Escuela Superior de Ingeniería y Tecnología

Trabajo Fin de Grado

Implementación de un sistema de control de un dispositivo táctil a partir de microcontroladores PIC en lenguaje ensamblador

Titulación: Grado en Ingeniería Electrónica Industrial y Automática

Autor: Sergio Huarte Pulido

Tutor: Evelio José González González

Marzo 2017

Índice

Abstract

The present project describes the components and the process to obtain an assembly language (ASM) code capable to control the communication between a PIC microcontroller (MCU) and a Graphical Liquid Crystal Display (GLCD) with a touch panel.

To accomplish this aim was necessary to learn how all the components operate and the instructions in assembler of this MCU because each brand follow its own instructions.

The components which have been implement are these: Easy PIC V7 development board, PIC microcontroller model PIC18F45K22 emplaced in the development board and monochromatic GLCD 128x64 with a touch panel.

In summary, the process of creation code was developed by making smalls routines to control little process. When these routines work perfectly they were united to others routines that has been made before. This methodology allowed to develop more complex code necessary for this project. In the Anexo II is emplaced the final code.

Capítulo 1.
Introducción

1. Introducción

El presente trabajo de final de grado se centra en el desarrollo de una aplicación en lenguaje de ensamblador, usando la plataforma de desarrollo Easy PIC V7. Este nace de la propuesta del profesor Evelio José González González del departamento de Ingeniería Informática y de Sistemas.

El proyecto es propuesto para alumnos de último curso del Grado de Ingeniería en Electrónica Industrial y Automática, cuya experiencia previa en lenguajes similares se basa en la obtenida en las prácticas de la asignatura de Ingeniería Informática, existiendo la particularidad de que el lenguaje utilizado en dicha asignatura pertenece al lenguaje ensamblador de los microcontroladores AVR, siendo este sustancialmente diferente al que emplean los microcontroladores PIC, al que pertenece el microcontrolador utilizado en este proyecto.

Por tanto, se parte únicamente con el conocimiento de la dinámica necesaria para la creación de un programa en dicho lenguaje, pero siendo necesario adquirir los conocimientos sobre las instrucciones del lenguaje ensamblador de los PIC y la forma de llevarlas a cabo por este microcontrolador.

1.1. Objetivos

El proyecto consta de un objetivo principal, el cual está constituido de una serie de objetivos generales necesarios para llevar a cabo el proyecto de manera exitosa.

Objetivo Principal

- Crear un código capaz de realizar el control de un GLCD 128x 64 con panel táctil a través de un microcontrolador PIC en lenguaje ensamblador.

Objetivos generales

- Usar la placa de desarrollo Easy PIC V7, como soporte para la realización del proyecto y así probar su funcionamiento para el desarrollo de futuros trabajos.

- Obtener los conocimientos necesarios en el manejo del lenguaje ensamblador de los microcontroladores PIC.
- Aprender el manejo de las herramientas de desarrollo necesarias para la creación del código.
- Realizar una aplicación capaz de probar los aspectos principales, que rigen el funcionamiento de un panel táctil.

1.2 Estructura de la memoria del proyecto

El presente proyecto consta de seis capítulos en los que se explicará el procedimiento desarrollado para cumplir con los objetivos. También se describen de los componentes y herramientas usados, así como el costo total del mismo y las conclusiones finales.

Se ha decidido separar de esta manera, para dejar en la parte central de la memoria únicamente la descripción de cómo se obtuvo la solución final del proyecto, apartando de la explicación de aspectos que puedan resultar obvios, como los componentes del hardware, su configuración o las herramientas de desarrollo usadas.

Capítulo II: En él se han descrito aspectos básicos de los componentes, así como su funcionamiento y algunas características generales necesarias para el desarrollo del software, como puede ser en el caso del microcontrolador la explicación de los registros usados o en el caso de la placa de desarrollo, la configuración de sus jumpers y switches.

Capítulo III: Aquí se detallarán las herramientas de diseño necesarias para la materialización del código del proyecto y como se usaron.

Capítulo IV: Se trata de la parte central del proyecto, donde se detalla el trabajo de programación, en él se puede intuir el proceso de aprendizaje realizado a lo largo de su desarrollo, las dificultades que se han ido presentando y la justificación de porque se tomaron ciertas soluciones en la construcción del código.

Capítulo V: Este capítulo se trata el presupuesto del proyecto, estimando un precio total del desarrollo del mismo.

Capítulo VI: última parte del proyecto donde se valora el grado de consecución de los objetivos y lo que ha supuesto el proceso de desarrollo desempeñado para el alumno.

Capítulo 2. Descripción del hardware

2. Descripción del hardware

En este capítulo de la memoria se procederá a describir los componentes que se utilizaron para el desarrollo del proyecto, así como las configuraciones necesarias que se llevaron a cabo para establecer la comunicación entre ellos.

Estos son los tres dispositivos de los que consta este proyecto:

- Placa de desarrollo EasyPIC V7 del fabricante MikroElektronika.
- Microcontrolador modelo PIC18F45K22 del fabricante Microchip Technology.
- GLCD modelo WDG0151 de 128x64 del fabricante Winstar.

Figura 2.1: Placa de desarrollo Easy PIC V7

A pesar, que durante el apartado de desarrollo del software se especifica algunos aspectos de cómo se configura cada uno de los componentes, aquí se explicarán algunas características relevantes, así como la configuración final que se requiere para el funcionamiento del programa desarrollado.

Primero se hará la descripción de la placa de desarrollo debido a su importancia en el proyecto, por ser el elemento central que sirve de unión entre los demás componentes.

2.1. Easy PIC V7

La placa de desarrollo modelo EASY PIC V7, está preparada para el uso de microcontroladores PIC de 8 bits, viene con 8 zócalos dispuestos para MCUs con encapsulado Dual In-line Package (DIP) que van desde los 8 pines a los 40.

Viene con espacio para disponer de una pantalla LCD alfanumérica de 2x16, una pantalla gráfica 128x64, tiene un display 7 segmentos incorporado, cuenta con pulsadores, LEDs, resistencias de "pullup" y "pulldown" en cada terminal de E/S, también cuenta con un programador sobre la propia placa y un depurador en circuito, una fuente de alimentación capaz de suministrar 3.3 V o 5V, además de puertos RS 232 Y USB, dispone de 2 conexiones para termómetros, otras dos para osciladores, viene con un zumbador incorporado, es posible añadirle dos click boards mediante la interfaz mikro BUS (las cuales pueden añadir funciones a la placa como conexión bluetooth, GPS, etc…).

Algunos de los componentes de la placa pueden entrar en conflicto con otros, al usar los mismos buses para comunicarse con el MCU, así que para que el microcontrolador pueda acceder a las funciones de las que dispone la placa, es necesario hacer uso de una serie de switches y jumpers, los cuales establecerán las conexiones con los buses de comunicación que se necesiten. Su configuración viene descrita en los manuales del fabricante.

La caja viene con todos los dispositivos a utilizar en el proyecto. También se suministra un cable USB, el lápiz para la pantalla táctil, los manuales y un CD con Drivers.

2.1.1. Condiciones iniciales del dispositivo

En un principio la placa viene con el microcontrolador PIC18F45K22 incluido en el zócalo DIP40 y un cristal oscilador de 8Mhz situado en la conexión X1.

La pantalla GLCD 128x64 se suministra en la caja, pero hay que conectarla en el espacio dispuesto para ella, esta cuenta con un panel táctil incorporado, el cual es necesario conectarlo en la posición CN29.

La placa ya viene con una configuración de jumpers y switches de fábrica, para trabajar con los dispositivos emplazados en un principio. No obstante, hay que revisarlos antes de proceder al encendido, para evitar producir un cortocircuito de manera accidental en algún pin.

A continuación, se detallará la configuración necesaria de los jumpers y los switches. A pesar de que en el Capítulo de Desarrollo del software se fue variando (solo la configuración de los switches), aquí se indica la configuración final con la que se trabaja para el funcionamiento del programa.

2.1.2. Configuración de los jumpers

Primero se comprobó la configuración de la fuente de alimentación, su funcionamiento puede variar dependiendo de la situación de los jumpers J5 y J6. Se situó el jumper J6 en la posición USB, pudiendo usar este puerto para suministrar la alimentación a la placa. También se comprobó que el J5 estuviera situado en la posición correcta para que la fuente trabaje a 5V.

Figura 2.2: Jumpers J5 y J6 (Manual del fabricante)

Se comprobaron los jumpers que corresponden a la parte del mikroporg (J1, J2, J8 y J9). Son los encargados de fijar el esquema de conexiones necesarios para poder realizar la programación del MCU y dependiendo del zócalo de la placa en que se emplace este, se han de colocar de diferente manera.

Para este caso, al estar el microcontrolador emplazado en el zócalo DIP40 ha de tener la siguiente configuración “(Ver figura 2.3)”:

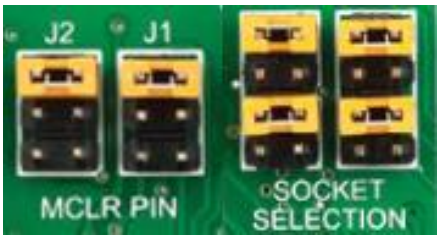


Figura 2.3: Jumper mikroProg J2 y J1 (Manual del fabricante)

En la misma posición de la placa se encuentra otro jumper(J19), se usa para conectar el pin RE3 del MCU al pulsador que dispone la placa como botón de RESET. En nuestro caso se conectó con dicho pulsador, puesto que en el software, programaremos este pin del puerto para ese cometido.

Figura 2.4: Jumper J19 (Manual del fabricante)

También hay que prestar atención a los jumpers (J22, J7, J10, J23) que realizan la conexión con los VCAP de la placa, en el manual de la misma se muestra una lista de los MCU que deben ir conectados al VCAP. No fue necesario que se usara para el MCU.

En la posición J13 se encuentran los jumpers encargados de conectar el oscilador de cristal, dispuesto en la conexión de la placa con el pin RA6 y RA7 del MCU, como vamos a hacer uso del cristal emplazado en la posición X1, se debe colocar el jumper en OSC.

Figura 2.5: Jumper J13 (Manual del fabricante)

Otro jumper al que se le prestó atención fue el J17. Este marcará la lógica de los pulsadores de la placa, en caso de estar en la posición VCC, aplica un 1 lógico cada vez que se use un pulsador, en caso de estar en GND aplica un 0. Se dejó en la posición VCC.

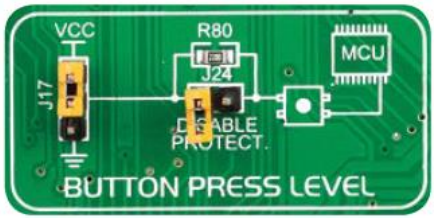


Figura 2.6: Jumper J17 y J24 (Manual del fabricante)

Por último está el Jumper J24, el cual ha de permanecer desactivado, ya que desactiva la protección de los pulsadores conectándolos directamente con GND o VCC (dependiendo del J17), si se conecta este jumper se desactiva la protección y puede dañarse la MCU con un uso erróneo.

El resto de jumpers corresponden a otros dispositivos de la placa que no se usarán para este proyecto.

2.1.3. Configuración de los switches

Los switches de las posiciones SW1 y SW2 son de uso exclusivo para la conexión con otros dispositivos a través de RS 232 o USB. No será necesario su uso.

Para la comunicación con la GLCD, hay que configurar los switches situados en las posiciones SW3 y SW4, estos se encargan de conectar los pines de los puertos del MCU con los LEDs de los terminales de E/S de la placa, la I^2C EEPROM, el display 7 segmentos, la pantalla LCD 2X16, el GLCD y su pantalla táctil.

Como el proyecto se centrará en el uso del GLCD con la pantalla táctil, se configurarán los switches en exclusiva para ese cometido. Los switches de la posición SW3, son los necesarios para que el MCU reciba la información del panel táctil y los de la SW4 sirven para elegir el brillo de la pantalla, la cual se configuró al máximo.

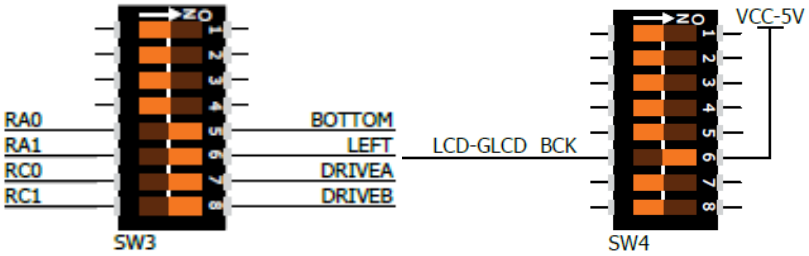


Figura 2.7: Configuración SW3 y SW4 (Manual del fabricante)

Con estas configuraciones para los Jumpers y los Switches, estaría dispuesta la placa para funcionar sin ningún conflicto con los componentes elegidos.

2.2. Microcontrolador PIC

Los PIC son microcontroladores fabricados por la marca Microchip, estos cuentan con una arquitectura tipo Harvard, cuya principal característica es que su CPU está conectado a la memoria de programa y a la de datos a través de dos buses separados.

Se clasifican según el tamaño de su bus de datos y el de sus registros. En el caso de PIC, estos se dividen en tres grandes familias de 8, 16 y 32 bits de CPU, dentro de cada una se dividen a su vez dependiendo de las propiedades de cada modelo.

Para este proyecto el MCU seleccionado es el PIC18F45K22, que se suministra ya emplazado en la placa de desarrollo.

Como particularidad en este microcontrolador cabe resaltar la existencia del WREG, que es un Special Function Registrer (Ver apartado 2.2.2. Organización de la memoria), este es el registro de trabajo y es necesario para que la ALU realice cualquier operación, puesto que la hará entre el valor cargado en este registro y otro de la memoria. Esto hace que para cualquier operación, primero se ha de cargar el valor en el WREG y posteriormente el del otro registro con el que se quiera operar, siendo necesario realizar dos instrucciones. El resultado de la operación se puede volcar en uno de los dos registros usados.

2.2.1. Microcontrolador PIC18F45K22

Este es un MCU modelo PIC18 de 8 bits, corresponde a los modelos más avanzados dentro de esta familia, cuenta con una CPU de alto rendimiento con reduced instruction set computer (RISC).

Figura 2.8: Microcontrolador PIC18F45K22

Este viene en un encapsulado de 40 pines PDIP incluido en su zócalo correspondiente y estas son algunas de las características generales de su familia:

- 8 bits de datos y 16 bits de instrucciones
- Multiplicador en un ciclo 8x8 por hardware
- Cuenta con 31 niveles de Pila
- Cuenta con un oscilador interno de hasta 16 MHz
- Puede trabajar a una velocidad de hasta 64Mhz
- Cuenta con prioridades en las interrupciones
- Tiene 75 instrucciones estándar, así como 8 nuevas instrucciones extendidas

La siguiente tabla muestra las características del modelo PIC18F45K22, así como de los de periféricos con los que cuenta.

Device	Program Memory		Data Memory		I/O(1)	10-bit A/D Channels(2)	CCP	ECCP (Full-Bridge)	ECCP (Half-Bridge)	MSSP		EUSART	Comparator	CTMU	BOR/LVD	SR Latch	8-bit Timer	16-bit Timer
	Flash (Bytes)	# Single-Word Instructions	SRAM (Bytes)	EEPROM (Bytes)						SPI	I²C™							
PIC18(L)F45K22	32K	16384	1536	256	36	30	2	2	1	2	2	2	2	Y	Y	Y	3	4

Tabla 2.1: Características del PIC18F45K22 (Datasheet)

2.2.2. Organización de la memoria

La memoria está dividida en dos bloques, estos cuentan con buses independientes de datos.

Primero se encuentra la memoria de datos, que en este dispositivo es del tipo Static Ramdon Access Memory (SRAM). Para leerla no es necesario el sincronismo de un reloj, únicamente es necesaria una instrucción. Esta pertenece a la memoria volátil del programa, perdiendo su contenido en cada apagado del dispositivo.

La otra memoria es de tipo flash, se trata de la memoria del programa y se guardan en ella las instrucciones y constantes necesarias para la ejecución del programa, es una memoria no volátil.

