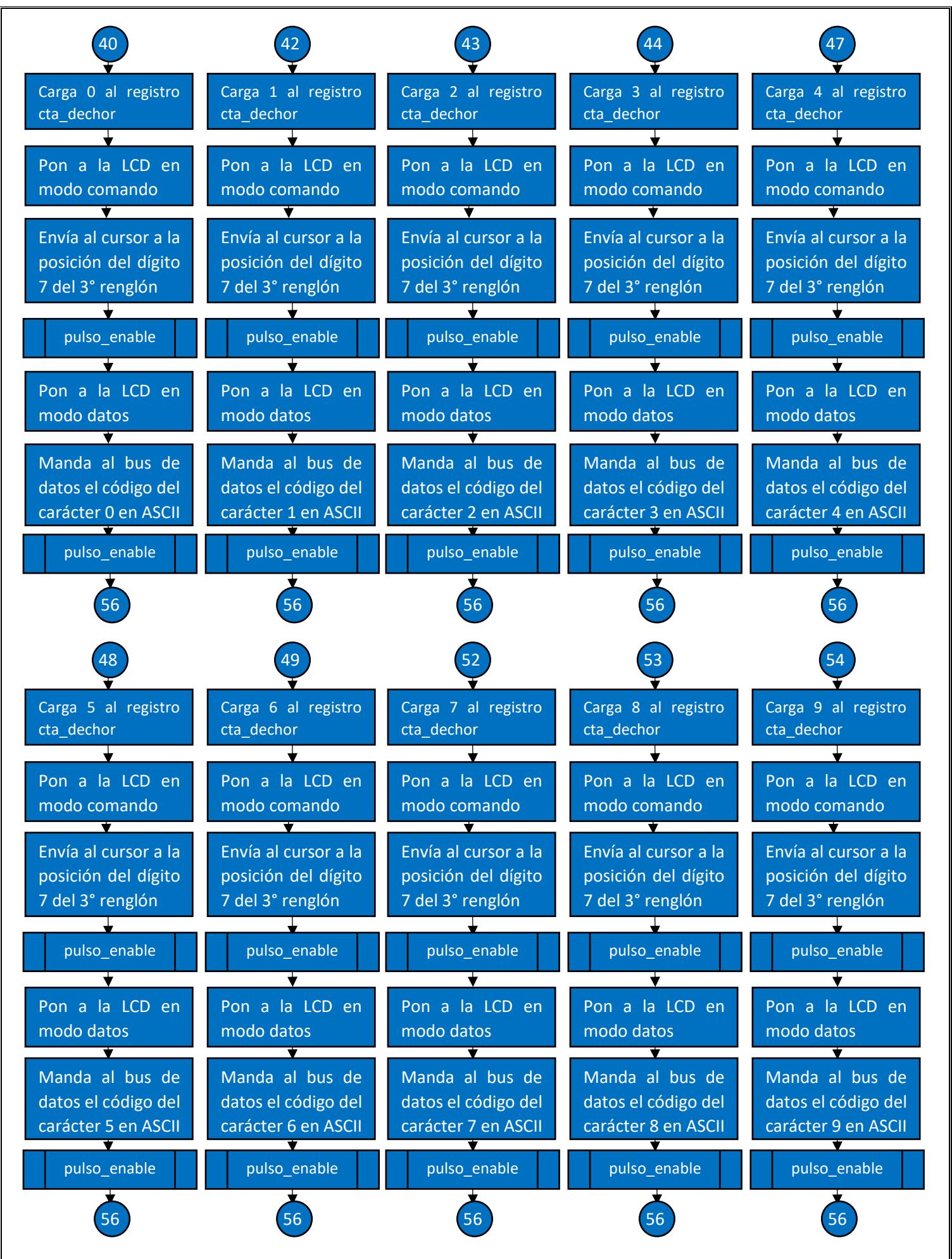
regresa

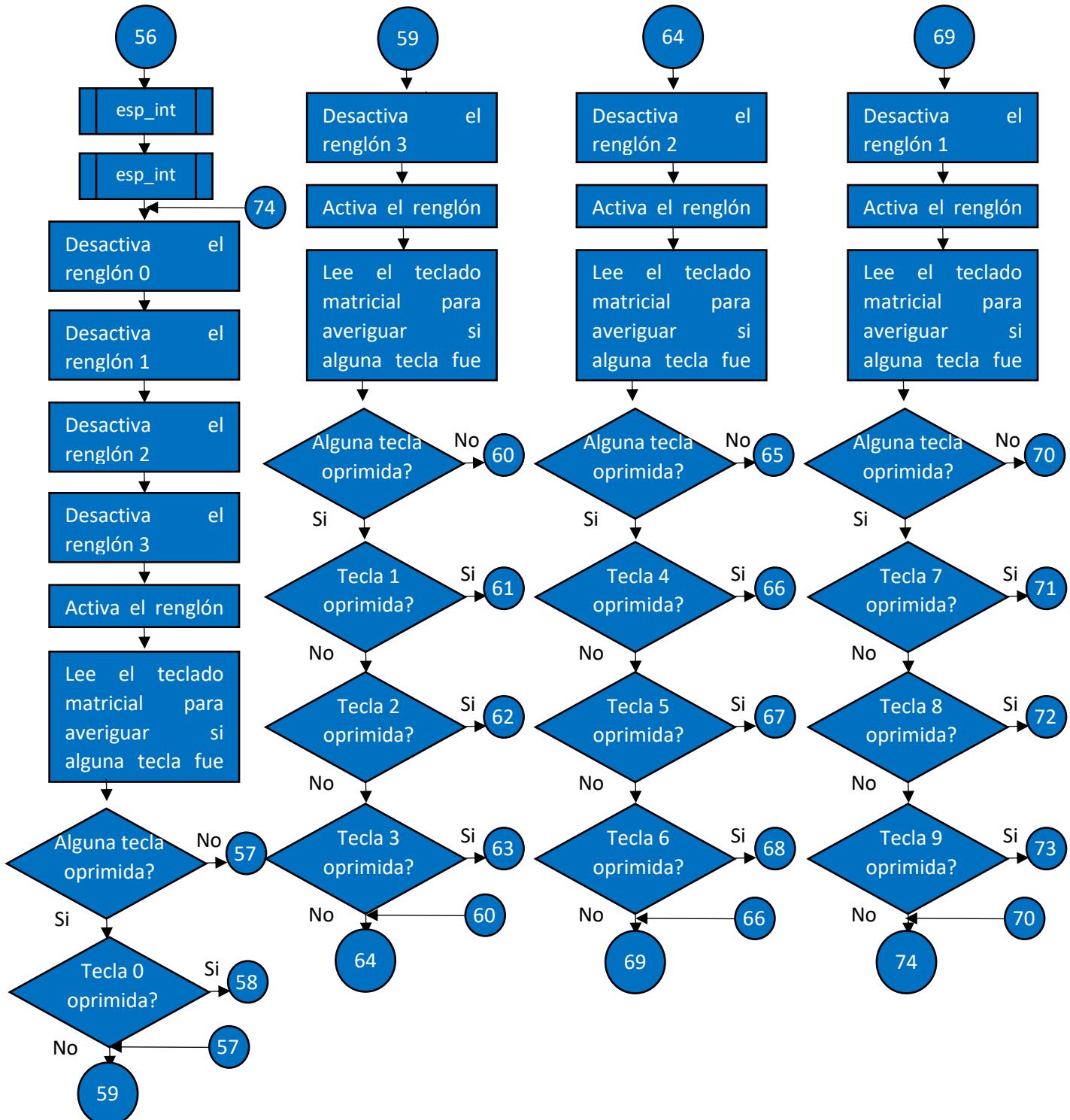


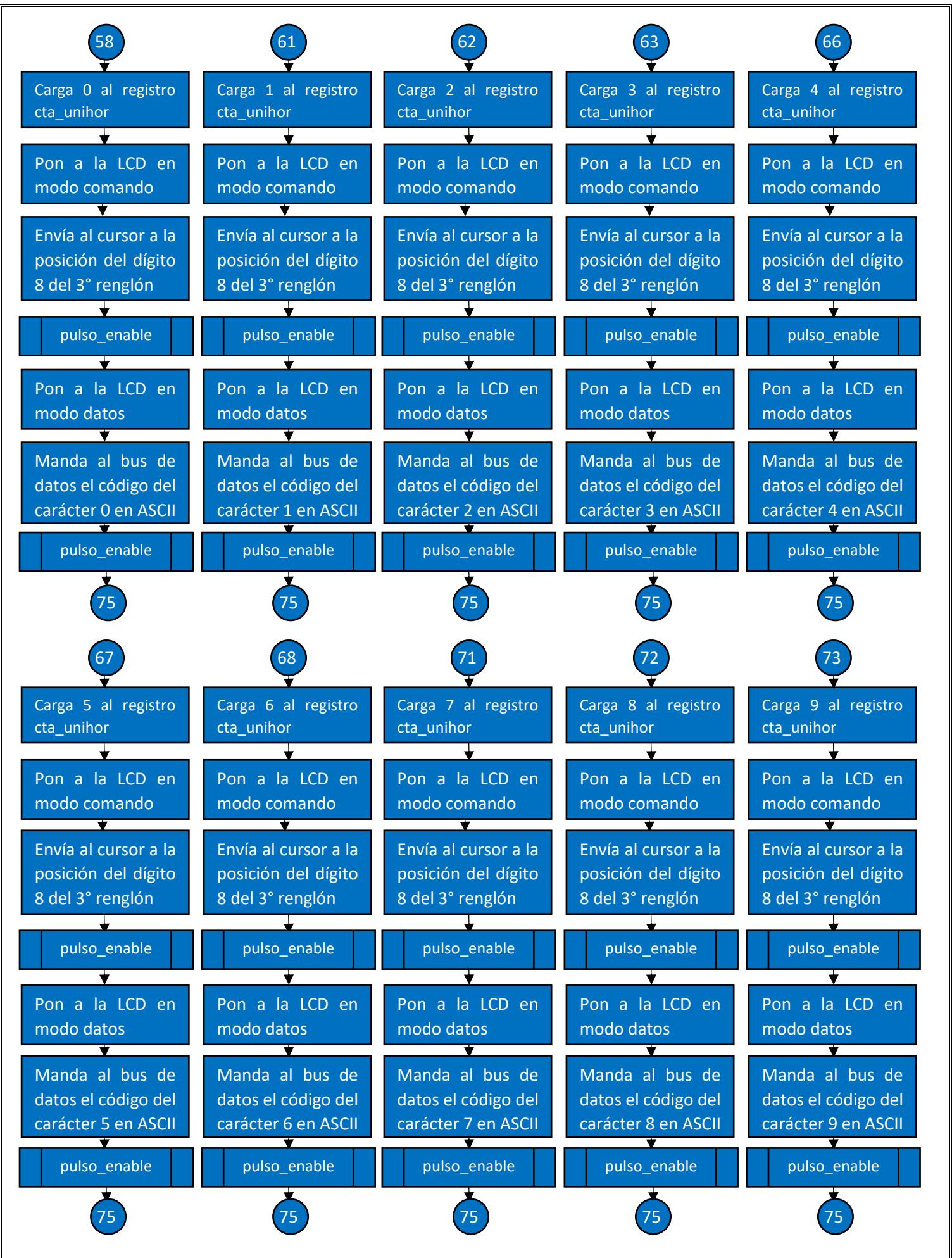


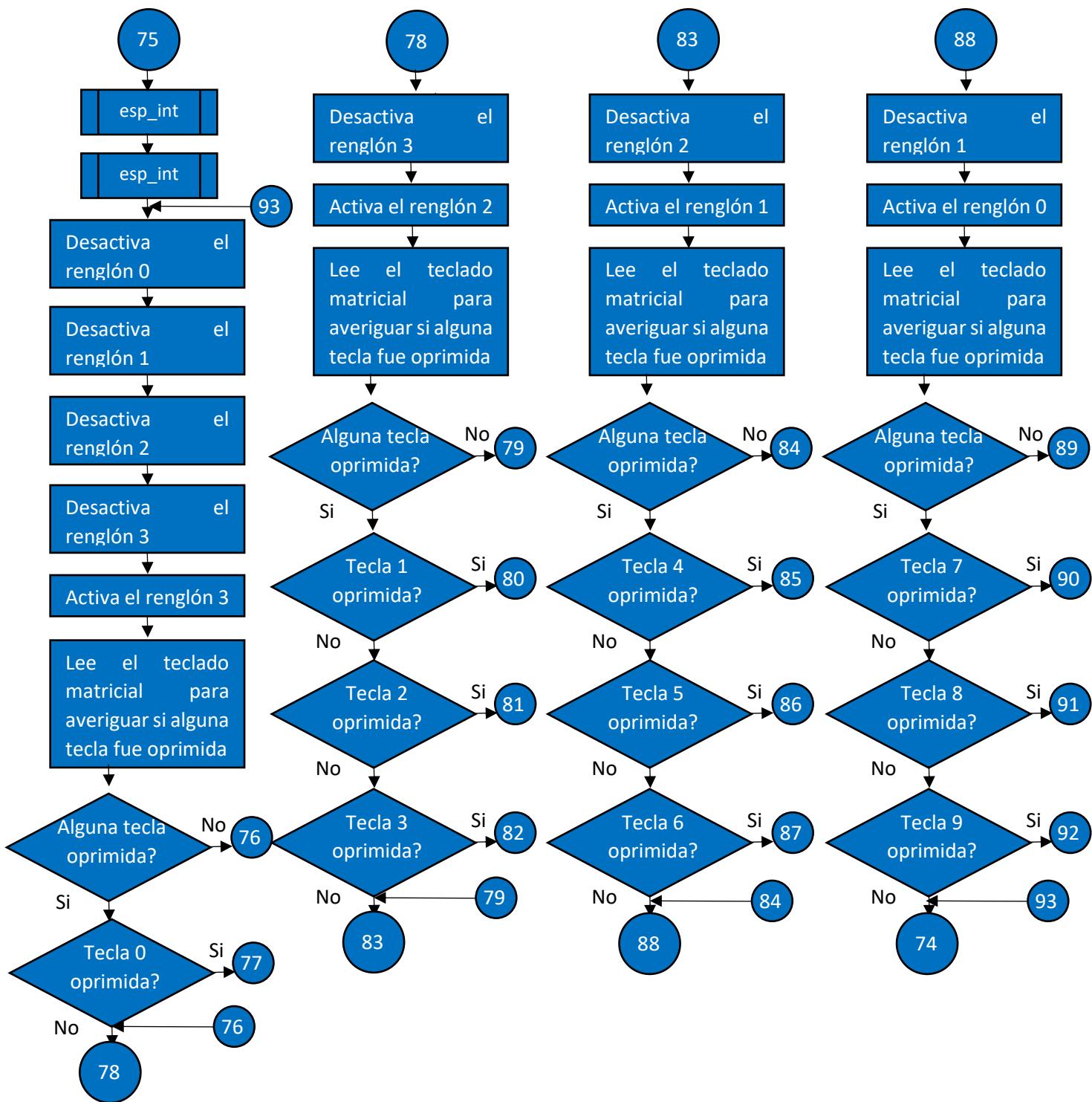


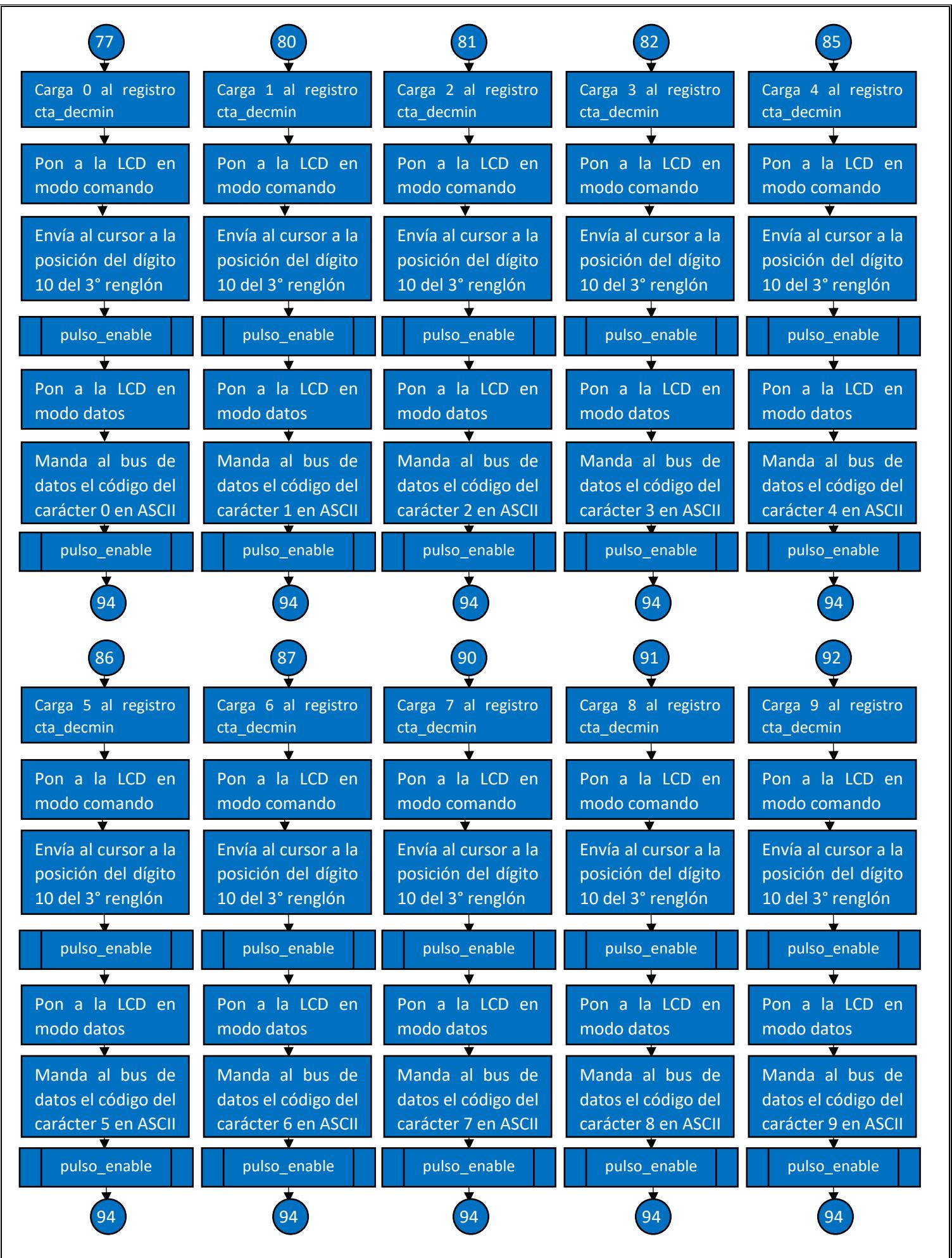


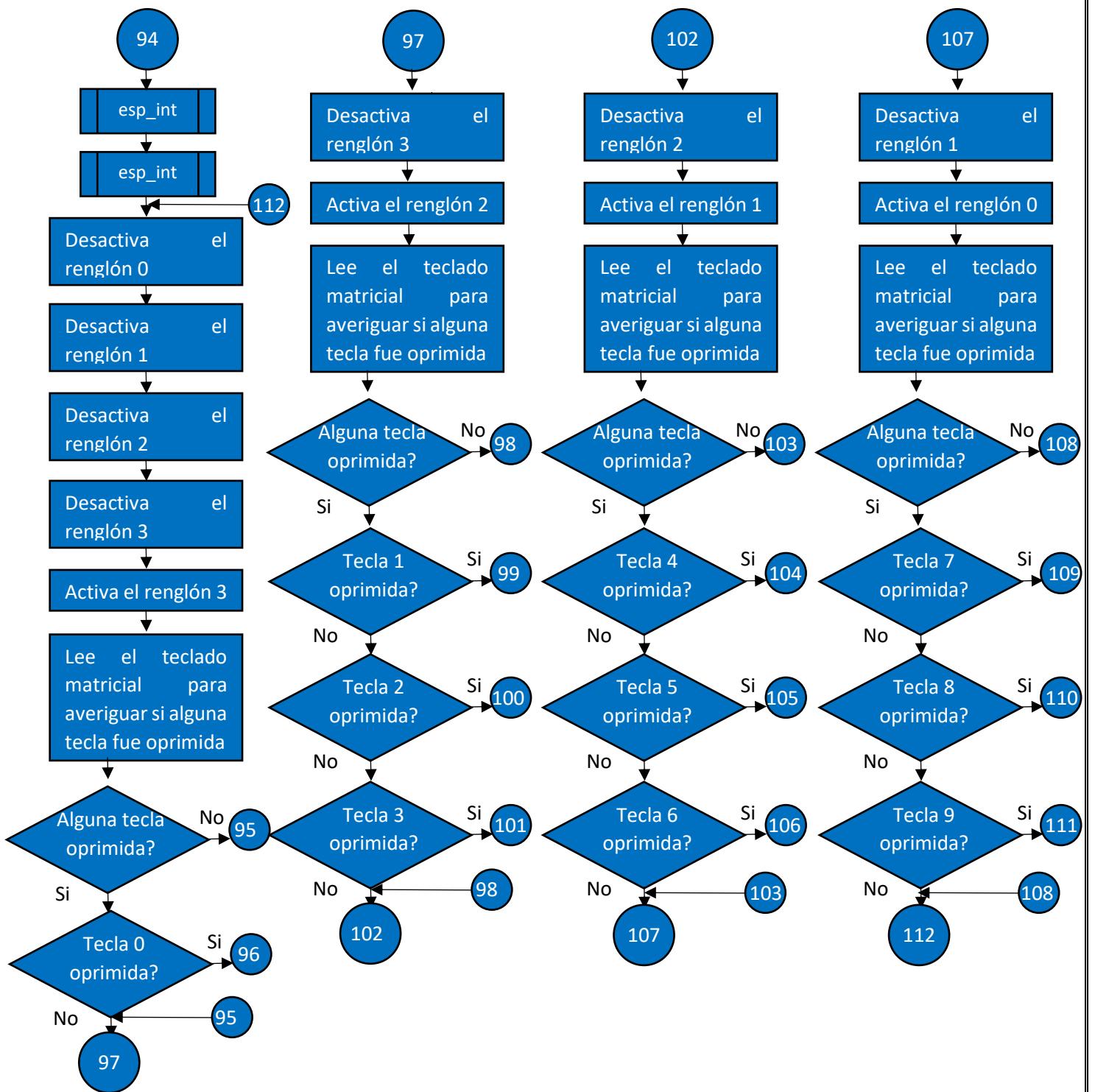


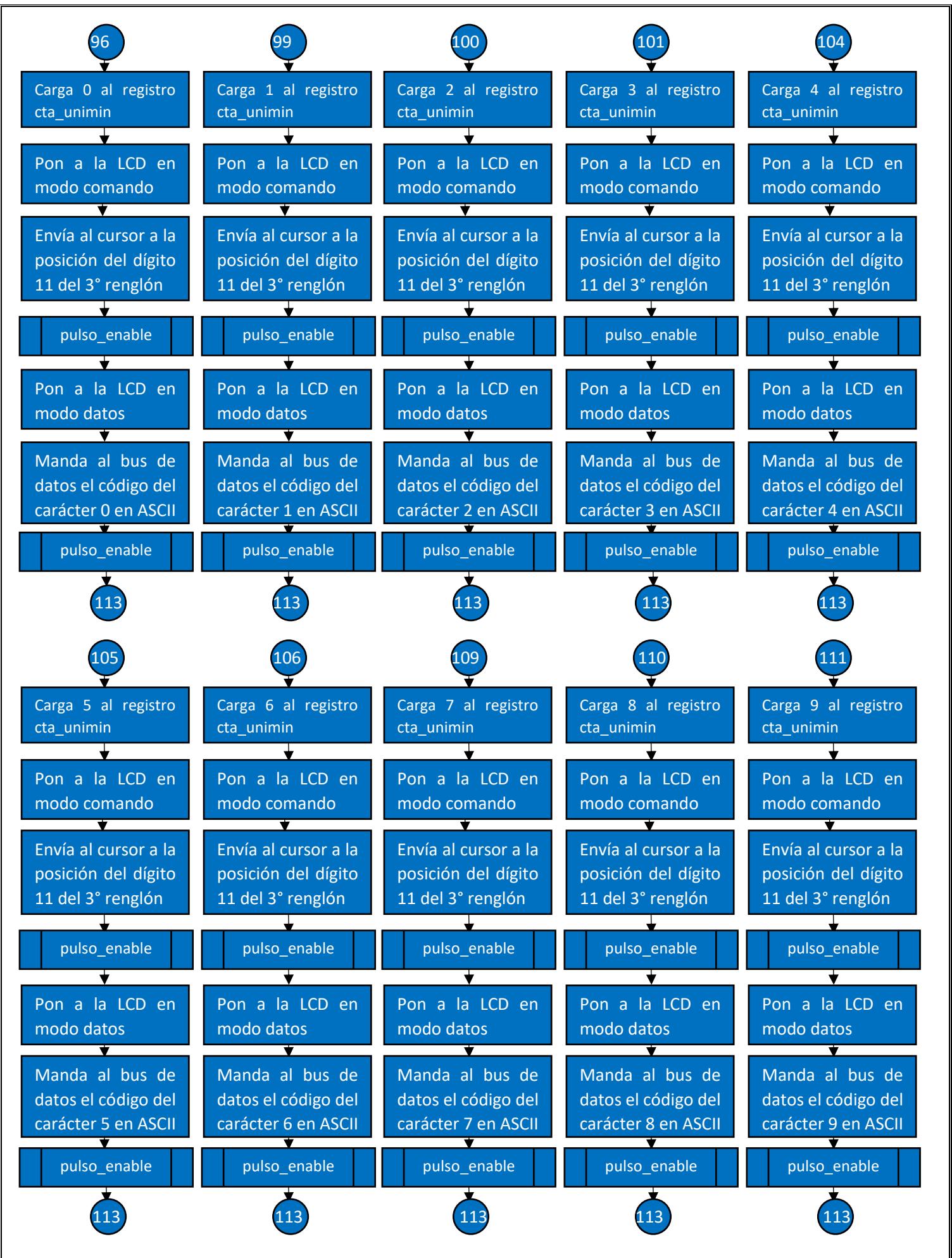


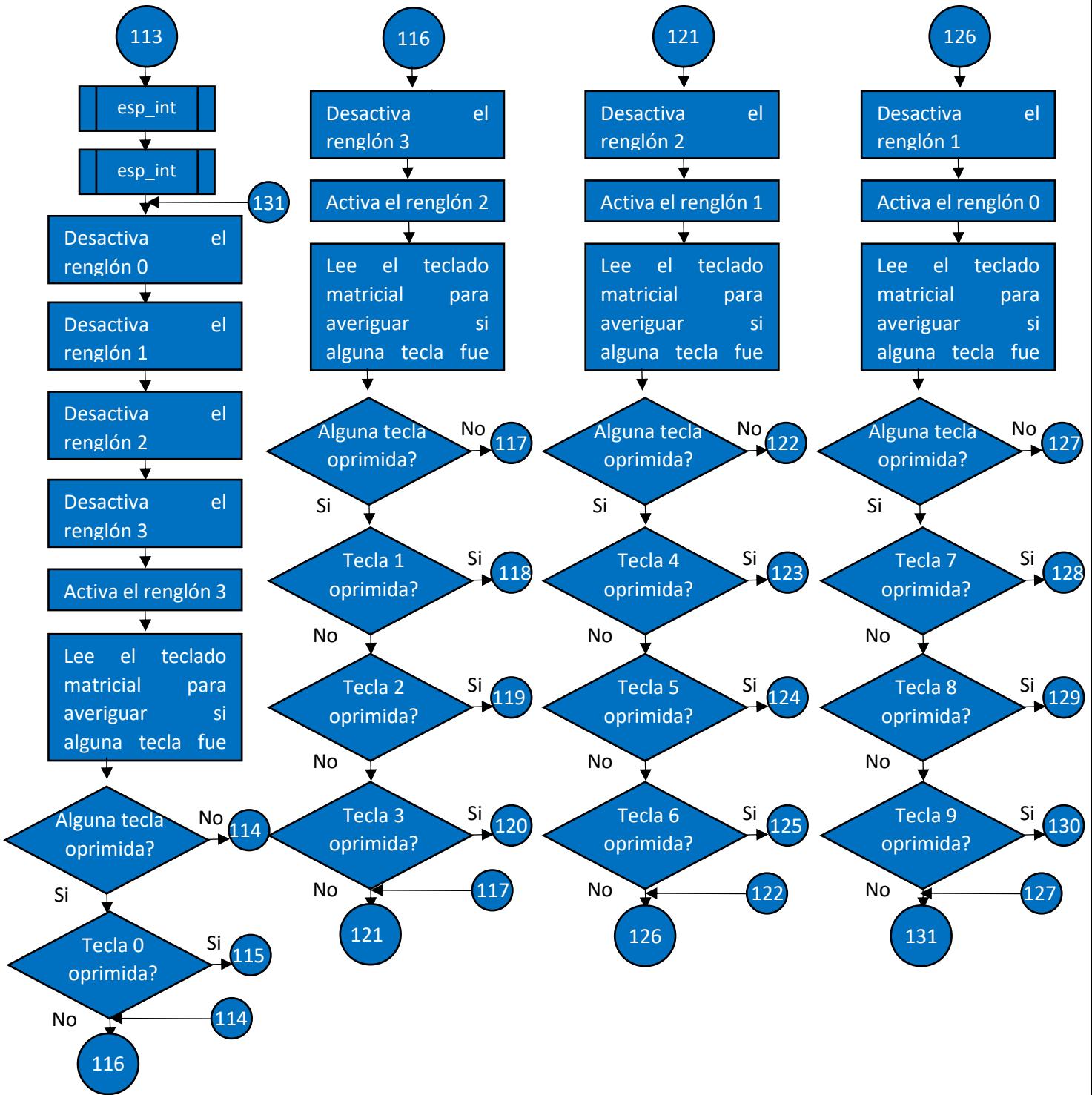


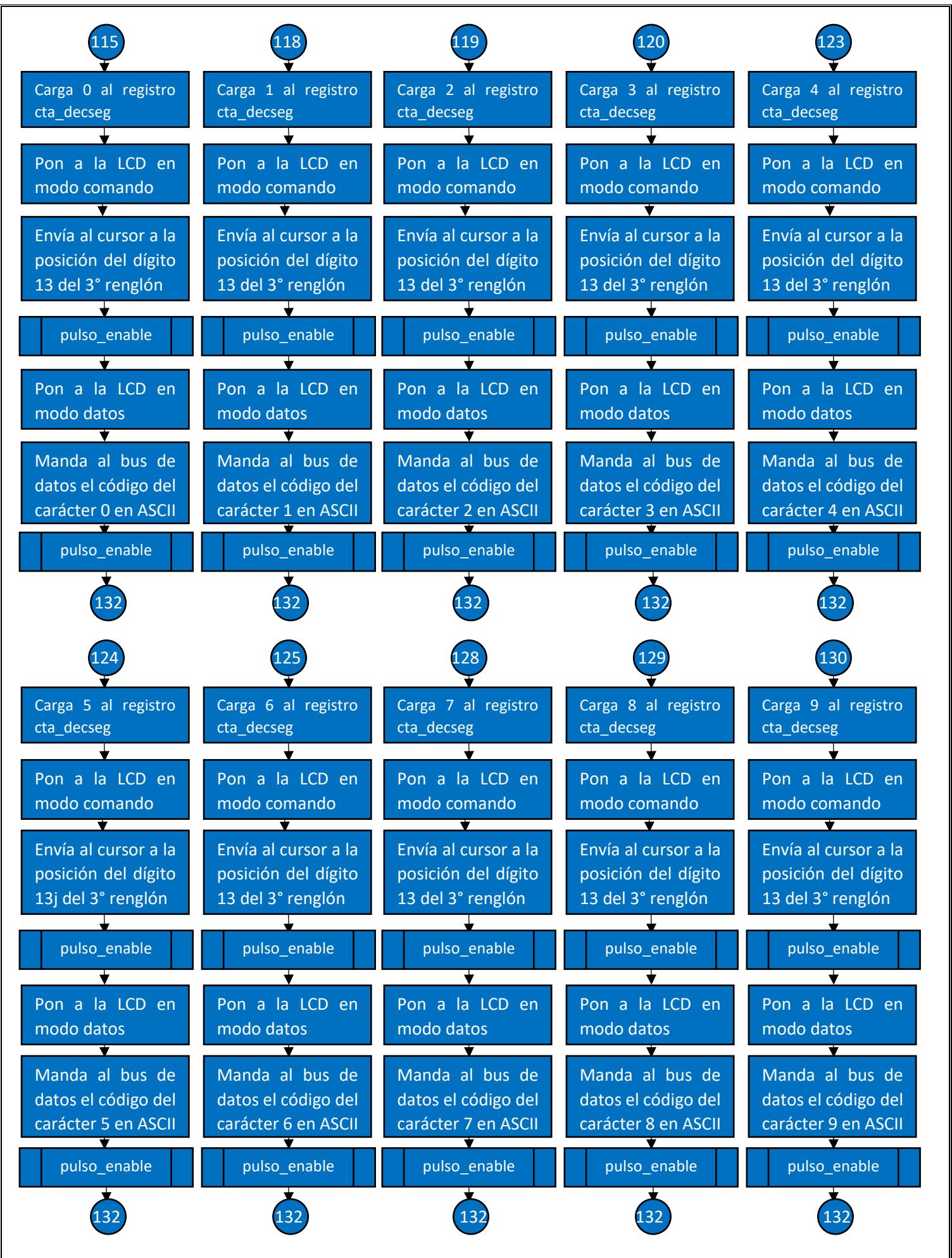