2.2.2.1. Memoria de datos

Esta memoria cuenta con un ancho de 8 bits de datos y cuenta con un BUS de dirección de 12 bits, está dividida en 16 bancos, algunos de los cuales no se encuentran implementadas sus direcciones de lectura. Esta se compone principalmente dos tipos de registros:

- Registros de Propósito General (GPR): Son los registros encargados de guardar datos y operaciones, que se lleven a cabo durante el funcionamiento del programa.
- Registros de Funciones Especiales (SFR): Estos registros son los encargados de albergar la información para la configuración de los periféricos del MCU, tales como los puertos de E/S, los temporizadores, los conversores analógico-digital, etc…

Este esquema corresponde a la organización de la misma, indicando las divisiones que presenta y la situación de los GPR y SFR. También se puede ver que desde el banco 6 hasta el 14, se encuentran las direcciones no implementadas, cuya lectura devolverá 0.

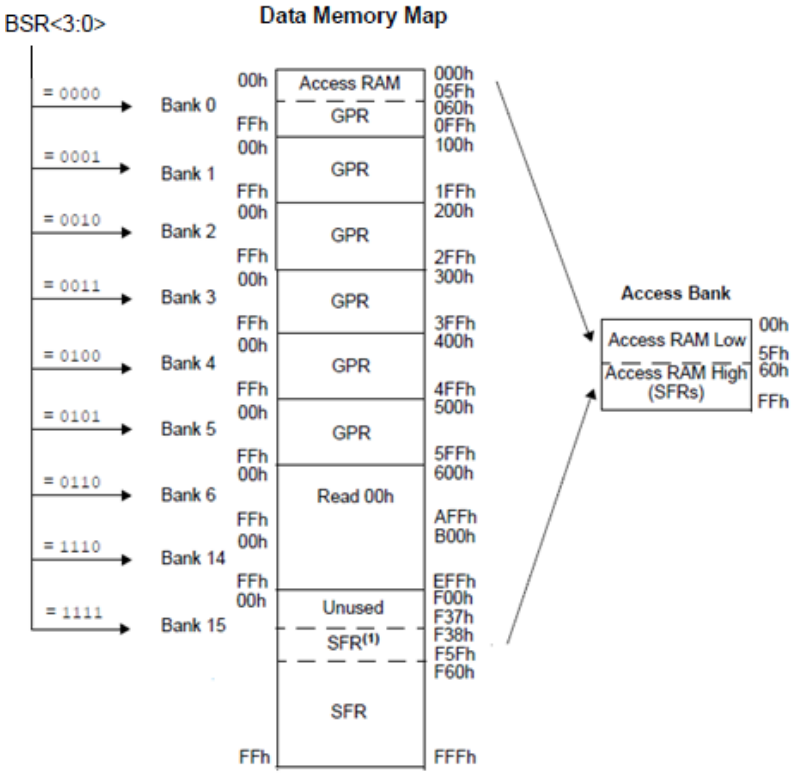


Figura 2.2: Esquema memoria de datos (Datasheet)

2.2.2.2. Memoria del programa

La memoria Flash es la encargada de la memoria del programa, en esta se puede leer, escribir y borrar durante su operación normal.

Cada registro de la memoria Flash del programa tiene un tamaño de 16 bits y cuenta con un BUS de direcciones de 21 bits, en medio se encuentra un espacio de SRAM utilizado para la pila de direcciones, este PIC cuenta con 32 Kbytes de capacidad, lo cual implica que no todas las direcciones de acceso están implementadas.

Para ir leyendo las instrucciones, el contador del programa (PC) irá indicando la posición de la instrucción que se ha de ejecutar. Este PC está dividido en tres registros conocidos como PCL, PCH y PCU. Para modificar el PC es necesario modificar estos registros, este solo reconoce las posiciones pares, puesto que el LSB del registro PCL siempre es 0.

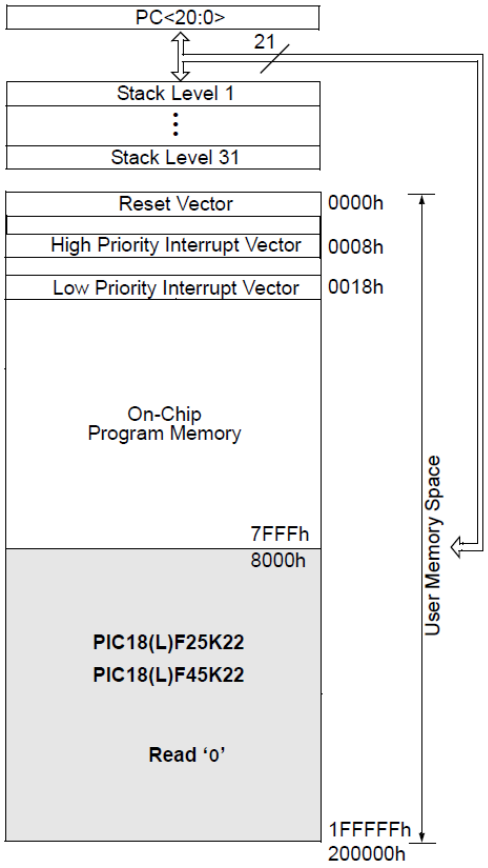


Figura 2.3: Esquema memoria del programa (Datasheet)

En la memoria del programa, se encuentran algunas posiciones especiales a tener en cuenta, la posición 0x000000 corresponde al vector de RESET y será la posición a la que apunte el PC después de cada reinicio. También son importantes las posiciones 0x000008 y la 0x000018 puesto que el PC apuntará a ellas al producirse una interrupción, dependiendo de la prioridad que tengan estas interrupciones irá a un vector u otro.

Una particularidad que presenta la memoria del programa, es la posibilidad de usar el Registro de Función Especial TBLPTR, para situarse en un registro concreto de la memoria de programa, pudiendo leerlo o modificarlo. Este permite leer o modificar las posiciones pares e impares, de la memoria del programa y guardar su resultado en el Registro de Función Especial TABLAT, permitiendo almacenar dos datos de 8 bits, en cada posición de la memoria del programa que deseemos, siendo útil para la creación de tablas en el programa "(ver Capítulo 4. apartado 4.4. Lectura de tablas)".

2.2.3. Oscilador

El MCU cuenta con la posibilidad de obtener la señal de reloj de diferentes maneras, este se puede configurar para obtenerla de un reloj externo, también puede obtenerla haciendo uso de osciladores externos, ya sea un cristal de cuarzo, un resonador cerámico o un circuito RC, otra opción con la que cuenta este modelo, es la de usar un oscilador interno para obtenerla.

Además, cuenta con un phase locked loop (PLL), que permite elevar la frecuencia de entrada por 4, este está diseñado principalmente para elevar a 16 MHz frecuencias de 4 MHZ. La selección de una de estas opciones para el reloj del sistema se realizó a través de los bits de configuración, al principio del código del programa, mediante la directiva CONFIG "(Ver Capítulo 3. apartado 4.2. Bits de configuración)".

Por último, resaltar que la velocidad de ejecución de instrucciones, se demora 4 ciclos de reloj del oscilador.

2.2.4. Puertos de I/O

Este modelo cuenta con 5 puertos, a los cuales pertenecen 36 de los 40 pines del MCU. Cada puerto cuenta con una serie de registros especiales, que configuran el modo de funcionamiento de los mismos, pudiendo elegir entre usarlos como entradas o salidas digitales y algunos de ellos como entradas analógicas.

La selección de estos modos de funcionamiento, se hace a través de la configuración de los registros *TRISx* y *ANSELx*.

- *TRISx:* Cada puerto cuenta con este registro, el cual realiza el control de flujo de los pines del puerto. Cada posición del registro pertenece a un pin determinado, asignando un 1 (bit = 1) en la posición correspondiente del registro, hará que el pin correspondiente actué como entrada, si ponemos un 0 (bit = 0) actuará como salida.
- *ANSELx:* Este registro se encarga de configurar el puerto como una entrada analógica o digital, solo se pueden ver afectados por él algunos pines y se activa como entrada analógica al situar un 1 (bit = 1) en la posición correspondiente al pin del puerto, en caso de dejarlo en 0 (bit = 0), actuará como entrada digital dependiendo.

Las configuraciones de estos registros también tienen efecto sobre algunos periféricos con los que cuenta el MCU, esto será importante tenerlo en cuenta a la hora de utilizar los pines de ciertos puertos, puesto que inhabilitaría el uso de los mismos para el desempeño de otras funciones, en este caso el diseño de la placa ya cuenta con esto y se han conectado evitando que se dé este caso.

2.2.5. Conversor Analógico-Digital (ADC)

Este MCU cuenta con 30 canales para el conversor, 28 vinculados a los pines de los puertos y 2 a canales internos, tiene una resolución de 10 bits y solo puede estar conectado a un canal a la vez. Para su uso en el proyecto es necesario configurar el pin elegido como entrada analógica y posteriormente indicar en los registros del ADC, que canal corresponde a ese pin para poder leer el valor.

Es necesario que el valor de la lectura se encuentre entre los 0 V y los 5 V o esté entre los rangos definidos por los pines VREF+ y VREF-, también es necesario que se fije el tiempo de adquisición para el ADC.

Todas estas consideraciones se configurarán haciendo uso de los registros *ADCON0, ADCON1* y *ADCON2*, los cuales llevan el control del conversor, a su vez el resultado se volcará en dos registros denominados *ADRESH* y *ADRESL*.

- *ADCON0:* En este registro se lleva a cabo la elección del canal para realizar la lectura con el ADC, también se activa el funcionamiento del mismo y se encuentra el bit de estado de la conversión, el cual indica si ha terminado o no.
- *ADCON1:* Configura la selección de un trigger especial para comenzar la conversión del A/D, también señala la configuración escogida para los valores de referencia que se usarán.
- *ADCON2:* Este último detallará la selección del reloj para la conversión, el tiempo que llevará la misma y como se volcará el resultado de 10 bits en los registros *ADRESH* y *ADRESL*, pudiendo ser con justificación a la derecha o a la izquierda.

Para la configuración del ADC se especifica una serie de pasos. Para este proyecto se han de configurar los pines RA0 y RA1 como entradas analógicas puesto que son los que van

conectados al panel táctil y se usarán como canales del ADC, la configuración que se llevó a cabo siguió estos pasos:

- Configurar ambos pines como entradas a través del registro *TRISA* y estas como analógicas a través de *ANSELA*.
- Seleccionar el reloj del conversor, tiempo de adquisición y la justificación elegida para mostrar el resultado a través de *ADCON2*.
- Elección de los voltajes de referencia en *ADCON1*.
- Seleccionar un canal (solo uno en cada lectura), activar el conversor y activar bit GO/DONE (GO/DONE = 1) en el registro *ADCON0*.

El bit GO/DONE del registro ADCON0 indica si empezar la lectura o si ya está terminada la misma. Se puede añadir una interrupción para indicar que se terminó la lectura del valor, pero se optó por un bucle que comprobara cíclicamente el valor del bit GO/DONE.

2.2.6. Temporizadores

El PIC18 cuenta con 4 temporizadores/contadores (TMR0/1/3/5) que pueden trabajar de 8 o 16 bits y 3 temporizadores (TMR2/4/6) de 8 bits.

Todos pueden generar una interrupción con testigo o bandera de desborde del registro que lleva la cuenta. En el programa que se desarrolló, únicamente se recurrió a los TMR0 y TMR1, estos se usaron para generar una interrupción, trabajando como temporizadores.

2.2.6.1. Timer0

Se trata de un Contador/Temporizador configurable a 8 o 16 bits, cuenta con un prescaler de 8 bits, se puede seleccionar una fuente de reloj interna o externa y es posible seleccionar el flanco de la fuente de reloj externa.

Para configurar el funcionamiento del mismo se ha de recurrir al registro *T0CON*, los registros *TMR0H* y *TMR0L* son los encargados de llevar la cuenta, en caso de trabajar con 8 bits solo se realizará la cuenta en el *TMR0L*.

- *T0CON*: Este registro cuenta con una serie de bits que controlan el funcionamiento del TIMER, se puede elegir si aplicarle o no el prescaler, que valor de prescaler, seleccionar la fuente del reloj, si realizar el incremento de la cuenta en cada pulso de subida o de bajada del reloj, fijar su trabajo a 8 o 16 bits y encender el temporizador.

Este temporizador activará una interrupción al desbordarse, los registros necesarios se especifican más adelante "(Ver apartado 2.2.7. Interrupciones)".

2.2.6.2. TIMER1

El Timer1 es un temporizador/contador de 16 bits, tiene un prescaler de 3 bits y como en el TMR0 se puede escoger la fuente del reloj. A pesar de que se usara como temporizador, este cuenta con la particularidad de que puede ser usado para trabajar en los módulos de captura y comparación.

Se configura el funcionamiento de este temporizador a través del registro *T1CON*, en los registros *TMR1H* y *TMR1L*, se llevará la cuenta.

- *T1CON:* En este registro se configuran aspectos como la selección de la fuente de reloj, el prescaler elegido. También da la opción de elegir un oscilador secundario y sincronizar la señal del exterior o del oscilador secundario, otra opción que habilita este registro es la lectura de los 16 bits de la cuenta, por separado o como uno solo.

El uso de algunas opciones de funcionamiento dentro de este registro depende de otros registros.

2.2.7. Interrupciones

El MCU cuenta con un sistema de interrupciones, que le permite centrarse en una tarea hasta que ocurra un evento. Estas interrupciones pueden estar vinculadas a eventos externos, por ejemplo: al cambio de estado de un puerto, al final de una conversión del ADC, a un overflow de los Timers o a eventos en otros periféricos en general.

Se pueden configurar como de alta o baja prioridad y dependiendo de ello, se asignan a uno de los dos vectores de interrupción, que se disponen en la memoria del programa.

Para el programa desarrollado en este proyecto, se han establecido dos interrupciones vinculadas a los temporizadores explicados anteriormente (TMR0/1), a pesar de existir la opción, no se optó por elegir una prioridad entre ellas, ambas quedan emplazadas en el mismo vector.

Para activar una interrupción es necesario activar su uso indicándolo en un registro de control, luego es necesario indicar que componente o periférico la puede producir, por último, una vez se produzca la interrupción, borrar el testigo o "bandera" que indica que se ha producido esta, en caso de querer que se repita.

Existen 19 registros que controlan las interrupciones del programa. El registro que activa el uso de las interrupciones es el *INTCON*.

- *INTCON:* Permite activar las interrupciones y discernir si solo la de los periféricos o todas. Además, permite indicar que esta las puede producir el TMR0, las interrupciones externas o un cambio de estado del puerto B, así como de los testigos que indican que la interrupción por estos eventos ha ocurrido (se ha de borrar por software).

La interrupción del TMR0 se puede controlar solo haciendo uso de este registro, pero para el TMR1 hay que atender a dos registros más, el *PIE* y el *PIR***.**

- *PIE1:* Aquí se activa la posibilidad de que el TMR1 active una interrupción, así como el ADC, la EUSART1, el MSSP, …
- *PIR1:* En este registro se encuentra el testigo, que indica que la interrupción a causa del TMR1 se ha producido y a ha de ser borrada a través del software una vez se produzca para que pueda volver a producirse.

En total la parte que corresponde a las interrupciones corresponde a los registros INTCON (Interrupt Control), PIR (Peripheral Interrupt Request), PIE (Peripheral Interrupt Enable) y IPR (Periprheral Interrupt Priority). Siendo este último, el que permitirá darle una prioridad u otra a cada interrupción.

2.3. GLCD con panel táctil

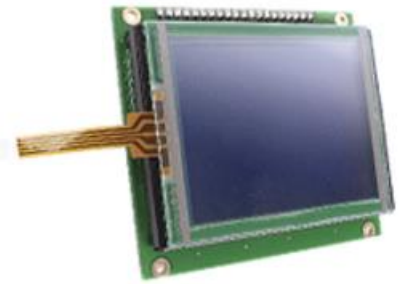


Figura 2.9: GLCD con panel táctil

La pantalla gráfica de cristal líquido (GLCD), viene con una un panel táctil resistivo de 4 hilos, situado en el frontal de la misma. Se incluirá su descripción con la de los componentes separados, puesto que, a pesar de estar en el mismo dispositivo, no existe una comunicación directa entre ellos.

2.3.1. Pantalla gráfica de cristal líquido (GLCD)

La pantalla es una GLCD con una resolución128x64 pixels monocroma, que funciona a través de dos controladores modelo NT708C o similares. Esta viene emplazada en un soporte, con las conexiones necesarias para encajarla directamente en su posición reservada en la placa de desarrollo.

Al tener las conexiones ya preestablecidas, facilita por una parte la instalación en la placa, pero condiciona la labor del programador, que se tiene que adaptar a usar unos puertos concretos del MCU.

Los puertos usados son el puerto D (transmitir datos) y el puerto B (control de la pantalla). El resto de conexiones, corresponden a las de alimentación y brillo.

En el datasheet viene indicada la función que controla cada puerto del GLCD. También viene una serie de instrucciones de control "(Ver Anexo I: Datasheet WDG0151)".

La información de la pantalla, se muestra a través de los dos controladores en los que se divide el funcionamiento de la misma, cada controlador tiene un espacio de memoria RAM de 512 bytes, cada bit de memoria RAM está conectado con un punto o pixel de la pantalla. Por lo

tanto, cada controlador llevará la información de los 4096 pixels, de los que está compuesto cada lado de la pantalla, haciendo un total de los 8192 (128x64) pixels.

Comentar que para enviar información por parte del MCU al GLCD, es necesario establecer a través del puerto B (RB0, RB1), para cuál de los dos controladores va destinada.

La memoria dentro de cada controlador está dividida de la siguiente manera, los 512 bytes están divididos en 8 páginas, siendo cada una de 64 bytes, para modificar el valor de un pixel de la pantalla es necesario posicionarse en su página correspondiente, seleccionar un byte de los 64 que tiene y modificar el byte entero en el que está contenido.

En la información del GLCD, la posición de las páginas corresponde al eje X y la de los bytes al Y. Comentar que durante del desarrollo del software se hacen referencia a ellos de manera inversa, tomando el eje Y como el de las páginas y el eje X como el de los bytes.

Esta decisión se tomó, debido a que es la que se usa como referencia en la obtención de valores de la pantalla táctil y resulta más natural referirse a los ejes siguiendo esta nomenclatura, a la hora de dar nombres a los registros y las rutinas.

2.3.1. Panel táctil

El panel táctil que viene incorporado sobre la GLCD, es un panel resistivo de 4 hilos y suele estar constituido por dos láminas de óxido de indio estaño (ITO) separadas entre ellas, en el momento en el que se produce la pulsación estas dos capas hacen contacto y se determina el punto pulsado a través de los circuitos control.

Para obtener el punto de la pantalla que se pulsa es necesario obtener las coordenadas de X e Y por separado, el funcionamiento se asemeja al de un divisor de tensión.

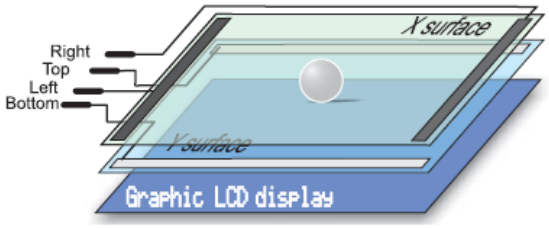


Figura 2.10: Composición del panel táctil (Manual del fabricante)

La coordenada X se obtiene aplicando tensión a la capa de X, esta cuenta con dos conectores, el izquierdo se conectar a tierra y el derecho a la alimentación, luego es necesario que en la capa Y se ponga el contacto inferior conectado al ADC, cuando ambas capas entren en contacto al ser presionadas, el convertidor recibirá un valor del que obtendrá la coordenada X.

Para la coordenada Y, se suministrará la tensión a la capa Y en el conector superior, dejando el inferior a tierra y será el conector izquierdo de la capa X quien realice la medida.

Los circuitos necesarios para el control de la lectura ya están implementados en la placa de desarrollo, por lo tanto para el uso del panel táctil solo es necesario situar los switches de la posición Sw3 en su configuración correcta.

Capítulo 3. Herramientas de desarrollo

Capítulo 3. Herramientas de desarrollo

En este capítulo se describen las herramientas usadas para poder desarrollar el código en ensamblador y programar el MCU, también se explicará las configuraciones necesarias en dichas herramientas de programación para este proyecto, así como el uso de la simulación para poder probar el programa antes de cargarlo en el MCU.

3.1. MPLAB X IDE

Se trata del compilador para PIC que suministra la empresa Microchip Technology, se suministra de manera gratuita en su página. Se escoge este compilador por ser el de la empresa que fabrica el MCU, ya que viene con facilidades a la hora de programar sus dispositivos, además cuenta con la posibilidad de realizar el código en ensamblador, lo cual es la base de este proyecto.

En la placa EASY PIC V7 se suministraron otros compiladores, pero estos solo permiten la programación en C, BASIC o PASCAL.

A continuación, se explican los pasos que se llevan a cabo para la creación de un código en ASM, probarlo en el simulador y compilarlo, además se muestra la aplicación usada para crear las tablas con los mapas de bits.

Lo primero que se hizo fue descargar el programa con la última versión de la página de microchip.

3.1.1. Creación de un nuevo proyecto

Una vez instalado el programa MPLAB X se procede a ejecutarlo. Para empezar un proyecto elegimos la opción de *new proyect*, situada en la parte superior izquierda de la ventana del programa.

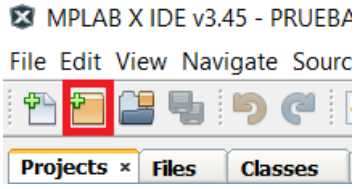


Figura 3.1: Situación opción *new proyect*

Se nos abrirá una consola que nos permitirá ir seleccionando las características que presentará este nuevo proyecto. La primera ventana que se abre nos permite elegir entre una serie de opciones, en la casilla de Categories se escoge la opción de Microchip embebed y en la de Projects seleccionamos la de Standalone Project, la cual nos permite empezar un proyecto desde cero.

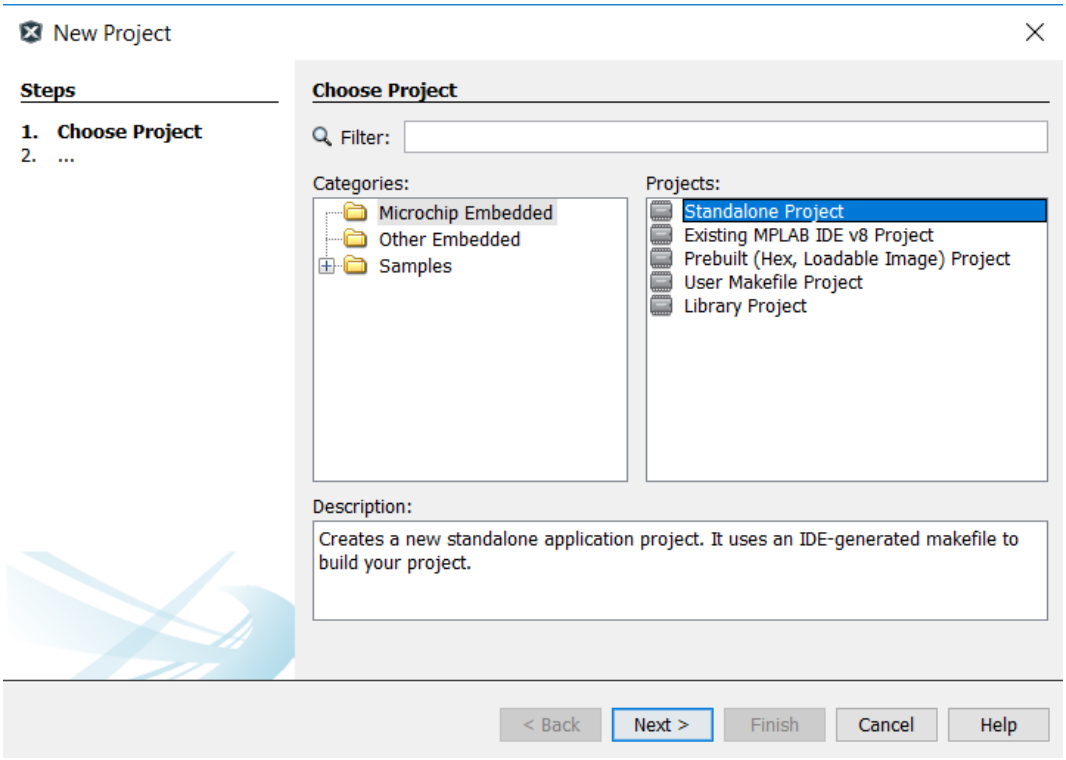


Figura 3.2: Ventana *New Project*

La siguiente ventana nos pide que seleccionemos a que familia pertenece el MCU y el modelo de PIC, lo hacemos y seleccionamos Next.

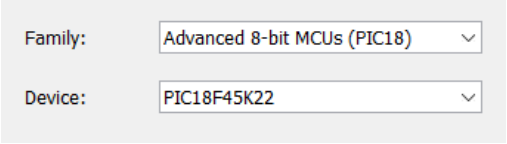


Figura 3.3: Ventana de selección del dispositivo

Después, el siguiente paso es la elección de la herramienta de hardware, dependiendo del modelo de MCU se permite usar una serie de grabadores o el simulador. Se escogió este último puesto que nos permite probar el código en el MPLAB X antes de cargarlo en el MCU.

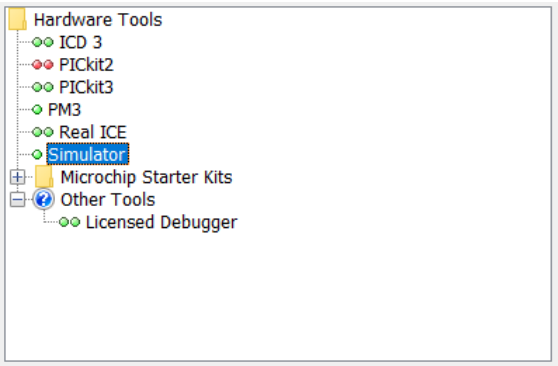


Figura 3.4: Elección herramienta de Hardware

Luego se seleccionó el compilador, como este proyecto requiere ser desarrollado en ensamblador se elige la opción de mpasm (Microchip assembler).

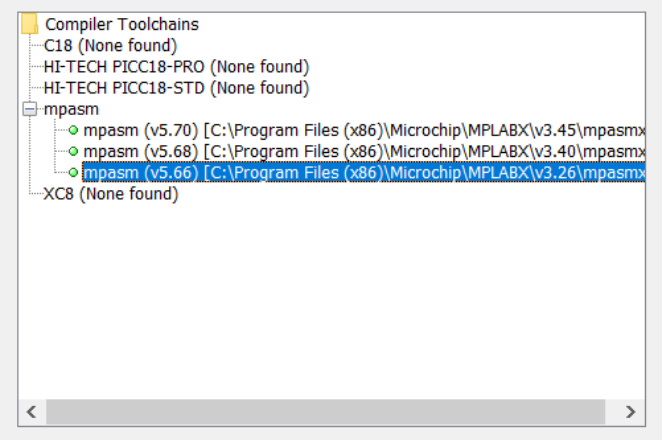


Figura 3.5: Ventana de elección del compilador

Por último, hay que indicar el nombre de la carpeta donde se guardarán los archivos de este proyecto. Una vez se le dé a *Finish* se creará la carpeta del proyecto.

Para poder empezar escribir el código es necesario crear un archivo que contenga el código desarrollado, hay dos maneras de hacerlo, una puede ser situándose en la carpeta del proyecto que hemos creado, seleccionar en la barra de menú la opción *File* y después *New File*, se abrirá una ventana donde indicaremos que será un archivo en ensamblador y que contendrá la extensión *.asm*, al darle a *Next* indicaremos el nombre que se le quiere dar a este archivo.

La otra opción para hacerlo, consiste en seleccionar dentro de la carpeta del proyecto el fichero *Source Files* y con el botón derecho, darle al desplegable *New* y seleccionar *assemblyFile.asm*, siendo únicamente necesario indicar el nombre.

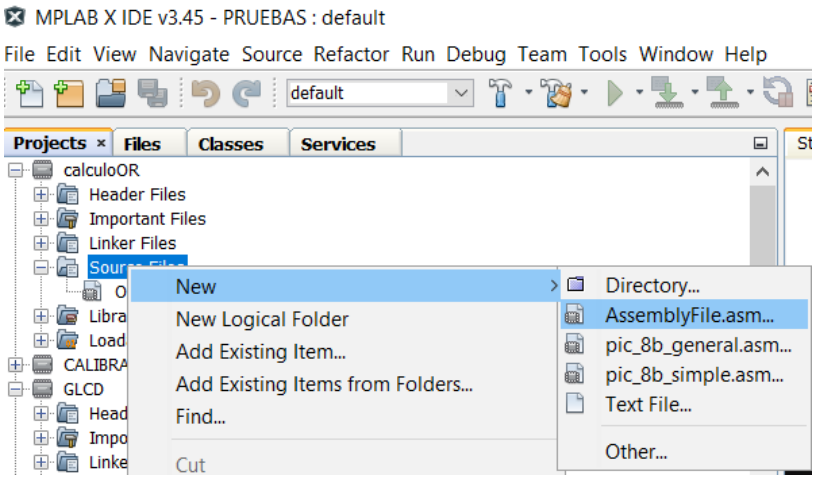


Figura 3.6: Crear archivo .asm

En este archivo irá el código del programa, aquí se irá desarrollando el código con las instrucciones en ensamblador necesarias, una vez terminado con la opción en la barra del menú, *Clean and Build Main Project* compilaremos el código.

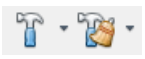


Figura 3.7: Menú para compilar el código

En caso de que la compilación sea exitosa o hubiera ocurrido un error se indicaría a través de la ventana en la parte inferior de la pantalla llamada *output*, en caso de ser exitosa se podrá ejecutar la simulación o cargar el código en el MCU.

3.1.2. Simulación

El MPLAB X cuenta con un simulador que permite probar el código compilado como si estuviera en su ciclo de ejecución dentro del MCU.

Da opciones como, el cargar valores analógicos o digitales en los puertos, permite modificar los registros durante la ejecución y observar su contenido.

También se puede realizar la ejecución del código paso a paso, hasta un breakpoint o de manera continua, estas opciones de control de ejecución se llevan a cabo a través de las opciones presentes en la barra superior de la ventana del programa.

Figura 3.8: Control debugger

Para poder observar los valores de los registros generales que hemos usado, es necesario realizar una pequeña modificación dentro del fichero del proyecto, es necesario configurar el proyecto para que trabaje en modo absoluto, en caso contrario no será posible examinar el valor de los registros durante la ejecución del debugger.

Esto se hace presionando con el botón derecho sobre el nombre del proyecto, situándose en el desplegable *Configuration Set* y seleccionando la opción de *Customize*, se abrirá una ventana en la cual deberemos seleccionar en el desplegable *mpasm (Global Options)*, una vez dentro de este apartado deberemos seleccionar la opción *Build in absolute mode.*

Con esto podremos ver los registros generales y modificar su contenido para realizar la simulación, solo habrá que realizar el proceso de Debug seleccionando la opción de *debug main Project.*

Figura 3.9: Debug Main Project

Los registros se mostrarán en la ventana Variables y habrá que ir incluyendo las que queramos que se muestren durante la ejecución de la simulación.

Otra ventana importante es la de Stimulus, la cual nos permite simular la entrada de un valor por uno de los puertos del PIC, hay que incluir en esta ventana el pin que queramos excitar e indicar con qué valor queremos hacerlo.

Estas serían las principales consideraciones a la hora de usar la opción del simulador del MPLAB X.

3.2. mikroProg suite for PIC

Se trata del programador on-board con el que cuenta la placa EASY PIC V7, el MPLAB X puede usar los programadores de Microchip para el MCU, pero para ello se necesita cambiar la configuración sustituyendo el simulador por uno de los programadores con los que cuenta.

Se optó por usar el de MikroElektronika puesto que es el que se encuentra en la placa y no requiere buscar uno compatible en el MPLAB X. para ello hay que instalar primero los drivers que vienen en el CD de la caja, para seguido hacer lo propio con el programa mikroProg suite. Este será encargado de cargar el archivo .Hex generado por el compilador del MPLAB X.