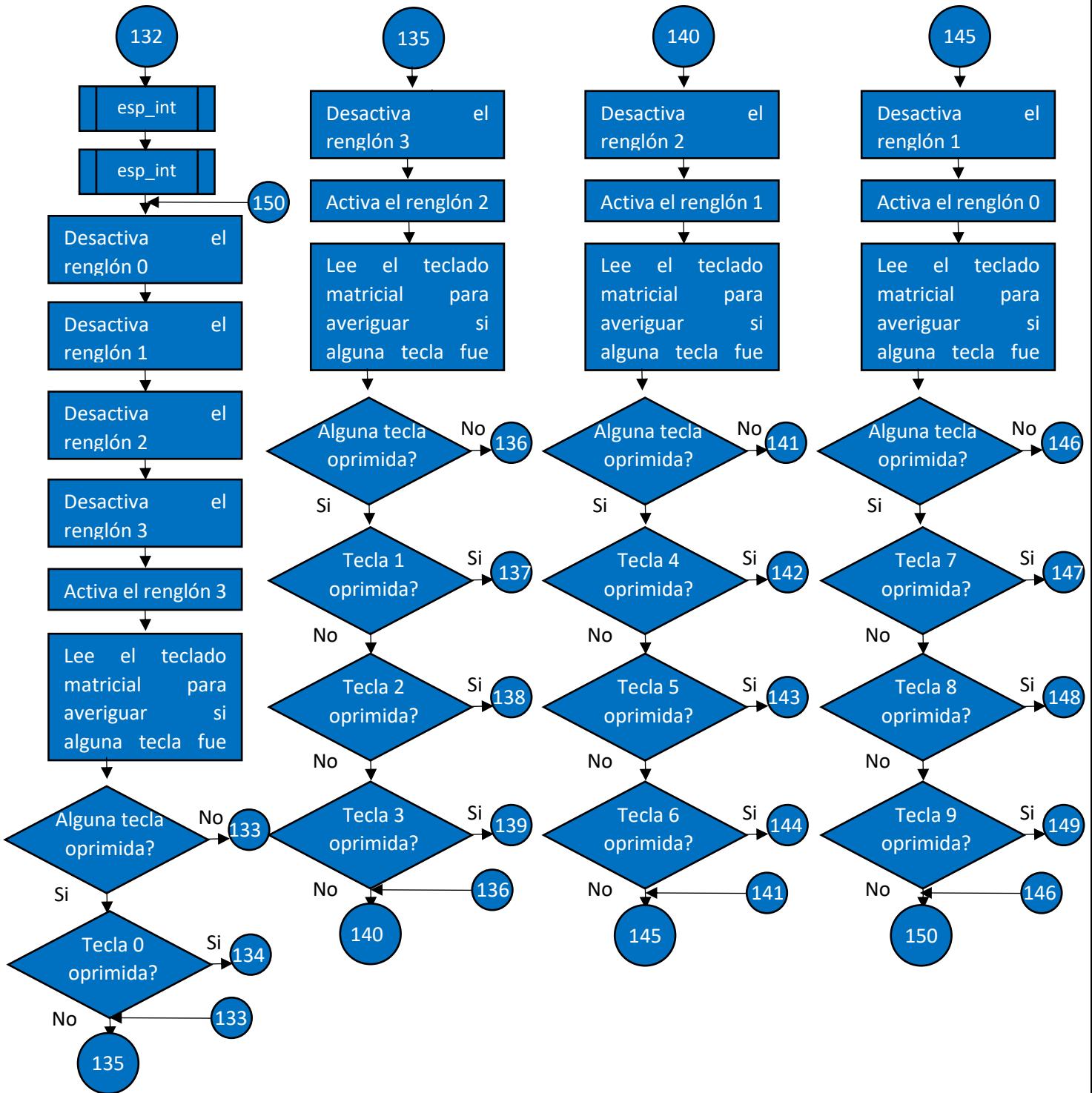


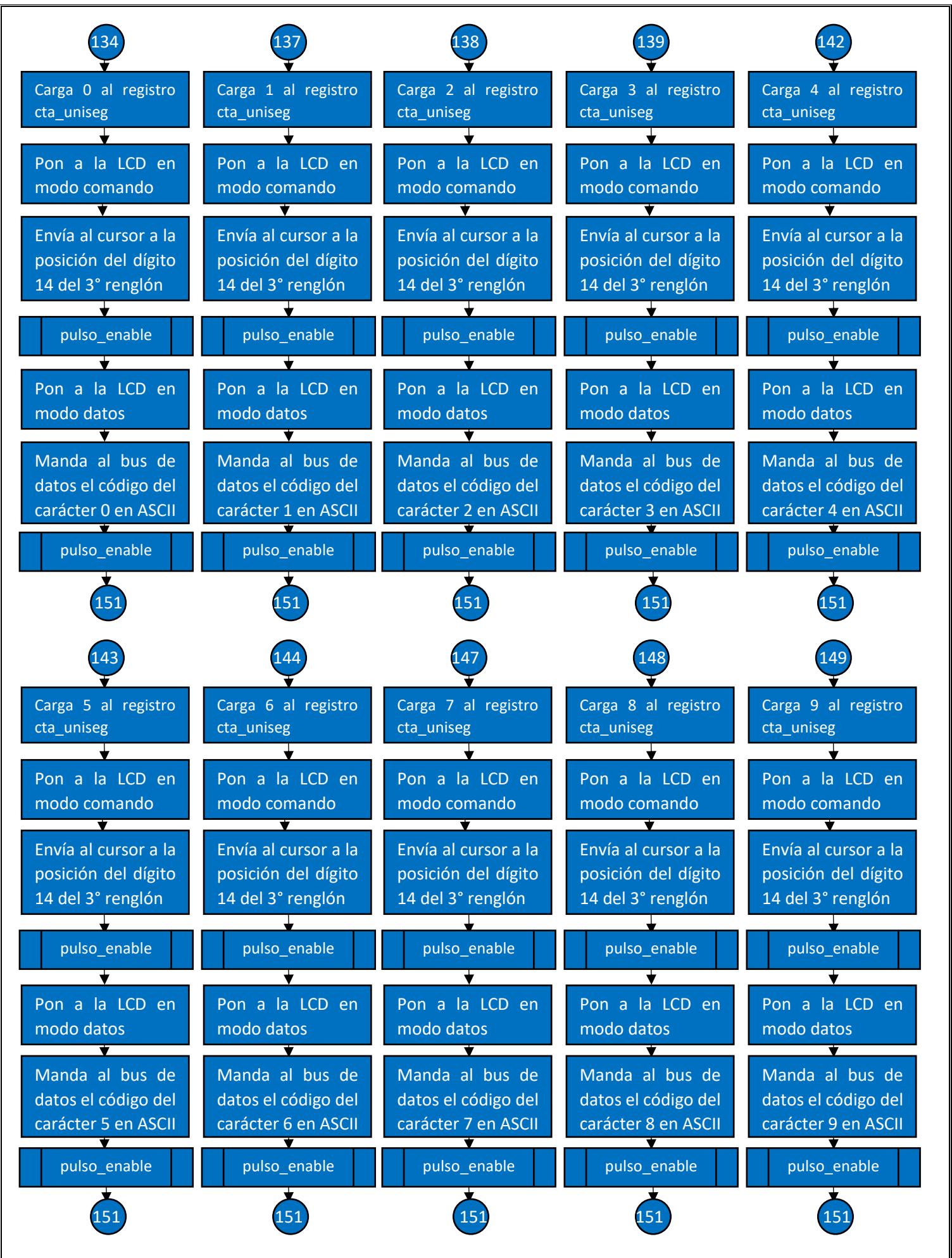


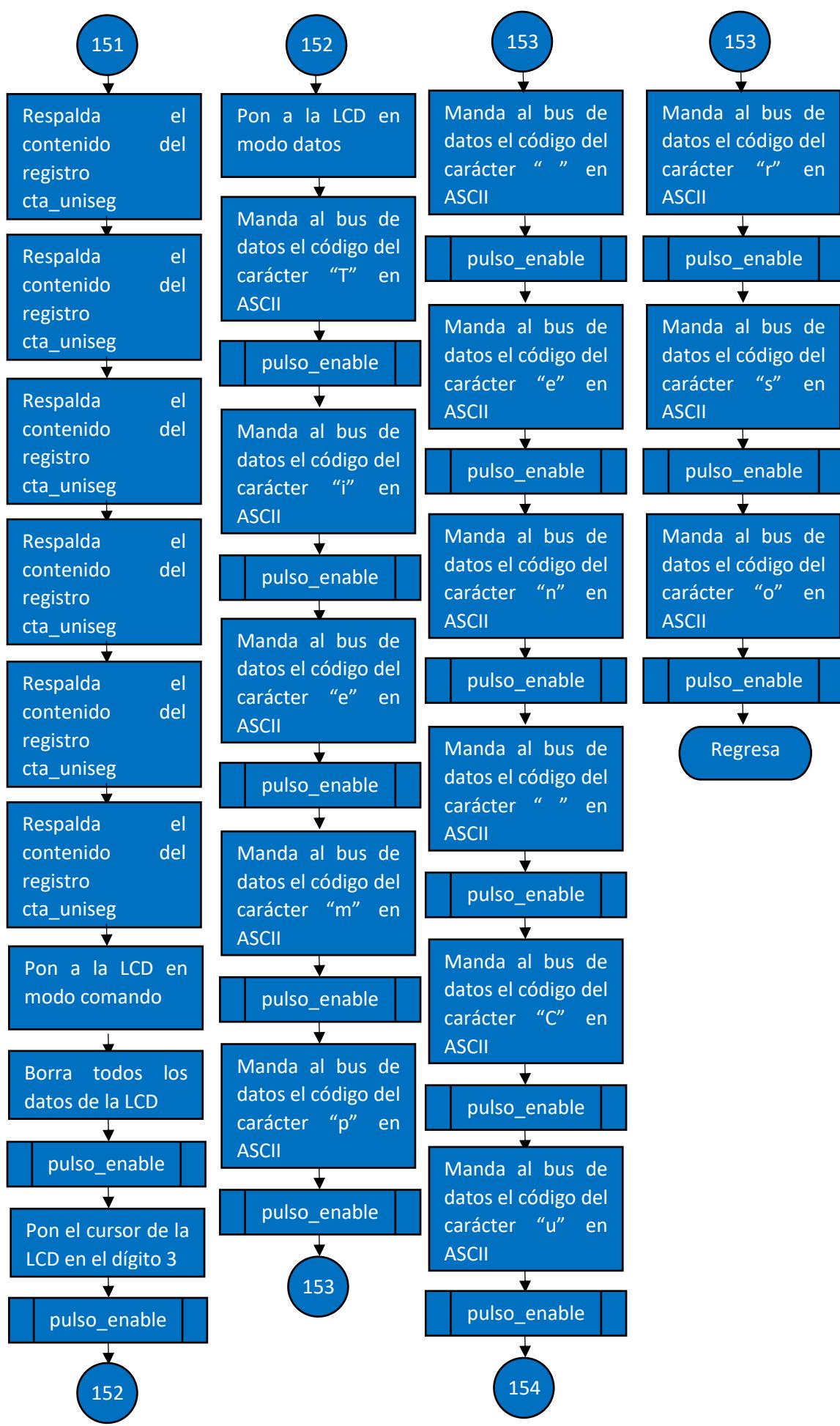












## Conclusiones

### **Alvarado Rodríguez Fernando Bryan**

Considero que el proyecto que hemos desarrollado durante estos 2 años en el área de Sistemas Digitales ha llegado a un grado especial de refinamiento y desarrollo muy