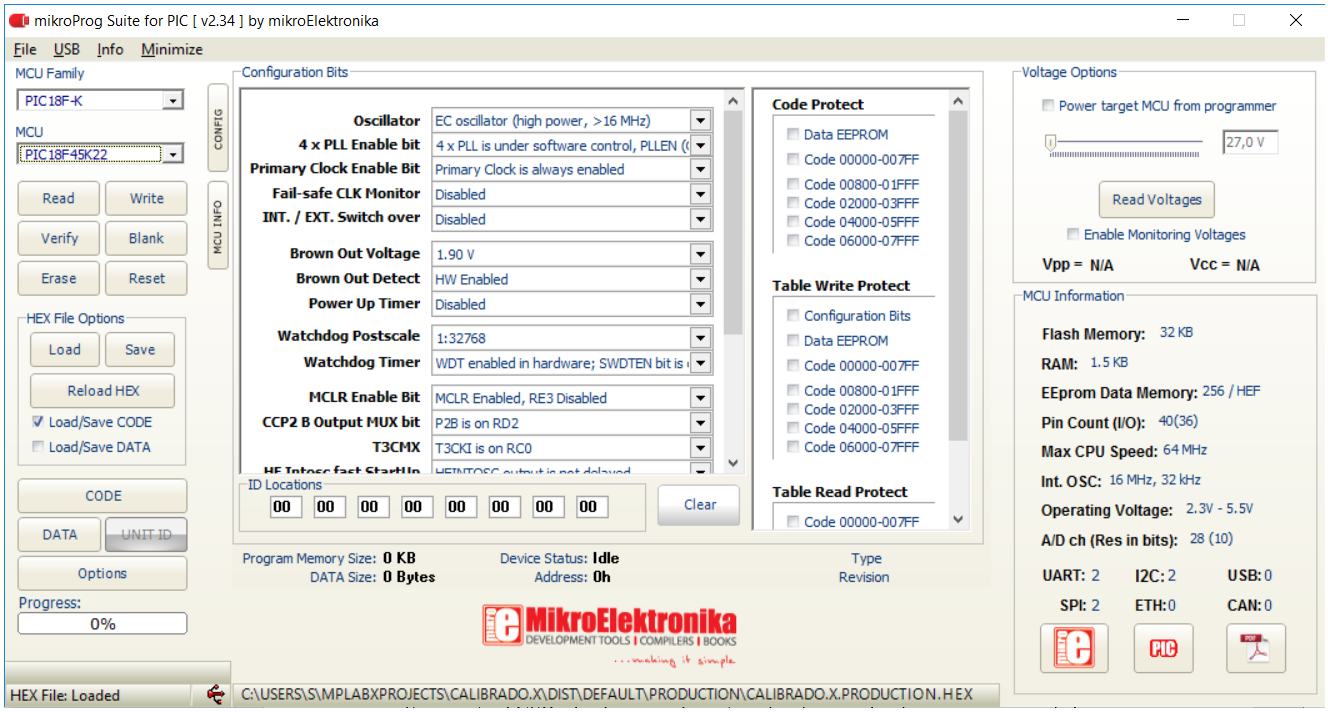


Figura 3.10: mikroProg suite

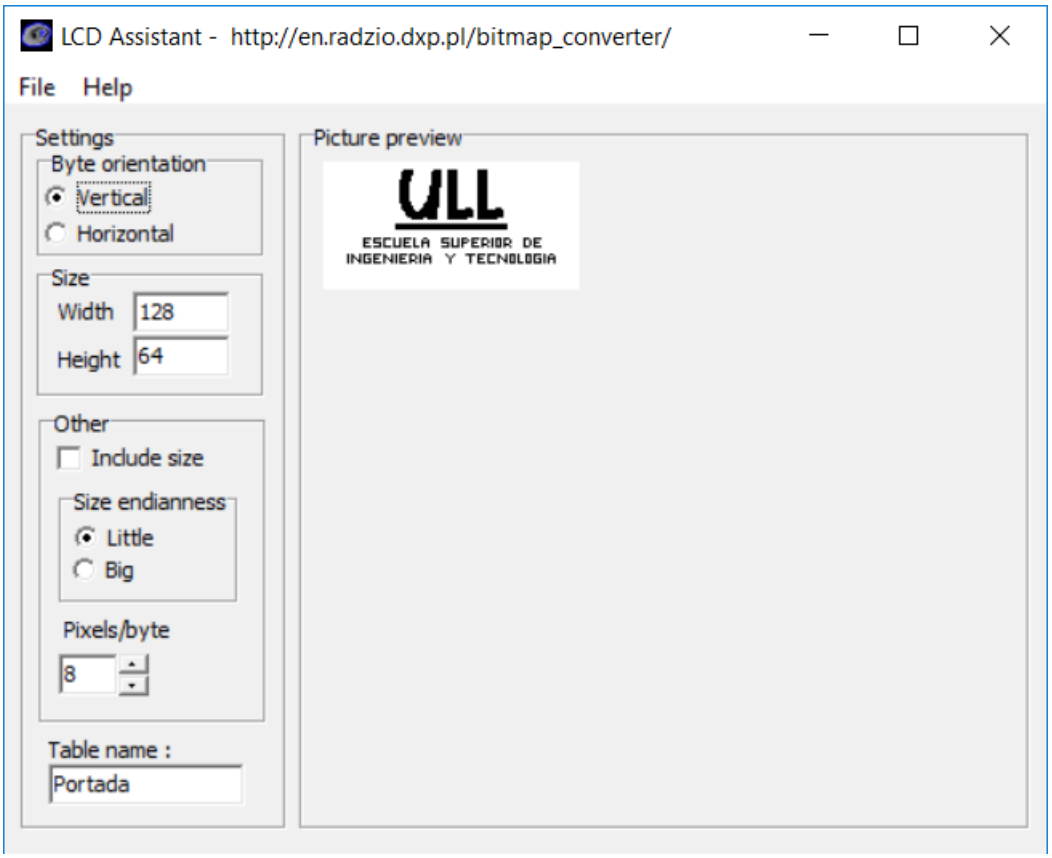
Se conecta el USB a la placa por el puerto indicado para el mikroProg, se enciende la placa y se espera a que el programa reconozca que esta está encendida, luego utilizando el botón *Read* se carga el archivo .Hex, una vez se cargue, al darle a *Load* se guarda en el MCU para que se ejecute.

3.3. LCD assistant

Esta aplicación se usó para la creación de tablas en ensamblador, en ella se cargan imágenes.bmp y devuelve un archivo de texto con todos los valores de los pixels en hexadecimal, dispuestos para programar en C directamente.

El archivo.bmp se creó con el programa Paint, hay que confiurar este programa para que el tamaño de la imagen sea el mismo que el de la pantalla. Esto se hace en archivo>propiedades cambiando el tamaño de la hoja de dibujo por 128x64 pixels, luego se dibuja la imagen que se guardará en la tabla como “mapa de bits monocromático”.

Se abre la aplicación LCD assistan y en el menú *File* de la parte superior izquierda se selecciona la opción *Load image*, se carga la imagen.bmp creada y se elige el alto/ancho, la orientación de los bytes, se vuelve a abrir el menú *File* y ahora se presiona la opción de *Save* output, se indica el nombre con .txt al final para que cree un archivo de texto.



3.11: LCD assistant

Por último, se abre el documento de texto creado y se configura el documento como se menciona en el Capítulo de Desarrollo de software para poder aplicarlo al programa creado en lenguaje ensamblador.

Capítulo 4. Desarrollo del software

Capítulo 4. Desarrollo del software

En este capítulo se explicará el proceso llevado a cabo para la obtención del código en ASM necesario para que el MCU pueda hacer funcionar la GLCD junto con el panel táctil y crear una pequeña aplicación. Se ha ido separando el proceso en pequeños programas, que según se iba avanzando en la labor, pasaban a formar parte del programa final.

4.1. Primeros pasos

La primera tarea que se llevó a cabo fue la de familiarizarse con la placa de desarrollo EASY PIC V7, esta placa cuenta con una serie de switches y jumpers que requieren una configuración particular dependiendo del uso que se le vaya a dar "(ver Capítulo 2. apartado 2.1. Easy PIC V7)".

La placa viene con dos zócalos (X1, X2) en los que se puede colocar un oscilador de cristal, cuya frecuencia se usará para obtener la señal de reloj del MCU. En este caso la placa viene provista de un cristal de 8 Mhz, este dato es importante ya que aparte de ser necesaria una configuración en particular de los jumpers, es necesario especificar el tipo de oscilador y su frecuencia en los bits de configuración.

Primero se probó un código en C, el cual consistió en un programa ejemplo que venía provisto en los drivers de la placa, fue necesario prestar atención a la posición sw3. Esta cuenta con los switches necesarios para conectar el MCU con los LEDs situados en cada uno de los puertos de la placa, siendo el sw3.1 el del puertoA/E, el sw3.2 puertoB y así sucesivamente hasta el PuertoD.

La placa del fabricante MikroElektronika se suministra con compiladores en tres lenguajes, los cuales son Basic , C y Pascal. El código escogido viene en C preparado para funcionar en esta placa, únicamente es necesario indicar en el compilador el modelo de MCU y su velocidad de reloj. El programa ejemplo que se probó e iba encendiendo, cada cierto tiempo, los leds de uno de los puertos que se encuentran en la placa.

Una vez comprobado el correcto funcionamiento de la placa, pues podía ser que hubiera algún defecto, lo cual, a la hora de programar en ASM podría ser difícil de comprobar, se pasó a la siguiente etapa en el desarrollo del código.

4.2. Bits de configuración

En este apartado se comenta el proceso previo al desarrollo del código. La programación en ASM requiere de una serie de directivas que se sitúan en el encabezamiento del código y que, durante la ejecución del programa principal, no se podrán cambiar.

Primero se añadió la directiva #include <P18F45K22.inc>, la cual nos declara la dirección asociada a cada registro en el mapa de memoria del MCU.

El siguiente paso consistió en la correcta programación de los bits de configuración. Estos bits fijan el funcionamiento de ciertos componentes especiales del MCU, en el datasheet del mismo viene una tabla en donde se indica el valor de cada uno de estos bits en caso de no ser especificado su valor en el código “(ver tabla 4.1)”.

TABLE 24-1: CONFIGURATION BITS AND DEVICE IDs

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Default/ Unprogrammed Value
300000h	CONFIG1L	—	—	—	—	—	—	—	—	0000 0000
300001h	CONFIG1H	IESO	FCMEN	PRICLKEN	PLLCFG	FOSC<3:0>				0010 0101
300002h	CONFIG2L	—	—	—	BORV<1:0>		BOREN<1:0>		PWRTEN	0001 1111
300003h	CONFIG2H	—	—	WDPS<3:0>				WDTEN<1:0>		0011 1111
300004h	CONFIG3L	—	—	—	—	—	—	—	—	0000 0000
300005h	CONFIG3H	MCLRE	—	P2BMX	T3CMX	HFOFST	CCP3MX	PBADEN	CCP2MX	1011 1111
300006h	CONFIG4L	DEBUG	XINST	—	—	—	LVP(1)	—	STRVEN	1000 0101
300007h	CONFIG4H	—	—	—	—	—	—	—	—	1111 1111
300008h	CONFIG5L	—	—	—	—	CP3(2)	CP2(2)	CP1	CP0	0000 1111
300009h	CONFIG5H	CPD	CPB	—	—	—	—	—	—	1100 0000
30000Ah	CONFIG6L	—	—	—	—	WRT3(2)	WRT2(2)	WRT1	WRT0	0000 1111
30000Bh	CONFIG6H	WRTD	WRTB	WRTC(3)	—	—	—	—	—	1110 0000
30000Ch	CONFIG7L	—	—	—	—	EBTR3(2)	EBTR2(2)	EBTR1	EBTR0	0000 1111
30000Dh	CONFIG7H	—	EBTRB	—	—	—	—	—	—	0100 0000
3FFFFEh	DEVID1(4)	DEV<2:0>			REV<4:0>					qqqq qqqq
3FFFFFh	DEVID2(4)	DEV<10:3>								0101 qqqq

Legend: – = unimplemented, q = value depends on condition. Shaded bits are unimplemented, read as '0'.

Note 1: Can only be changed when in high voltage programming mode.
2: Available on PIC18(L)FX5K22 and PIC18(L)FX6K22 devices only.
3: In user mode, this bit is read-only and cannot be self-programmed.
4: See Register 24-12 and Register 24-13 for DEVID values. DEVID registers are read-only and cannot be programmed by the user.

Tabla 4.1. Bits de configuración (Datasheet)

Normalmente se desaconseja utilizar estos valores sin tener en cuenta si pueden crear incompatibilidades con el hardware de nuestro dispositivo.

Aunque a grandes rasgos los bits de configuración suelen tener partes similares en la mayoría de los MCU, pueden tener algunas características particulares, así como, unas posiciones concretas para cada uno de ellos, por lo tanto, era necesario conseguir una lista con los correspondientes para el PIC18F45K22.

En el archivo p18f45k22.inc antes mencionado en la directiva, podemos encontrar como define cada uno de esos bits y así poder incluirlos en nuestro programa añadiendo delante de ellos la directiva CONFIG, no obstante, el compilador MPLAB de Microchip, cuenta con una herramienta que permite configurar dichos bits con mayor facilidad y que posteriormente podremos usar añadiéndolo directamente en el código.

Utilizando dicha herramienta de ayuda se prestó especial atención a los siguientes registros de los Bits de configuración.

1. *FOSC:* La placa de desarrollo EASYPICV7, como se especificó en el apartado anterior, cuenta con un oscilador de cuarzo externo de 8 Mhz, por lo tanto, con la ayuda del datasheet, se configuro el bit para que funcione con cristal de cuarzo de alta velocidad y media potencia, indicándolo en el registro con las siglas HSMP (high speed-medium power).

   De no hacerlo, me hubiera dejado la configuración estándar, la cual quedará pre-configurada para el uso de un reloj externo al sistema, lo cual provocaría que con la configuración de jumpers colocados en la placa, el código cargado en el MCU no funcione, sea cual sea.
2. *WDTEN:* También, se comprobó que el WDTEN (“Watchdog Timer Enable”) quedara desactivado. Como esta función reinicia el sistema en caso de quedarse bloqueado, se suele aconsejar que durante el proceso de desarrollo del código se desactive, para así evitar que interfiera en el proceso de búsqueda de errores. No se consideró necesario habilitarlo una vez finalizado el desarrollo del código.
3. *BOREN, BORV, PWRTEN:* Como la placa viene con una fuente de alimentación propia, también se desactivo esta medida de protección, puesto que ya estaba suficientemente protegido. No obstante, no se recomienda habilitar esta función si no es necesaria, sobre todo para principiantes.

4. *MCLRE:* La placa cuenta con un interruptor en la parte superior derecha, que está directamente conectado con el pin MCLRE del MCU, al no necesitar el pin del puerto asociado a esta función. Se habilitó para poder reiniciar la ejecución del código manualmente, facilitando el proceso de búsqueda de errores en la ejecución del programa.
5. *CP0/1/2/3:* Por último, se modificó los bits de protección de memoria del programa, puesto que podía darse el caso de necesitar leer el código cargado en el MCU, además de no ser una función necesaria para el uso final del programa.

La mayoría de estos bits de configuración no afectan al funcionamiento final del código. Por tanto, si así se considerara necesario, más adelante se podría modificar la configuración realizada en el código de este proyecto, puesto que su configuración final corresponde a criterios de comodidad en el proceso de diseño, en caso de hacerlo habría que prestar especial atención al registro FOSC, puesto que para la modificación de este bit es necesario que se haga de acuerdo con la configuración de los jumpers de la placa.

Una vez finalizado este proceso, obtendríamos una plantilla, la cual podemos usar para empezar el código del programa, despreocupándonos del proceso de configuración previo para el funcionamiento del MCU. En el Anexo II se encuentra la configuración usada.

El siguiente paso, consistió en comenzar a desarrollar el código necesario para el funcionamiento del programa que controle la GLCD.

4.3. Comunicación con la GLCD

El primer código en ASM se centró en establecer las primeras instrucciones de control para la comunicación entre el MCU y la GLCD. En la placa, ya vienen los puertos conectados entre ambos dispositivos, así que solo será necesario colocar en posición correcta los switches situados en sw4.

El puertoD del MCU está conectado al BUS de datos del GLCD, mientras que el puertoB estará conectado a las líneas de control del GLCD, el resto de pines que llegan al GLCD no tienen mayor importancia a la hora de programar, puesto que son los pines de Vcc, ground y contraste de la pantalla, los cuales funcionan de manera independiente al MCU.

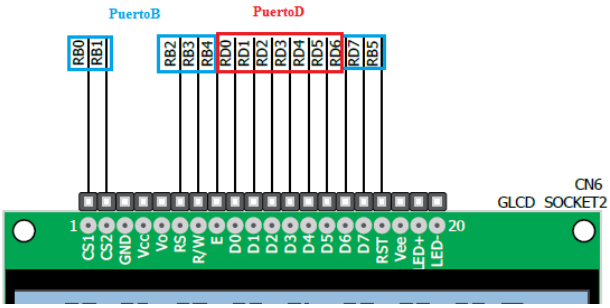


Figura 4.1: Esquema de conexiones con el GLCD (Manual del fabricante)

Atendiendo al funcionamiento descrito del GLCD, cada bit de su memoria RAM está conectada a un pixel de la pantalla, estos muestran encendiéndose (1) o apagándose (0) el valor guardado en el dicho bit.