bueno y satisfactorio, pues lo que comenzó como un proyecto escolar solo para acreditar las materias, se convirtió en un proyecto a largo plazo que nos gustaría seguir trabajando a nivel profesional. Como en muchas ocasiones de la vida, se presentan adversidades en cualquiera ámbito, pues los retos representan una oportunidad para ampliar el conocimiento. Dentro de las adversidades que presentamos principalmente fue el poder organizarnos en la nueva modalidad a distancia y el poder encontrar la forma de plasmar nuestras ideas y conceptualizarlas en una forma que todo agarrara la forma correcta, tanto en estética, funcionalidad, practicidad y costos, y creo que ahí me ayudo a reflexionar sobre lo importante, difícil, retador, pero a la vez bonito de ser INGENIERO o en nuestro caso futuros ingenieros.

Considero que la parte más difícil de poder realizar este proyecto/prototipo fue el papeleo administrativo de tener que estar llenando formularios de academia constantemente y tener que estar trabajando de manera muy diferente cada asignatura, pues unos profesores solicitaban avances enfocados en su materia cada parcial de forma muy específica y otros solo solicitaron el reporte final y el video, entonces al hacer muchos y muy variados reportes de asignaturas nos consumió demasiado tiempo, tiempo que quizás podríamos haber aprovechado en perfeccionar un poco más a CAD, sin duda alguna todos esos trámites más administrativos creo que quitan un poco de tiempo valioso. A pesar de todo lo mencionado anteriormente, me gustaría extender en esta presente conclusión un reconocimiento y agradecimiento no solo a mis compañeros de equipo, si no a mis amigos por haber complementado tan bien este proyecto, cosa que creo fue esencial para obtener el resultado, pues cada uno de nosotros aportaba de una u otra forma algo para enriquecer al producto y sin duda es algo excepcional.