Para modificar los bits de la RAM hace falta usar una serie de instrucciones de control, las cuales consisten en posicionarse en una página (Y) concreta, luego indicar la coordenada X en la que se encuentra el bit a modificar, y por último enviar el valor que quieres darle a ese bit "(ver apartado Capítulo 2. apartado 2.3.1 Pantalla gráfica de cristal líquido)".

Como se comentó, la información a la GLCD será transmitida a través de los Puertos B y D, así que el primer paso consistió en configurar estos puertos como salidas digitales.

Se desarrolló unas sencillas subrutinas siguiendo el datasheet del GLCD, para realizar las instrucciones de control necesarias para encender la pantalla y modificar los valores de la RAM, a las que el programa fuera llamando según se fuera ejecutando, en este primer código, únicamente implemente las necesarias para activar el controlador de cada lado de la pantalla y encenderla mostrando los valores guardados en la RAM.

A la subrutina de datos se la denomina S_WRITE, esta será la encargada de modificar el byte de RAM seleccionado, a la de instrucciones de control se la denomina S_INSTRUCTION, enviará la posición de X, la página, además de encender y apagar la pantalla en caso necesario y a las encargadas de encender seleccionar uno de los dos controladores en los que está dividido el GLCD se les llamo S_DER y S_IZQ.

```
S_IZQ                   ; Subrutina de selección del controlador de la izquierda de la pantalla
        BCF LATB,0      ; CS1 = 0
        BSF LATB,1      ; CS2 = 1
        RETURN

S_DER                   ; Subrutina de selección del controlador de la derecha de la pantalla
        BSF LATB,0      ; CS1 = 1
        BCF LATB,1      ; CS2 = 0
        RETURN

S_INSTRUCTION           ; Subrutina que envía la información del puertoD al controlador elegido
                        ; APAGADO/ENCENDIDO...ESPECIFICAR: PAGINA, COORDENADA X
        CLRF TRISD      ; Todo el puerto D como SALIDA
        BCF LATB,2      ; D/I = 0
        BCF LATB,3      ; R/W = 0
        BSF LATB,4      ; E = 1
        CALL Delay_10us
        BCF LATB,4      ; E = 0
        CALL Delay_10us
        RETURN

S_WRITE                 ; Subrutina que carga el puertoD en el Byte de RAM elegido
        CLRF TRISD      ; Todo el puerto D como SALIDA
        BSF LATB,2      ; D/I = 1
        BCF LATB,3      ; R/W = 0
        BSF LATB,4      ; E = 1
        CALL Delay_10us
        BCF LATB,4      ; E = 0
        CALL Delay_10us
        RETURN
```

Figura 4.2: Código correspondiente a las instrucciones de control del GLCD

Al probar este código surgió el primer problema, se debido a que el bit correspondiente con la posición RB05, correspondiente con el pin de RSTB de la pantalla, si este pin se encuentra a 0 estará activo la función de Reset y no admitirá ninguna instrucción en la pantalla. Por tanto, es necesario ponerlo a 1 para poder empezar a enviar ordenes como el encendido o los datos al GLCD.

También, otro problema fue debido a la velocidad de ejecución de las instrucciones de control, para que el GLCD recibiera la orden, aparte de tener el bit RB5 a 1, es necesario que una vez fijado el valor de los bits para dicha orden, hay que activar un último bit de Enable, dejando un margen de tiempo antes de desactivarlo, este bit corresponde con el RB4 del MCU. En el datasheet del GLCD se especifica que el tiempo mínimo necesario para el ciclo de Enable ha de ser de 1us, en nuestro caso debido a la velocidad de reloj del MCU cada instrucción se ejecuta en 0,5us por tanto era necesario dejar un tiempo de retardo antes de apagar el Enable. Con una sencilla subrutina, a la que se llamó Delay_10us, se añadió un retardo, este en un

principio se dejó en un valor muy alto para finalmente, más adelante fijarlo en unos 10us, dando tiempo de sobra para que la instrucción enviada fuera leída correctamente.

No obstante, aún quedaba otro problema sin resolver, puesto que aplicando las ordenes antes comentadas se encienden todas las posiciones de la pantalla. Esto se podía deber a que después del encendido las posiciones de RAM muestran que funcionan correctamente, no se encontró referencias a esto en el datasheet de la misma, así que se continuo con el desarrollo del programa.

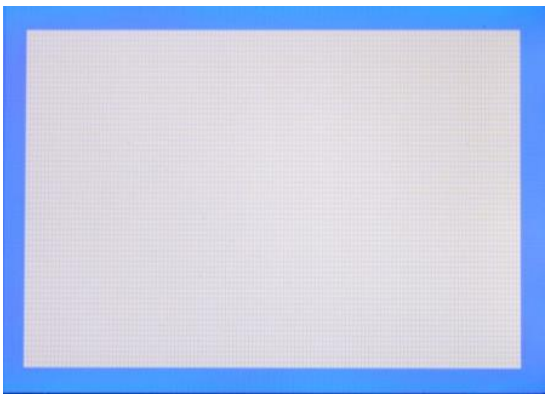


Figura 4.3: Resultado de la prueba

El siguiente paso en el código, consistió en modificar una de esas posiciones de RAM y que se mostrara por pantalla, el proceso consiste en elegir el lado de la pantalla usando las subrutinas S_IZQ o S_DER, seleccionar una página del controlador, luego una posición de X, ambas usando la subrutina S_INSTRUCTION. Por último, se cargará dichas posiciones en el puertoD, el cual envía el valor del byte de RAM con la subrutina S_WRITE.

Aunque la posición de RAM que se seleccionó para ser modificada lo hizo correctamente, nuevamente, se dio el caso en el que las demás posiciones de memoria quedaron encendidas.

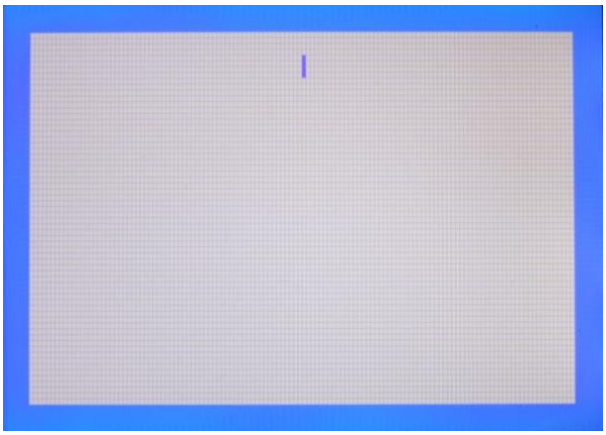


Figura 4.4: Resultado de la prueba

Una vez creadas y probadas estas subrutinas, tenía en gran parte las instrucciones de control necesarias para comunicarse con la pantalla. No obstante, en el datasheet de la pantalla habría una tercera instrucción de control para leer la información contenida en la RAM de la pantalla, se planteó en el código del programa la subrutina, pero no se probó debido a que no se consideró necesario para su uso en el programa, puesto que el programa sería el que llevara el registro de los datos que se volcaban en la RAM de la pantalla y no haría falta leer los valores en la pantalla.

La característica principal que diferenciaba estas tres instrucciones de control era el estado de los bits RS y R/W del GLCD, que correspondían con los conectados a los pines RB2 y RB3 respectivamente del MCU, dependiendo de su configuración enviaban instrucciones, datos o leían la RAM.

Habiendo logrado la comunicación con la pantalla, el siguiente objetivo fue el de lograr una rutina en el programa que permita grabar en la pantalla un mensaje.

4.4. Lectura de tablas

Para lograr imprimir un mensaje en la pantalla, el programa seguía la secuencia descrita en el apartado anterior.

Tanto para indicar la posición de X e Y, a la que se quiere apuntar para enviar la trama que fije los estados de la RAM, es necesario el uso del PuertoD. Esto hacía que después de cada subrutina hubiera que modificar los valores de dicho puerto.

Para un simple mensaje en la pantalla que constara de cuatro letras, contando con que cada letra estuviera formada por el alto de una página (1 byte) y el ancho fuera de 4 pixeles, provocaba que hubiera que modificar alrededor de unas 20 veces el valor del puertoD, lo cual no resultaba práctico hacerlo sin el uso de una subrutina, que repitiera de manera cíclica el proceso, el problema radicó en diseñar una manera de almacenar los valores necesarios para mostrar el mensaje e irlos cargando en el puerto.

La opción más clara fue la de usar una tabla que el programa fuera recorriendo y volcando los datos en la pantalla.

Primero, se probó la manera más común para guardar una lista de datos, haciendo uso de la instrucción en ensamblador “RETLW”, la cual vuelca el valor asignado a la instrucción en el registro de trabajo (WREG).

La configuración de la tabla consistirá en situar esta instrucción, con su valor asignado, una detrás de otra y utilizar el contador del programa (PC) para ir recorriéndolas.

El primer problema que surgió es debido a que el LSB del PC es 0, por lo tanto, recorre las posiciones pares, esto se debe a la configuración que tienen las direcciones de memoria del programa.

Otro problema fue que en caso de que nos pasáramos del último valor de la tabla, al estar modificando el contador del programa, este se bloquearía haciendo que se detenga la ejecución del programa.

Por último, la pantalla cuenta con unos 128 bytes en total por página y esta cuenta con 8 páginas, lo que hace que se tengan que recorrer usando el PC unas 1024 líneas de código para modificar una “pantalla” entera ocupando mucho espacio en la memora, aparte de hacer más engorroso el código.

Debido a estos problemas, se optó por otra opción para configurar la tabla de datos a mostrar por el GLCD. Usando la directiva ORG, busco un espacio en la memoria del programa que previsiblemente quede lo suficientemente alejado del programa principal, en este caso fue

la posición ORG 0x1000, seguido se indica el nombre de la tabla y empezando cada línea con la directiva DB, se indica que se trata de una tabla de datos en la cual cada elemento es de un byte, esto permite crear una tabla del tamaño que se desee, además cuenta con la particularidad de que esta aprovecha las posiciones pares e impares de la memoria del programa, a diferencia del primer método probado.

En el código principal del programa, cada vez que se quiera acceder a la tabla, habrá que cargar el registro de puntero de tabla TBLPTR el cual apunta a la dirección en la que se encuentra el primer byte, a partir de ahí utilizando la instrucción TBLRD*+, cargar el valor de la posición actual de la tabla en el registro TABLAT y aumentar posteriormente el puntero que la recorre, después de cada lectura cargamos el valor del registro TABLAT en el puertoD y lo volcamos en la RAM del GLCD.

De esta manera se consiguió mostrar un pequeño mensaje de manera más compacta, lo cual facilitaba que a la hora de añadir mensajes más complejos o diseños en concreto, de las diferentes “pantallas” a mostrar, se pudiera reutilizar este código con mayor facilidad.

Figura 4.5: Resultado de la prueba

Comentar, que debido a que todos los bits de la RAM del GLCD se ponen a 1 después del encendido, no conseguía que se mostrara el mensaje correctamente, esto se debía a que lo tenía que hacer con lógica inversa y poner a 0 los valores de las posiciones que quiero que se muestren.

El siguiente objetivo fue el de realizar las modificaciones necesarias al código principal para que recorra las posiciones de la tabla y simultáneamente lo haga en la pantalla sin que se produzca ningún desfase.

Para esto, únicamente fue necesario seleccionar la primera posición de la RAM de la GLCD, luego se hace la lectura de la primera posición de la tabla y se carga en la posición marcada de la RAM. Al hacer esto el puntero que marca la posición de la RAM se aumenta en 1 automáticamente pasando a la posición contigua, no siendo necesario volver usar la subrutina S_INSTRUCTION para situarnos en una nueva posición de RAM. Lo único que se debía tener en cuenta, era cuando se llegara a la última posición de la página, se volviera a indicar la primera posición de la página siguiente, así hasta completar todos los bytes de RAM de la pantalla.

El funcionamiento de la tabla con la indicación del puntero y la instrucción especial TBLRD*+ funciona de manera autónoma, saltando de una línea a otra según se acaben, así que lo único a tener aspecto en cuenta es que los datos estén ordenados en la tabla según se quisieran mostrar.

Por último, se realizó un código capaz de borrar el valor contenido en todas las posiciones de la GLCD, en un principio este código se ideo separado de la función que cargaba los valores de la tabla en la GLCD, más adelante se unirían ambos códigos puesto que su funcionamiento era similar, para esto se utilizó una instrucción en ASM de comparación, la cual comprueba el estado de un registro que nosotros hemos configurado y dependiendo del que presente, toma los valores de la tabla o carga 0x00 en el puertoD.

Con esta última modificación, al ir modificando el valor del registro usado en la comparación, se controlará si se quiere borrar los valores que hay en la GLCD o mostrar la información guardada en la tabla.

```
S_PANTALLA                      ; Subrutina que irá cargando los valores en la pantalla dependiendo del puntero elegido
        MOVLW .64               ; Guardamos este valor en el registro para la comparación
        CPFSLT xfile            ; 'xfile' < 64 -> PC+4...'xfile'>=64 -> PC+2
        CALL S_DER              ; Continúa con los 64 primeros pixels horizontales
;*************************************************COORDENADA X*********************************
        MOVFF xfile,LATD        ; Indicamos coordenada X de la pantalla
        BSF LATD,6              ; Configuramos el puerto para que entienda que es coordenada X
        CALL S_INSTRUCTION      ; Pasamos a la rutina para cargar una instrucción
;*************************************************************************************************
;*************************************************COORDENADA Y*********************************
        MOVf yfile,0            ; Indicamos la Página (coordenada Y)
        MOVWF LATD              ; Lo movemos al puerto D
        CALL S_INSTRUCTION      ; Pasamos a la rutina para cargar una instrucción
;*************************************************************************************************
        BTFSC orden,0           ; Si orden (0) = 0 -> PC+4 Si orden (0) =1 -> PC+2
        CALL S_TABLA            ; En caso de no ser 0 paso a mostrar la tabla elegida
        BTFSS orden,0           ; Si orden (0) = 1 -> PC+4 Si orden (0) =0 -> PC+2
        CALL S_LIMPIAR          ; En caso de no ser 1 voy a la subrutina limpiar
        CALL S_WRITE            ; Vuelco el puerto D en la GLCD
        INCF xfile              ; Aumento la posición del byte en la pág
        MOVF maximoX,W
        CPFSGT xfile            ; 'xfile' > maximoX -> PC+4...'xfile'<=maximoX -> PC+2
        GOTO S_PANTALLA         ; Bucle hasta completar todas las posiciones de X
        MOVFF xreferencia,xfile
        CALL S_IZQ              ; paso la pantalla izquierda en caso de haber completado el eje y
        INCF yfile              ; Incremento el contador de páginas
        MOVF maximoY,W
        CPFSGT yfile            ; 'yfile' > maximoY -> PC+4...'yfile'<=maximoY -> PC+2
        GOTO S_PANTALLA         ; Bucle hasta completar todas las posiciones de Y
        RETURN
```

Figura 4.6: Código final que controla la muestra del contenido de las tablas por pantalla

Al borrar los valores de la GLCD nada más encenderla, se solucionan los problemas anteriores, puesto que el mensaje ya no es necesario mostrarlo en lógica inversa.

4.5. Funcionamiento del panel táctil

En este apartado se detallará el proceso necesario que se llevó a cabo para la obtención de un código capaz de hacer funcionar la pantalla táctil incorporada en la GLCD, se trata de un panel táctil resistiva de 4 hilos “(Ver Capítulo 2. apartado 2.3.1 Panel táctil)”.

Para esto, al igual que en los casos anteriores se dividirá el código en diferentes etapas, la primera será la de añadir al programa la configuración de otros dos puertos. Estos puertos son los que llevarán la comunicación entre el MCU y la pantalla táctil, a través de los switches situados en la posición sw3 de la placa.

Los pines AN0 y AN1 del MCU, situados en el puerto A, se configurarán como entradas analógicas y los pines RC0 y RC1, del puerto C, como salidas digitales.

EL circuito de control de la pantalla táctil, con el que cuenta la placa, se controla con los pines RC0 y RC1, dependiendo de su configuración podremos realizar la lectura de la pulsación realizada en la pantalla táctil para las coordenadas X (puerto AN0) o Y (puerto AN1).

La secuencia de lectura de una coordenada u otra de la pantalla táctil, se hará alternando los valores de estos dos pines del puerto C, no dándose en ningún caso, que ambos presenten el mismo estado.

Antes de comenzar a desarrollar el código, era necesario comprobar los valores umbrales que se obtenían a través de una pulsación en la pantalla táctil, para así poder decidir la resolución necesaria para el conversor analógico-digital (ADC). Primero se trató de hacerlo contando con que estos valores umbrales partían desde 0V hasta 5V y que para mostrar todos los puntos del eje X únicamente necesitaba 128 niveles, por tanto, valdría con una resolución de 7 bits.

Antes del implementarlo en el programa principal, se realizó un código de prueba para comprobar el resultado del ADC en la coordenada X. Se puso el pin RC0 a 1 dejando a 0 el RC1, también se añadió la configuración necesaria para el ADC del pin AN0 y una pequeña subrutina que volcaba a uno de los puertos el resultado del ADC. Al partir de la premisa errónea de que los diferenciales de potencial partían de 0 a 5V en la coordenada X de la pantalla táctil, se mostraban valores no esperados en los LEDs del puerto. Para comprobar el fallo desconecté el ADC y comprobé con un multímetro los valores que llegaban al pin AN0.

Observe que los voltajes que leía el ADC partían desde los 0,32 V hasta los 4,42V aproximadamente y debido a la resolución escogida, se obtenía un valor de 9 en el menor de los casos y de 113 en el mayor, por tanto, en todo el tramo que comprendía el eje, no había los 128 saltos necesarios para que la obtención de la posición X se pudiera hacer directamente con el valor del ADC.

Cambiando la resolución del conversor se trató de buscar una configuración de este, que permitiera obtener de manera sencilla la posición de X, al no lograrlo, hizo falta operar el resultado para obtener la coordenada exacta.

Se optó por escoger una resolución de 8 bits para el eje X y realizar una interpolación para conocer la posición exacta de la pantalla táctil que se presiona.

En las instrucciones que podemos usar para ASM, este MCU cuenta, con la instrucción MULWF la cual realizará la multiplicación entre un valor almacenado en una posición de la memoria de datos y el registro W. El problema surgió para realizar la división puesto que en ASM no existe instrucción para esto. Por tanto, hubo que recurrir a ir realizando restas sucesivas del valor obtenido por el ADC.

Para esto se dividió el programa principal en etapas. En la primera se obtendría el valor de ADC, para posterior restarle un valor fijado, correspondiente con la primera posición de X que nos volcaría el conversor, la cual tenía un valor de 18. Seguido, se pasaría a la multiplicación para acabar finalizando con las restas sucesivas. Para cada etapa se fijó unas instrucciones de comparación a fin de decidir si continuábamos a la etapa siguiente o desechábamos la medida pasando a obtener una nueva.

Después de obtener un valor, este se cargaba en la pantalla. En una última etapa se comprobaba si el valor de la coordenada de X obtenida era superior a 64, para, en tal caso, conectarse con el controlador de la derecha de la pantalla, también se fijaba que el valor que se volcara solo lo hiciera en la primera página de la pantalla, puesto que aun el código no era capaz de discernir a que valores de la coordenada Y correspondía la pulsación.

Una vez hecho esto se volcó el código en el MCU y se observó, que el resultado que se mostraba en la primera página de la pantalla volcaba mucho ruido, sin que ni siquiera se realizara una pulsación, provocando que parte del eje X quedara encendido.

Además, cuando se procedía a la pulsación en la pantalla se marcaban posiciones contiguas a la seleccionada.

Para el ruido mostrado en la pantalla, previo a la pulsación, se puso el pin RC0 a 1 desde que se inicia el código, además de un bucle de espera hasta que se diera paso a la lectura de la coordenada, esto corrigió este primer problema, puesto que la causa probable del mismo fuera, que no se dejaba pasar el tiempo necesario para que se estabilizara el voltaje en el circuito de control de la pantalla.

Para tratar de solucionar las selecciones contiguas, se añadió retardos al código, considerando que estas posiciones erróneas se deben, a que toma valores cuando aún se está estabilizando el voltaje en la pantalla táctil tras la pulsación.

Figura 4.7: Resultado de la prueba

Esto redujo el número de posiciones marcadas por error un poco, pero también la precisión de la pulsación puesto que al arrastrar el lápiz por la pantalla había valores que no se captaban, haciendo que la pantalla solo captara pulsaciones esporádicas en diferentes puntos de la pantalla.

La solución a este nuevo problema podía ser, el tomar una medida cada cierto tiempo y compararla con la última obtenida, en caso de coincidir se muestra por pantalla, de lo contrario se desecha. Para esto necesitaba el uso de interrupciones, por tanto había que añadir al código la configuración para los registros del Timer 0 (TMR0) y de las interrupciones (INTCON), la cuenta del Timer solo se activaría una vez se empieza con la lectura de la pantalla táctil, la interrupción se producirá una vez se desborde este Timer, así que no se podrá realizar la comparación hasta la primera interrupción, quedando descartados los primeros valores del ADC, que previsiblemente serán los menos estables.

Esto se probó usando únicamente media pantalla para así restar complejidad al código durante las pruebas.

El primer resultado obtenido con esta solución fue similar al que nos daba los retardos, pero resultaba más fácil controlar el tiempo que dábamos entre interrupciones.

Tras múltiples pruebas variando los tiempos entre interrupciones y no obtener mejores resultados se probó a añadir otra comparación.

A causa de esto el código no servía para dibujar siluetas o letras en la pantalla, debido a que se saltaba la pulsación de valores intermedios.

Primero, guardo un nuevo valor tras cada interrupción, correspondiente al valor que se había guardado en la interrupción anterior, lo llamo “ante” y luego, en el caso de que el valor actual no corresponde al guardado en la interrupción, pasamos a la nueva comparación, en esta se resta 1 al valor obtenido, de no ser igual al valor guardado en “ante” pasamos a sumarle 1 y volvemos a compararlo, si una de estas dos comparaciones da como verdadera se mostraría el valor actual por pantalla, de no ser así, volveríamos al bucle a tomar una nueva medida del ADC.

Además, decido activar un nuevo Timer (TMR1), en la comparación donde compruebo si son pulsaciones contiguas, en este caso no realizará una interrupción sino que comprobaremos el estado de la bandera del mismo. Si esta activa volcamos los datos a la pantalla, si no volveremos a realizar la comparación con los valores contiguos.

La función de este Timer principalmente será la de dar un tiempo, para mostrar un valor por pantalla después de cada interrupción, pues por el corto espacio de tiempo que tarda el programa en ejecutarse por completo, es probable que tras la interrupción el valor que a continuación muestre el ADC sea el mismo y esto podía constituir una causa de error en la pulsación. A esta conclusión se llegó tomando la hipótesis que, si después de un largo espacio de tiempo el ADC seguía dando el mismo valor, significaría que el voltaje se ha estabilizado y que es el punto de la pantalla donde se está ejerciendo toda la presión con el lápiz.

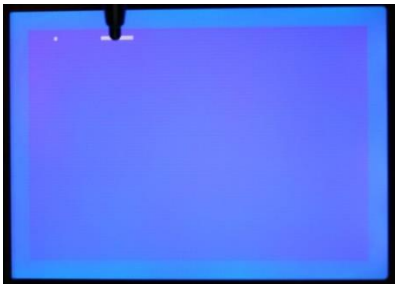


Figura 4.8: Resultado de la prueba

Con esta modificación y tras la optimización de los tiempos entre interrupciones todo lo posible, se consiguió que la obtención de valores en la coordenada X mejorara mucho, ya no se pierden valores en cada pulsación, incluso arrastrando el lápiz por la pantalla y reduciendo la cantidad de errores mostrados, aunque en las primeras posiciones sigue apareciendo algún pixel encendido por error.

Más adelante se sustituirá el uso de este nuevo Timer (TMR1), puesto que se encontró una manera mejor de realizar la función que prestaba.

Intentando eliminar esas pulsaciones erróneas en las primeras posiciones de la pantalla, se intercambió el uso de los Timer. Como para la interrupción no se requería un espacio de tiempo muy largo el TMR1 paso a activarla, y así poder usar el TMR0 para probar con mayores intervalos, dando más tiempo después de una interrupción para que se cargue un valor diferente.

Con esta última solución, se acabó con el ruido tras una pulsación en el eje X, pudiendo pasar a obtención del valor en el eje Y.

Para el eje Y se realizó el mismo proceso que para el eje X, intentando reutilizar la mayor cantidad de código posible.

Antes de modificar el código, hubo que comprobar los voltajes umbrales que daba la pantalla táctil, en el pin RC1, se volcará el valor 1 dejando a 0 el pin RC0, colocando el voltímetro a la entrada del pin AN1, vemos que este parte desde los 0,67V hasta los 4,09V. Como solo necesitamos 64 niveles de salida en el ADC podemos configurarlo con una resolución de 7 bits.

Con esta nueva resolución se hizo los cálculos para encontrar los nuevos valores de la interpolación, en este caso se dio la casualidad de que el valor correspondiente a la primera pulsación era 18, al igual que con el eje X. Por tanto, al valor volcado por el ADC se le restaría este valor sin necesidad de hacer distinción del eje en el que se está leyendo.

Luego hubo que modificar el valor que se utiliza en la rutina de restas sucesivas para realizar la división.

El principal problema que surgió fue debido al método necesario para volcar los datos en la GLCD. En el caso del eje X fijamos una página de la pantalla y únicamente era necesario indicar la posición de X, mediante una de las instrucciones de control implementadas. En el

caso de Y el problema se volvía más complejo, había que descifrar en el valor que volcaba el ADC, la página que se había pulsado y el pixel que se muestra en esa página.

El resultado de la interpolación que nos da la coordenada Y tras las comparaciones, dará un valor de 6 bits de ancho, para obtener el valor de la página y del pixel seleccionado, tengo que trocear este valor. Los 3 bits menos significativos corresponden a la posición del pixel, siendo los otros 3 restantes los de la página.

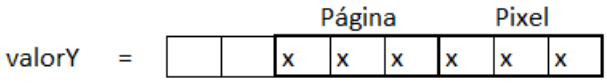


Figura 4.9: Configuración de la coordenada Y

Para separarlos y poder enviarlos en sus respectivas instrucciones a la GLCD, multiplico por 32 el resultado, desplazando los bits que indican la página a las primeras posiciones del registro PRODH, en el cual se van guardando los valores que han desbordado el registro PRODL, ambos registros son los que pertenecen a la instrucción en ASM MULWF, en los que se guarda el resultado de cualquier multiplicación, cargo el valor de PRODH en el puerto D, utilizado las instrucciones de control, indicamos al GLCD en que página hemos pulsado.

Fijamos un pixel de cada página primero, para así poder comprobar esta parte del código y que este funcionaba para seleccionar una página de la GLCD.

Debido a que el valor correspondiente a 0 en Y señala la página 7, hubo un problema puesto que mostraba la página opuesta a la seleccionada, haciendo que se tuviera que invertir el valor obtenido en PRODH antes de volcarlo en la GLCD.

A la hora de ir encendiendo los pixels de la pantalla el problema se complica, tal y como recibe el controlador de la pantalla la información con la subrutina S_INSTRUCTION, dependiendo del valor que envíe el puerto D, podemos seleccionar la posición de X y la página en Y. Seleccionadas estas dos posiciones, con la subrutina S_WRITE enviaremos los pixels que se encenderán.

La primera prueba conseguía encender correctamente ese punto del pixel seleccionado, pero al señalar uno contiguo dentro de la misma página borraba el anterior, esto se debía a que solo se enviaba el pixel escogido en la pulsación, ignorando los cargados ya en la RAM y volcando un 0 en los 7 restantes no escogidos.

Esto obligaba a tener un registro de los pixels encendidos en cada 1 byte de RAM, para sumarlos al seleccionado tras la pulsación y así volcarlos con la subrutina S_WRITE, sin modificar los valores anteriores.

Una solución al problema podía ser el ir creado una tabla en el código del programa, la cual fuera modificando sus posiciones dependiendo de las pulsaciones que vayamos realizando, sumando cada nueva pulsación a la tabla.

Se descartó esta solución, debido a que la propia GLCD contaba con una instrucción más útil para este cometido. En el datasheet del controlador del GLCD aparece la instrucción READ, la cual se implementó en las subrutinas usadas para la comunicación del MCU con la GLCD.

El proceso sería, que una vez el código se situara en la página y la posición de X, se leerían los valores guardados en ese byte de RAM, añadiéndole el valor de la pulsación actual. La prueba dio el mismo resultado que en el caso anterior, la primera pulsación se señalaba correctamente, pero al continuar realizando pulsaciones en la misma página se borraban los valores de las pulsaciones anteriores, para solo conservar la actual.

Esto en un principio parecía deberse a la manera en la que se realizaba la lectura de los datos en la memoria del GLCD, puesto que en las pruebas previas a obtener resultados con este método, el código no mostraba resultados. Se debía a un descuido en la programación, puesto que el puerto D se debe configurar como entrada para poder leer el valor del GLCD y configurarlo posteriormente como salida para enviar la información.

No obstante, el problema se debía a la manera en la que el GLCD volcaba los valores al puerto D, en el datasheet del mismo no se especifica la manera de hacer esta lectura.

Se buscó en el datasheet de los controladores del GLCD (NT7108), para ver en profundidad como es el proceso de lectura en la RAM del controlador.

En él, se especifica que es necesario la realización de una “dummy read”, lo cual se trata de una lectura ficticia, consultando códigos similares para casos parecidos, normalmente esta se hacía realizando dos ciclos de activación seguidos en la instrucción de control, sirviendo el primero para cargar los datos en el registro de salida del GLCD y enviándolos en el siguiente.

Añadiendo estas modificaciones en las instrucciones y tras múltiples pruebas se consiguió obtener los datos de la posición de RAM seleccionada.

```
S_READ                  ; Subrutina de lectura de los datos de la posición de RAM escogida

        SETF TRISD      ; Todo el puerto D como ENTRADA
        BSF LATB,2      ; D/I = 1
        BSF LATB,3      ; R/W = 1
        BSF LATB,4      ; E = 1
        CALL Delay_10us
        BCF LATB,4      ; E = 0
        CALL Delay_10us
        BSF LATB,4      ; E = 1
        CALL Delay_10us
        MOVFF PORTD, puertod
        BCF LATB,4      ; E = 0
        CALL Delay_10us
        CLRF TRISD      ; Configuro de nuevo el puertoD como SALIDA
        RETURN
```

Figura 4.10: código de la instrucción de control para leer la RAM del GLCD

Con una tabla, que se recorre a través del PC, se carga el valor del pixel pulsado al registro de trabajo, para luego sumarlo haciendo uso de la instrucción en ensamblador IORWF al valor de RAM leído, como se indica anteriormente. Finalmente lo volvemos a enviar al GLCD, evitando así que se borren las pulsaciones anteriores guardadas en la RAM.

Surgió otro problema cuando enviamos los valores finales a la GLCD, puesto que la posición de X se modifica después de realizar una lectura, por tanto había que refrescar la posición de la RAM en la cual se encontraba el pixel seleccionado.

Por último, comentar que los valores de la GLCD en el eje Y, siguen lógica inversa, la página 0 viene marcada por los valores vinarios ‘111’, así como el LSB del byte de RAM.

Una vez solucionado el último problema, se realiza la prueba fijando la posición de X, luego vamos modificando con la pulsación en la pantalla los valores de página y pixel. Como

resultado obtenemos que se marca con nitidez los pixeles seleccionados sin ruidos ni pulsaciones erróneas.

Para finalizar esta etapa del proyecto queda unir ambos códigos en uno solo, ambos utilizan las mismas funciones desarrolladas, para obtener los valores de las pulsaciones en la pantalla, pero utilizan subrutinas diferentes para enviar estos datos a la GLCD.

Además, es importante el orden en que se envíen los valores al GLCD, como se indicó antes, puesto que el posicionamiento en la RAM requiere seleccionar primero una página, luego una posición de X, para finalizar con el pixel seleccionado.

También, está la dificultad añadida de que al realizar la instrucción de control READ, se modifica la posición de X saltando a la siguiente.

Estos dos problemas a los que se les busco soluciones por separado en cada código, hacían más compleja la labor de unir ambos en uno solo.