### **Ayala Bernal Israel**

En general este fue un proyecto trabajado desde tercer semestre, en el cual nos hemos encontramos con muchas barreras de conocimiento en un inicio, debido a que recién estábamos entrando al mundo de los sistemas digitales. Conforme fuimos avanzando, íbamos adquiriendo más conocimientos acerca de dispositivos electrónicos, lenguajes de programación y construcción de circuitos, poco a poco se fue estructurando nuestro dispensador. CAD en lo personal me ha ayudado a reafirmar mis conocimientos mediante la construcción de circuitos principalmente. Se llegaron a tener varios prototipos y propuestas para la elaboración de este, pero al final escogimos las más adecuadas y óptimas.

En estos momentos CAD es un proyecto funcional que podríamos llevar de lo digital a lo físico, y probablemente aún pueda mejorar mucho más si se decide seguir trabajando en algún futuro con él, ya que al en estos momentos tenemos las bases de Técnicos en Sistemas Digitales, pero al trabajarla con el conocimiento que maneja un ingeniero, este puede ser mucho más eficiente.

### **Barrera Castillo Erick Abraham**

Como conclusión puedo mencionar que estoy bastante satisfecho con el resultado final de nuestro proyecto Aula, aunque el cambio fue muy notable a comparación de la idea original, pero personalmente este modelo final es mi favorito, ya que es mucho más completo gracias a todas las funciones que le implementamos a lo largo de su desarrollo, claro que estas modificaciones fueron el resultado de la creatividad e innovación de todos los integrantes del equipo, el cual nos dio al final este gran producto. En mi opinión, todo el trabajo y esfuerzo que le imprimimos a este proyecto nos dio un resultado mejor que el esperado, y claro que también se debe a que llevamos 2 años trabajando en el proyecto, porque a lo largo de este tiempo pudimos aplicar cada vez más de los conocimientos aprendidos de nuestra carrera, e implementarlos en nuestro proyecto, el cual ha ido mejorando hasta quedar como el modelo final.

Me hubiera gustado trabajar completamente en presencial, ya que nos hubiera dado una mayor experiencia en el trabajo manual y la construcción física de todos los circuitos necesarios, pero aunque no fue así, disfrute varias partes del desarrollo del mismo; personalmente me gustó mucho aplicar los conocimientos de la programación de microcontroladores para poder obtener el sistema que lleva a cabo todo su funcionamiento, y de esta manera me doy cuenta que, aunque estamos en línea, los conocimientos esperados se cumplieron, y en relación a las demás materias también estoy seguro, ya que puedo observar el desempeño de mis compañeros en el desarrollo de las demás partes del sistema y es otra prueba de los grandes conocimientos que nos ha ofrecido Bátiz desde el principio.

### **Hernández Caballero Mauricio**

Se ha concluido con el desarrollo de Capsule Automatic Dispenser y analizando los resultados finales se puede llegar a algunas conclusiones. Se han implementado todos los conocimientos adquiridos en la elaboración del proyecto, incluso en algunos casos se ha investigado más de lo visto en clase. Además, a lo largo del tiempo se han actualizado algunos aspectos, conforme a la nueva información adquirida. En general el proyecto se ha desarrollado a la par de nuestra trayectoria académica, por esta razón ha sido un incentivo para el aprendizaje, investigación e interés en algunos temas.

En cuanto al producto final, aunque no tenemos un prototipo físico, en cualquier momento podríamos construir uno, debido a que tenemos investigación, diagramas de circuitos, simulaciones, cálculos y medidas. Por lo tanto, podemos decir que estamos a un paso del producto final y funcional para ser lanzado al mercado. De esta manera CAD es un proyecto totalmente documentado, el cual tiene una utilidad real.

Si se llevara al mercado este producto, podría ayudar a muchas personas y ese es un enfoque que siempre tuvimos presente. Se trata de un producto útil, desarrollado con un amplio respaldo de investigación y trabajo. Para finalizar, PA ha sido el proyecto más complejo que he realizado personalmente, y creo que a todo el equipo nos ha ayudado a desarrollarnos y darnos cuenta de lo que somos capaces.

### **Sánchez Martínez Felipe**

En conclusión, el proyecto el cual se ha generado desde tercer semestre se ha ido perfeccionándose más en estos dos últimos semestres haciéndose cada vez más funcional y atractiva para el usuario, lo que empezó con conocimientos básicos, y no tan avanzados sobre el manejo de los sistemas digitales se han convertido en un proyecto digno de ponerse en venta junto a productos con la misma utilidad y puedo afirmar que sería mejor que muchos si lo pudiéramos hacer físico. Lamentablemente

por la pandemia del virus Covid-19 no pudimos enfrentar todos los retos de hacer todo el proyecto en físico, recordando que haciendo los circuitos en una simulación muchas veces es casi imposible que tengamos un error porque todos los componentes son ideales, en cambio al hacer un circuito tan complejo en físico no contamos con diversos factores, como la corriente necesaria para hacer funcionar cada componente, o no tomar en cuenta algunos valores de voltaje de muchos componentes, etc.

En cambio, haber hecho el circuito en simulación no fue fácil tampoco, porque en si las clases impartidas por los profesores son muy buenas, pero no son suficientes para terminar un proyecto tan complejo, las clases impartidas son como el impulso que le dan a cada estudiante a adentrarse más a fondo y lograr grandes cosas, y ahí está el detalle que le puede dar cada compañero con los que realice este proyecto, sin la aportación de cada uno no pudo haber sido un proyecto casi completo, me hubiera gustado haber terminado este proyecto de forma presencial.

### **Villegas Monroy Emilio**

Finalmente, el proyecto que venimos trabajando desde tercer semestre ha quedado en su forma final. Considero que hemos hecho un trabajo más que notable considerando que nos encontramos en una modalidad no presencial. De todas formas, fue muy interesante ver la manera en que el proyecto fue evolucionando y aún más ser una parte de las cosas que lo hicieron mejorar.

Definitivamente fue un reto en algunas ocasiones especialmente con los reportes al requerir sintetizar información sin perder impresión al mismo tiempo, creo que la expobatiz nos dará una oportunidad de demostrar un poco mejor lo que hemos podido lograr, así como la elaboración de este trabajo debido a la cantidad de detalles que se nos permite agregar.

Personalmente mi parte favorita del proyecto fue durante la elaboración de los diseños del proyecto tanto en forma, organización o demás. El sistema de motores me gustó mucho, sé que como profesionales podríamos hacerlo mejorar, pero en vista de que nunca había hecho algo de este calibre me gustó la forma en que lo manejamos igual que el resultado final. Me gustaría seguir trabajando con más personas como mis compañeros actuales por su forma de dirigir, ayudar, acompañar y otras cosas que ellos hicieron por mí.