Hubo que desarrollar varias subrutinas que se llamarán según se fuera ejecutando el código, algunas como S_XOY cargando los valores de las operaciones necesarias para obtener el pixel pulsado, que variaban para la obtención de X o de Y.

```
S_XOY                   ; Subrutina que alterna la lectura de 'X' e 'Y'
        CLRF valor      ; Limpio el valor guardado tras la interrupción
        BTFSC PORTC,0   ; Si RC0 = 0 -> PC+4 Si RC1 =1 -> PC+2
        GOTO LECTY
        GOTO LECTX

LECTX:                  ; Rutina de configuración para leer el eje 'X'
        MOVLW 0x01      ; RC0 = 1 RC1/2/3..=0
        MOVWF LATC
        MOVLW .105      ; Valor necesario para la interpolación con 'X'
        MOVWF vresta
        CALL Delay_5ms  ; Retardo para evitar problemas en la lectura
        RETURN
LECTY:                  ; Rutina de configuración para leer el eje 'Y'
        MOVLW 0x02      ; RC1 = 1 RC0/RC2/RC3..=0
        MOVWF LATC
        MOVLW .88       ; Valor necesario para la interpolación con 'Y'
        MOVWF vresta
        CALL Delay_5ms  ; Retardo para evitar problemas en la lectura
        RETURN
```

Figura 4.11: Código perteneciente a la subrutina que alterna la lectura de X e Y

Sin embargo, la más complicada de ellas era la que enviaba los valores de X e Y necesarios para encender el pixel. Primero se intentó hacer una subrutina que después de la obtención de cada coordenada, utilizara la subrutina correspondiente para enviar la orden al GLCD, en caso de que el valor obtenido correspondiera a la posición de Y, enviaría la página y el pixel, de ser la coordenada de X enviaría la posición de este eje donde se debía encontrar el pixel.

Añadir, que realizando pruebas con el simulador MPLAB pude comprobar que el TMR0 no estaba funcionando correctamente, y que la desaparición del ruido de las pulsaciones se debía únicamente a las comparaciones y al TMR1. Cada vez que se mostraba el valor por pantalla, este se quedaba bloqueado en los siguientes ciclos, se creyó que se debía, a que se reiniciaban los registros donde se llevaba la cuenta cada vez que se producía el evento comentado.

Se eliminó entonces del código la parte correspondiente al TMR0 dejando únicamente sus registros pre configurados en vista de que puedan ser usados más adelante.

Al probar el código final en la placa pude observar que el código mostraba valores sin sentido a la pantalla, haciendo que esta funcionara de manera errónea. Esto se debía previsiblemente al orden que debían de llevar los datos para cargarlos en la GLCD, al hacerlo en dos tandas, una para los valores de X y otra para los valores de Y era fácil que se produjeran errores inesperados.

Se optó por añadir dos funciones a la rutina de obtención de los valores, las cuales guardarán un valor de X primero y luego un valor de Y, para luego proceder a enviarlo todo junto a la GLCD.

Con esta nueva solución se consiguió mostrar las pulsaciones en la mayoría de la pantalla, no obstante había áreas en la misma que no respondían a la pulsación con el lápiz.

Añadiendo retardos al código, intuyendo que este problema se podía deber a causa del cambio de estados en el puerto C debido a problemas al principio del desarrollo del programa, se observó que las franjas en la pantalla que no responden a la pulsación del lápiz van variando, por lo tanto mediante el método de ensayo y error, obtengo un tiempo de retado que permite obtener valores en la pulsación de cualquier parte de la pantalla, solucionando este problema.

Corregidos los problemas presentados durante el desarrollo de esta parte del código, se consiguió que la escritura en la pantalla con el lápiz fuera bastante fluida, quedaban algunas pulsaciones erróneas que esporádicamente, que se irán depurando más adelante.

El siguiente paso en el proyecto consistió en conseguir mostrar una interfaz, para que el usuario pudiera desplazarse por las diferentes funciones que permitiría el código del programa.

4.6. Interfaz del Programa

Figura 4.12: Portada de la interfaz

Esta parte se centrará en el desarrollo de la interfaz del programa, cada pantalla del menú estará guardada en una tabla independiente, gracias a las subrutinas desarrolladas en los primeros pasos de este proyecto únicamente habrá que ir diseñando las diferentes pantallas y lograr pasar de una a otra.

La labor de diseño se realizó a través de la herramienta de Windows PAINT, configurándolo para crear una imagen del ancho de la pantalla 128x64, para posteriormente guardar el diseño creado en el formato .bmp.

El último paso para obtener el mapa de bits y poder copiarlo en nuestro código consiste en crear un documento de texto con ayuda del programa LCD assistant. El código viene preparado para C, así que hay que realizar modificaciones en él.

Las modificaciones consisten en incluir la directiva DB al principio de cada fila de datos y en quitar la última coma presente en el final de cada fila.

Una vez hecho esto se crea una tabla en el código del programa y se copia el contenido del documento de texto donde esta nuestro mapa de bits. Habrá que añadir una subrutina para cada tabla, la cual posicionará el puntero al principio de ella.

A la hora de probar a cargar diferentes pantallas, surgen los primeros problemas en el programa. Los valores que se cargan en la pantalla no corresponden a sus respectivas posiciones en las tablas.

Esto se debió a que no se reinició la posición de X, ni la página después de mostrar una pantalla, así que si se realiza una modificación accidental de estos valores, puede que ya no empezara por la primera posición de la RAM del GLCD, sino por la correspondiente al último valor que se guardó en ella, el cual puede provenir de una pulsación aleatoria.

Para solucionar esto se creó unas subrutinas llamadas S_RESETX y S_RESETY, las cuales situaban el puntero de la RAM de la GLCD en la primera posición de la misma, situando estas subrutinas antes de la encargada de cargar el valor de la tabla se evitaba el problema.

```
S_RESETX                ; Subrutina donde se reinician los valores de referencia de la tabla (Y)
        MOVLW .127      ; Valor de X máximo a mostrar en pantalla
        MOVWF maximoX
        MOVLW .0        ; Valor de X del que se partirá para mostrar el contenido en la pantalla
        MOVWF xfile
        MOVWF xreferencia
        RETURN

S_RESETY                ; Subrutina donde se reinician los valores de referencia de la tabla (X)
        MOVLW .184      ; Página 0 -> 10111000
        MOVWF yfile
        MOVLW .191      ; Página 1 -> 10111111
        MOVWF maximoY
        RETURN
```

Figura 4.13: Código que reinicia la posición el puntero que indica la posición de la RAM

Ahora el programa principal consta de una serie de subrutinas en las que se encuentran las diferentes funciones, en un principio estas solo van pasando de una pantalla a otra cuando se pulsa el pin RB7 situado en la placa.

El siguiente paso consistirá en introducir el código encargado de tomar los valores de la pantalla táctil en una subrutina, la cual se llamará S_LECTURA, esta guardará únicamente las coordenadas X e Y de la pulsación en la pantalla.

La nueva rutina pasa de ejecutarse en el programa principal a hacerlo en una subrutina, para poder usarlo cada vez que queramos pasar de una pantalla a otra. La interfaz constará de una serie de opciones que se mostrarán en pantalla, por las cuales se irá desplazando el usuario con la ayuda de la pantalla táctil, sin necesidad de recurrir a los pulsadores de la placa, por esta razón solo necesitaremos en un principio, la parte correspondiente del código que nos dice la coordenada pulsada sin necesidad de mostrarlo en la pantalla, una vez elegida la opción en la pantalla se podrá escribir libremente en la pantalla o probar otras funciones del código a través de un sencillo juego.

Para que el programa distinga entre una opción u otra se creó una rutina llamada ELECCIONMENU. Dicha rutina conoce la posición de cada opción en la pantalla y comparará que el valor obtenido esta dentro de la posición que marca cada opción, diferenciando entre ellas.

El funcionamiento consistió en que, una vez leído el valor de la pulsación, se empezaba operando con el valor de la coordenada Y, puesto que las dos opciones del menú principal se encontraban en la misma franja de Y. A este se le restaba el valor mínimo que debía poseer, si el resultado era mayor que 0 se pasaba a comparar por el máximo valor de Y, de ser menor, se procedía a operar con el valor de la coordenada X, en caso contrario se volvía a la subrutina S_LECTURA a tomar una nueva pulsación.

```
ELECCIONMENU:                   ; Rutina encargada de obtener qué valor del menú que se escogió

        MOVFF valorY,eleY       ; Traslado el valor final de la pulsación a eleY
        MOVLW .11               ; Valor mínimo de 'Y' que ha de tener
        SUBWF eleY              ; Se resta al valor de la pulsación
        BTFSS STATUS, C         ; Si C = 1 -> PC+4 (valorY>11)..Si C=0 -> PC+2 (valorY<11)
        GOTO MENU               ; Repito la medida porque el valorY no es de una opción del menú
        MOVLW .23               ; Valor máximo de eleY para que este dentro de una opción
        CPFSLT eleY             ; eley < 23 -> PC+4...eley>=23 -> PC+2
        GOTO MENU               ; Repito la medida porque el valorY no es de una opción
        MOVFF valorX,eleX       ; Traslado el valor final de la pulsación a eleX
        BCF eleX,6              ; Elimino el último valor tomando para que la operación sea simétrica
        MOVLW .8                ; Valor mínimo de 'X' que ha de tener
        SUBWF eleX              ; Se resta al valor de la pulsación
        BTFSS STATUS, C         ; Si C = 1 -> PC+4 (valorX>8)..Si C=0 -> PC+2 (valorX<8)
        GOTO MENU               ; Repito la medida porque el valorY no es de una opción del menú
        MOVLW .47               ; Valor máximo de eleX para que este dentro de una opción del menú
        CPFSLT eleX             ; eleX < 47 -> PC+4...eleX>=47 -> PC+2
        GOTO MENU               ; Repito la medida porque el valorY no es de una opción
        BTFSS valorX ,6         ; Al estar en posiciones simétricas decide qué lado de la pantalla se pulsó
        GOTO ESCRITURA  ;       ; Lado Izquierdo....opción de Escribir libremente
        GOTO TOPOS              ; Lado Derecho.....opción de Juego "Dale al topo"
```

Figura 4.14: Código de selección de una opción del menú

El procedimiento con la coordenada X era similar, si está la coordenada X dentro de los parámetros solo hay que distinguir si el valor pertenece a la parte izquierda o a la derecha de la pantalla, puesto que ambas opciones están colocadas de forma simétrica.

Este primer código que distingue entre dos posibles opciones en la pantalla, escribir libremente, la cual está situada a la izquierda de la pantalla, o jugar, situada a la derecha.

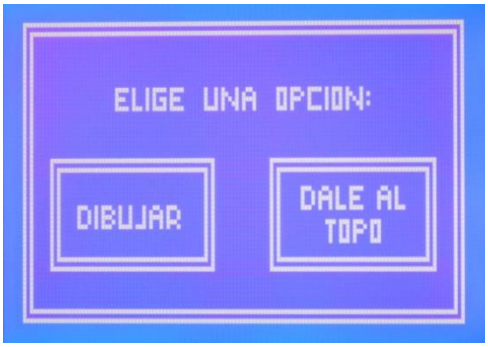


Figura 4.15: Pantalla de Menú

La prueba ejecuto el programa sin ningún contratiempo, pasando a una pantalla a otra según la opción escogida.

La última parte de esta etapa consistió en desarrollar un pequeño juego que hiciera necesario interactuar en mayor profundidad con la pantalla, donde se pudiera probar más a fondo el código desarrollado.

4.7. Juego “dale al topo”

Se trata del clásico juego de darle al topo, en el que la posición en donde se encuentra el topo irá variando aleatoriamente, estando un determinado tiempo activo en cada una de las 5 posiciones posibles, si el jugador acierta se suma un punto y en caso de fallar se resta una vida.

Primero se creó la pantalla del juego, el diseño de la misma constará de los marcadores correspondientes (Vidas, Puntos), dos opciones de menú, para reiniciar la partida o volver al menú principal y los “agujeros”, de donde irán saliendo los topos.

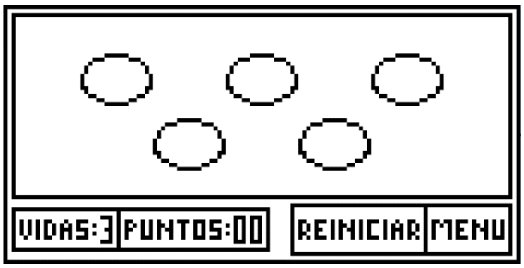


Figura 4.16: Interfaz diseñada a través de Paint

Como para el desarrollo de los apartados anteriores, se siguió el mismo método creando pequeñas funciones.

Hará falta una función que reconozca la pulsación en las posiciones correctas, también hará falta otra que modifique los valores de los marcadores. Además, será necesario otra función que dé como resultado números aleatorios, la cual se usará para elegir porque agujero saldrá el topo, aparte de mostrarlo.

Para la primera función se buscó la forma de desplazarse por las posiciones especiales de la pantalla, optando por el mismo método que en el menú principal. Realizar la comparación del valor pulsado con las posiciones en las que se encontrarían las posibles opciones.

Al haber más opciones en la pantalla, la función que reconocerá si la pulsación pertenece a un valor correcto o hay que desecharlo resulto ser más compleja, a diferencia de la función para el menú principal. Esta contaría con tres posibles franjas de Y que comparar, para luego pasar a la comparación de hasta siete franjas de X, las cuales había que variar dependiendo de la selección previa de Y.

Debido a la complejidad que iba a tomar esta función se fijó el ancho de todas las franjas de Y, haciendo que una de las comparaciones pudiera realizarse con un valor constante, sin necesidad de irlo variando, también se trató de hacer lo mismo con las franjas de X, pero solo se pudo aplicar a los agujeros de los topos, las opciones del menú dependerían del ancho que alcanzará su nombre.

Una vez planteada la función de selección se dividió nuevamente el proceso, primero se procedería a obtener la pulsación en los agujeros de los topos. Aun no se ha añadido la función que muestra los topos de manera aleatoria, así que únicamente se tratará de que el programa sepa reconocer estas posiciones especiales.

Para esto se copió el código usado en el menú principal, con la diferencia de que, a la hora de cargar los valores de referencia para realizar las comparaciones, entrará en una subrutina que los cargará. Se crearon dos, una para las comparaciones de cada eje.

Estas subrutinas contarán con todos los valores posibles de los que partirá cada franja, en el caso de Y, se empezará desde la parte inferior de la pantalla, e irá cargando el valor de la franja superior hasta llegar al máximo. Para el eje X había la particularidad de que el valor a comparar dependía del de Y, puesto que el reparto de los "agujeros" en el eje X no era simétrico por la pantalla, por tanto, había que añadir un contador que indicara en que comparación de Y se dio el valor correcto de la franja.

La salida de la función de selección tenía dos opciones. La primera, que una vez alcanzado el valor máximo de comparación se volviera a proceder a la lectura de un nuevo valor si ninguna comparación era verdadera y en caso de obtener una pulsación correcta se volvería a la pantalla del menú principal.

Esta primera función presento algún problema en las pruebas debido al número de comparaciones que había que realizar, tanto en las subrutinas como en la función principal, puesto que esto resultaba bastante tedioso.

Una vez solucionado esos pequeños problemas y obteniendo el resultado deseado en las posiciones especiales de la pantalla, se pasó a desarrollar una función que dependiendo de si la pulsación daba en una posición especial o no, modifique los marcadores.

El primer paso de esa función ya lo desarrollaba el código anterior, sobo hubo que centrarse en una subrutina que modificara la pantalla dependiendo del resultado de las comparaciones. La única opción viable que se presento fue la de utilizar el mismo código para mostrar las tablas por el GLCD.

Surgió un problema a la hora de realizar esto, debido a la manera usada para imprimir los valores por pantalla. Puesto que el código usado se situaba en la primera posición de RAM del GLCD, a partir de ahí solo había que ir saltando de página en página según las fuera completando, pero para mostrar valores en el marcador había que situarse en una posición concreta de la RAM de la GLCD, además estaba el problema añadido de que la información enviada tenía el tamaño de un byte, que era el alto de cada página, así que al tener el marcador situado en dos páginas era necesario modificar ambas a lo alto.

Hubo que modificar el código de la subrutina S_PANTALLA encargada de mostrar las tablas, se añadió cuatro registros al código, los cuales eran modificados dependiendo de la posición de partida y la última que debía tener la imagen mostrada en la pantalla.

Aparte fue necesario crear el mapa de bits para los números del marcador e introducirlos en una tabla, esta tabla tenía la particularidad con respecto a las demás de que hay que poner el puntero en la posición exacta, puesto que cada número está repartido por la tabla. A la subrutina que indica el puntero de la tabla se le añade una función que varíe este puntero dependiendo del número que se quiera leer.

Finalmente, hubo que realizar una modificación en la función de selección, ahora en caso de una pulsación en la posición correcta, modificará el valor del marcador. Primero se probó el código para variar un número del marcador, al solo ser necesario corregir puntualmente la operación que modifica el puntero de la tabla, se pasó a modificar los demás valores del marcador.

Modificando la función de selección para que sus únicas posibles salidas sean restar Vidas o variar la Puntuación, aun no se había implementado las opciones del menú que se encuentra en la parte derecha inferior de la pantalla pues requiere otras dos salidas diferentes para la rutina de selección y aun no se han desarrollado.

Dejando de lado la selección de las opciones del menú para más adelante, se pasa a la parte del código necesaria para mostrar los topos, esto se hará igual que se hizo con los números del marcador, guardando el dibujo del "topo" en una tabla, la diferencia es que en la posición que se mostrara, irá variando dependiendo de un número que se obtendrá de manera aleatoria.

Primero se hace la prueba de mostrar el topo en la pantalla, aquí surge un problema que no se presentó antes. Para mostrar los topos hay algunas posiciones que requieren pasar de un lado al otro de la pantalla, esto no causo errores antes para el funcionamiento del marcador puesto que solo se modificaba la parte izquierda de la pantalla, pero en este caso hay topos que empiezan en una parte de la pantalla y acaban en otra.

La causa del problema era que debido a que cada vez que se usaba la subrutina S_WRITE el controlador del GLCD apuntaba directamente a la siguiente posición de X. Debido a la forma de escribir la información en las páginas que conforman la figura, cuando se pasaba a la segunda página, continuaba por el último valor que se escribió en la pantalla de la derecha.

Para solucionar esto se añadió en la subrutina S_PANTALLA, que cargaba los valores de la tabla, la instrucción de control que indicaba la posición de X, realizándose cada vez que se guardaba un valor en la GLCD, evitando así que se vuelva a producir este fallo.

Por último, se añadió una función que generara un número aleatorio, se probó el funcionamiento de varios métodos adaptados a ASM, todos basados en el funcionamiento de puertas XOR.

Se llamó S_NUMEROALEATORIO a esta subrutina y como condición de inicio requiere de una semilla, tratando de evitar que se repita el patrón cada vez que se inicie el juego, el valor de la semilla se tomará a partir de la primera pulsación que se realice en la pantalla.

```
S_NUMEROALEATORIO
        ; subrutina que genera un valor aleatorio
    MOVFF raiz,Dato
    CALL RANDOM
    movff Dato,raiz
    movf optopo,W

SELECCIONTOPO:
    RLNCF optopo
    BTFSC optopo,5
    CALL S_INICIOCUENTA
    decf Dato
    BTFSS   STATUS, Z
    goto SELECCIONTOPO
    CPFSEQ optopo
    return
    goto S_NUMEROALEATORIO

RANDOM
    btfsc   Dato, 4
    incf   Conta, f
    btfsc   Dato, 3
    incf   Conta, f
    btfsc   Dato, 2
    incf   Conta, f
    btfsc   Dato, 0
    incf   Conta, f
    rrCf   Dato, f
    bsf   Dato, 7
    btfss   Conta, 0
    bcf   Dato, 7
    clrf   Conta
    return
```

Figura 4.17: Código encargado de la función Random

Para mostrar los “topos” de manera aleatoria se utilizaba el valor que nos devolvía la Subrutina RANDOM. Se hacía desplazar un bit dentro de un registro por sus 5 posiciones menos significativas, esto se hace de manera cíclica decrementando en cada salto 1 al valor devuelto por RANDOM, cuando este llegaba a 0 se salía de la rutina. Donde se hubiera parado dicho bit, indicará más adelante el “topo” que ha de mostrarse.

Por último, hacía falta utilizar un Timer para fijar el tiempo que cada topo estaría fijado en la pantalla antes de pasar al siguiente, como ya estaban configurados los registros del TMR0, lo único que hizo falta es añadir las instrucciones para encenderlo y que realizara una interrupción cuando se desbordara su registro.

Como ahora hay funciones diferentes dentro de las interrupciones, hubo que crear dos rutinas puesto que el código no cabe en el espacio destinado en el programa para las interrupciones. Dependiendo del Timer que activara la interrupción se pasaba a una rutina u otra, una para el TMR1, que cargaba los valores del ADC cada cierto tiempo, y otra para el TMR0 que borraba el topo que este activado y muestra uno nuevo.

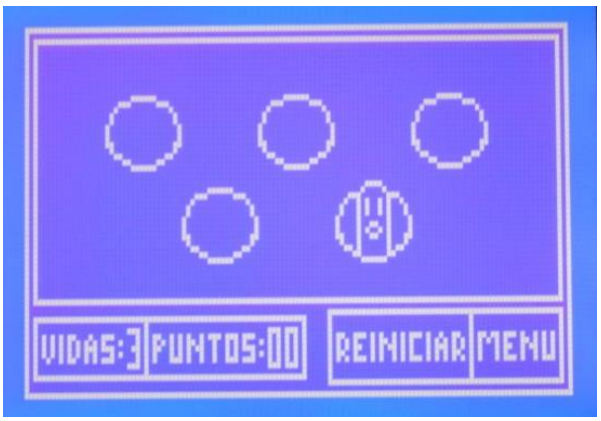


Figura 4.18: Pantalla del juego implementada en el GLCD

Esta última parte del programa que corresponde con mostrar los topos de manera aleatoria en la pantalla, se fue probando por separado y aunque en ninguno de los casos el código funciono al primer intento, los problemas solían deberse a descuidos en otros registros o pequeñas modificaciones necesarias en las subrutinas existentes.

Para finalizar el código y que el programa pudiera ejecutarse al completo, quedaba modificar la rutina que alberga la función de selección, para que solo tomara como medida correcta la posición en donde se mostrara el topo a la hora de realizar la pulsación, también habilitar las dos opciones de menú cuyo desarrollo se pospuso.

Se decidió modificar la función de selección eliminando las subrutinas que iban cambiando los valores de los registros de comparación. Se tomó esta solución puesto que solo

habría una posición correcta en cada instante, la del topo activado y luego estaría la del menú de opciones. Por tanto, a la hora de activar el topo, simultáneamente se modificaban los valores en los registros que se usaban para la comparación, siendo estos los de la posición del topo activado.

Por último, se añadió una comparación al principio de rutina de selección la cual en caso de encontrarse la pulsación en la franja de Y, donde está el menú de opciones, actualizaría los registros de comparación de X para comprobar si la pulsación ha sido encima de ellos, en caso contrario continuaría con los valores guardados en dichos registros, los cuales corresponderían a los cargados cuando se activaba el "topo" correspondiente. Además, se añadió una rutina al final de la comparación en caso de que la pulsación hubiera sido en las posiciones especiales (topos, menú), para discernir en cuál de ellas ha sido y actuar en consecuencia.

Una vez probadas estas modificaciones del código, se procedió a añadirle mejoras, como la de mostrar un mensaje cuando se pierda o se gane, esto se hizo añadiendo la imagen a una tabla y llevando una cuenta del marcador, una vez se llegar a cierto valor se mostraría el mensaje para que el usuario sepa que ha ganado o ha perdido.

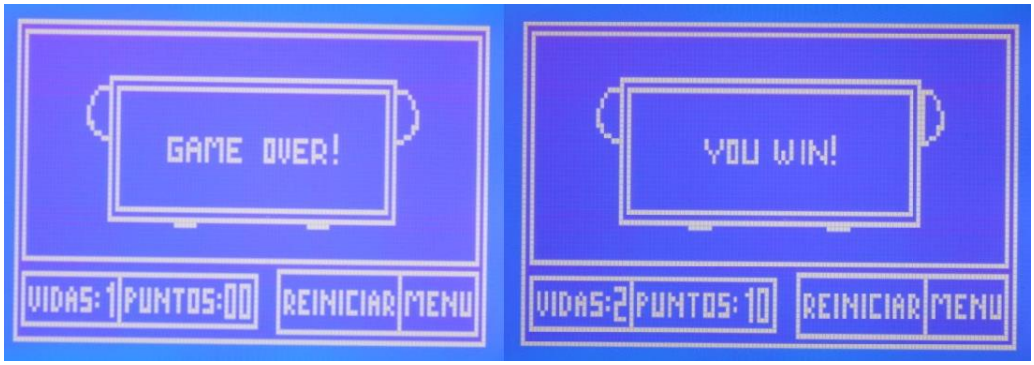


Figura 4.19: Mensajes al final de la partida

Con esto estaría terminado el código del juego, pudiéndose probar haciendo uso únicamente de la pantalla táctil, puesto que se habrían implementado todas las rutinas a las que se puede acceder en el programa a través de la interfaz creada.

4.8. Posibles mejoras del código

Finalmente se ha logrado un código fluido que permite el funcionamiento de las aplicaciones sin errores apreciables. Este último apartado tratará de reflejar las líneas abiertas del código, susceptibles de poder mejorarse.

- La principal mejora del programa radica en reducir los tiempos de adquisición del valor de la pulsación en la pantalla, el código final muestra la pulsación de manera correcta, pero a velocidades muy altas de desplazamiento tiende a no mostrar todos los valores formando una línea de puntos separados en la pantalla, esta parte es susceptible de mejorarse añadiendo rutinas de comparación que no requieran del uso de temporizadores.
- Otra mejora apreciable es la unir las rutinas de elección, usadas para discernir qué opción de la pantalla se ha usado, actualmente en el código existen tres rutinas de este tipo separadas, las cuales a grandes rasgos realizan la misma función.
- También existe la opción de añadir funciones de control de los demás dispositivos de la placa de desarrollo a través de la pantalla táctil, puesto que el código que la controla ya estaría implementado para un uso de este tipo.
- La última mejora apreciable tiene que ver con la forma de guardar los gráficos dentro del código, esto se ha hecho con el uso de mapas de bits, como el MCU cuenta con una gran cantidad de memoria se optó por este método. La mejora podría ser cambiar los mapas de bits por gráficos vectoriales, reduciendo en gran medida el espacio usado por las imágenes.

Capítulo 5.
Presupuesto

Capítulo 5. Presupuesto

En este capítulo se especificará el coste total del presente proyecto, se ha dividió en dos partes, una correspondiente a los costes de los componentes utilizados para el desarrollo del mismo y otro para la mano de obra necesaria para la creación del código.

5.1. Componentes

Componente	Cantidad	Precio unitario (€)	Subtotal
Placa de desarrollo EASY PIC V7	1	141,81	141,81 €
Graphic LCD 128x64 con pane táctil	1	22,84	22,84 €
		Total	164,65 €

5.2. Mano de obra

Concepto	Horas	Precio/hora (€)	Subtotal
Programación	200	40	8.000,00 €

5.3 Presupuesto total

Resumen presupuesto	
Coste total componentes	164,65 €
Coste total mano de obra	8.000,00 €
Total	8.164,65 €

Capítulo 6.
Conclusiones

Capítulo 6. Conclusiones

The result of this project is a solid code capable to control a GLCD with touch panel. it has been possible to obtain multitude of subroutines capable of being used in similar applications or components.

Another aspect of this project is the gained experience during the development process in assembly language. it was a secondary objective in this project due the previous knowledge in this language.

Finally, It has to be emphasize the utility of the Easy PIC V7 development board for accomplishing the aims. This component allowed to focus in the program activity, giving the possibility for a reducing work time and to avoid break the components accidentally.

Bibliografía

Bibliografía

[1] Salas Arriarán, Sergio. Todo sobre sistemas embebidos. 1º Edición. Ed Universidad Peruana de Ciencias Aplicas S.A.C. 2015

[2] View and Set Configuración Bits: http://microchipdeveloper.com/mplabx:view-and-set-configuration-bits

[3] LAB20: Interfacing A Ks0108 based graphics LCD: http://embedded-lab.com/blog/lab-20-interfacing-a-ks0108-based-graphics-lcd-part-1

[4] Fun With PIC Assembly - Episode 13: http://www.instructables.com/id/Fun-With-PIC-Assembly-Episode-13

[5] Tutoriales PIC: http://picfernalia.blogspot.com.es/2012/07/conversor-adc.html

[6] Tema: Mis primeros programas en ASM. PIC16F84A y PIC16F62A/648A: http://www.todopic.com.ar/foros/index.php?PHPSESSID=okj5srkjtrqlbiapoqd96qnsu6&topic=24720.msg201408#msg201408

[7] Generador de números aleatorios: http://www.servisystem.com.ar/foro/viewtopic.php?f=24&t=202

[8] Microcontroladores PIC – Programación en C con ejemplos: https://learn.mikroe.com/ebooks/microcontroladorespicc/front-matter/introduction/

Anexo I. Datasheets

Anexo I. Datasheets

En este anexo se incluirán parte de los datasheets de los dos componentes usados. En el caso del datasheet del MCU se ha optado por hacer una selección de las páginas que contienen los registros usados, así como algunos esquemas de funcionamiento y tablas, también se añade el Instrucción Set Summary, básico para el lenguaje ensamblador, se tomó esta decisión debido a la extensión de su datasheet el cual tiene 560 páginas, en caso de querer ahondar más en profundidad en dichos registros sería necesario recurrir al original del mismo.

Para el datasheet del GLCD únicamente se han añadido las instrucciones de control, el reparto de pines y algunos esquemas y tablas indicativas, puesto que es la única información que hizo falta para la programación.

Ambas documentaciones aparecen por el orden indicado a continuación:

1º Datasheet PIC18F45K22

2º Datasheet WDG0151

MICROCHIP

PIC18(L)F2X/4XK22
Data Sheet

28/40/44-Pin, Low-Power,
High-Performance Microcontrollers
with XLP Technology

Pin Diagrams (40-pin)

40-pin PDIP

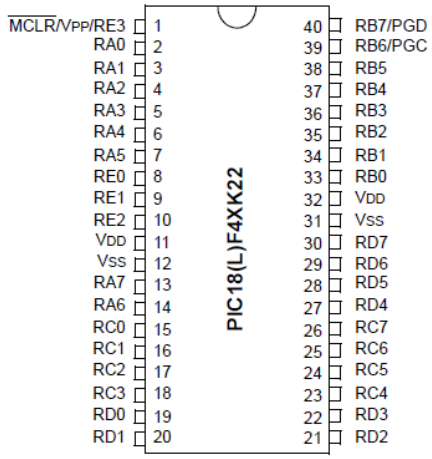


TABLE 2: PIC18(L)F4XK22 PIN SUMMARY

40-PDIP	40-UQFN	44-TQFP	44-QFN	I/O	Analog	Comparator	CTMU	SR Latch	Reference	(E)CCP	EUSART	MSSP	Timers	Interrupts	Pull-up	Basic
2	17	19	19	RA0	AN0	C12IN0-										
3	18	20	20	RA1	AN1	C12IN1-										
4	19	21	21	RA2	AN2	C2IN+			VREF- DACOUT							
5	20	22	22	RA3	AN3	C1IN+			VREF+							
6	21	23	23	RA4		C1OUT		SRQ					T0CKI			
7	22	24	24	RA5	AN4	C2OUT		SRNQ	HLVDIN			$\overline{SS1}$				
14	29	31	33	RA6												OSC2 CLKO
13	28	30	32	RA7												OSC1 CLKI
33	8	8	9	RB0	AN12			SRI		FLT0				INT0	Y	
34	9	9	10	RB1	AN10	C12IN3-								INT1	Y	
35	10	10	11	RB2	AN8		CTED1							INT2	Y	
36	11	11	12	RB3	AN9	C12IN2-	CTED2			CCP2 P2A(1)					Y	
37	12	14	14	RB4	AN11								T5G	IOC	Y	
38	13	15	15	RB5	AN13					CCP3 P3A(3)			T1G T3CKI(2)	IOC	Y	
39	14	16	16	RB6										IOC	Y	PGC
40	15	17	17	RB7										IOC	Y	PGD
15	30	32	34	RC0						P2B(4)			SOSCO T1CKI T3CKI(2) T3G			
16	31	35	35	RC1						CCP2(1) P2A			SOSCI			
17	32	36	36	RC2	AN14		CTPLS			CCP1 P1A			T5CKI			
18	33	37	37	RC3	AN15							SCK1 SCL1				
23	38	42	42	RC4	AN16							SDI1 SDA1				
24	39	43	43	RC5	AN17							SDO1				
25	40	44	44	RC6	AN18						TX1 CK1					
26	1	1	1	RC7	AN19						RX1 DT1					
19	34	38	38	RD0	AN20							SCK2 SCL2				
20	35	39	39	RD1	AN21					CCP4		SDI2 SDA2				
21	36	40	40	RD2	AN22					P2B(4)						
22	37	41	41	RD3	AN23					P2C		$\overline{SS2}$				
27	2	2	2	RD4	AN24					P2D		SDO2				
28	3	3	3	RD5	AN25					P1B						
29	4	4	4	RD6	AN26					P1C	TX2 CK2					
30	5	5	5	RD7	AN27					P1D	RX2 DT2					
8	23	25	25	RE0	AN5					CCP3 P3A(3)						

Note 1: CCP2 multiplexed in fuses.
2: T3CKI multiplexed in fuses.
3: CCP3/P3A multiplexed in fuses.
4: P2B multiplexed in fuses.

PIC18(L)F2X/4XK22

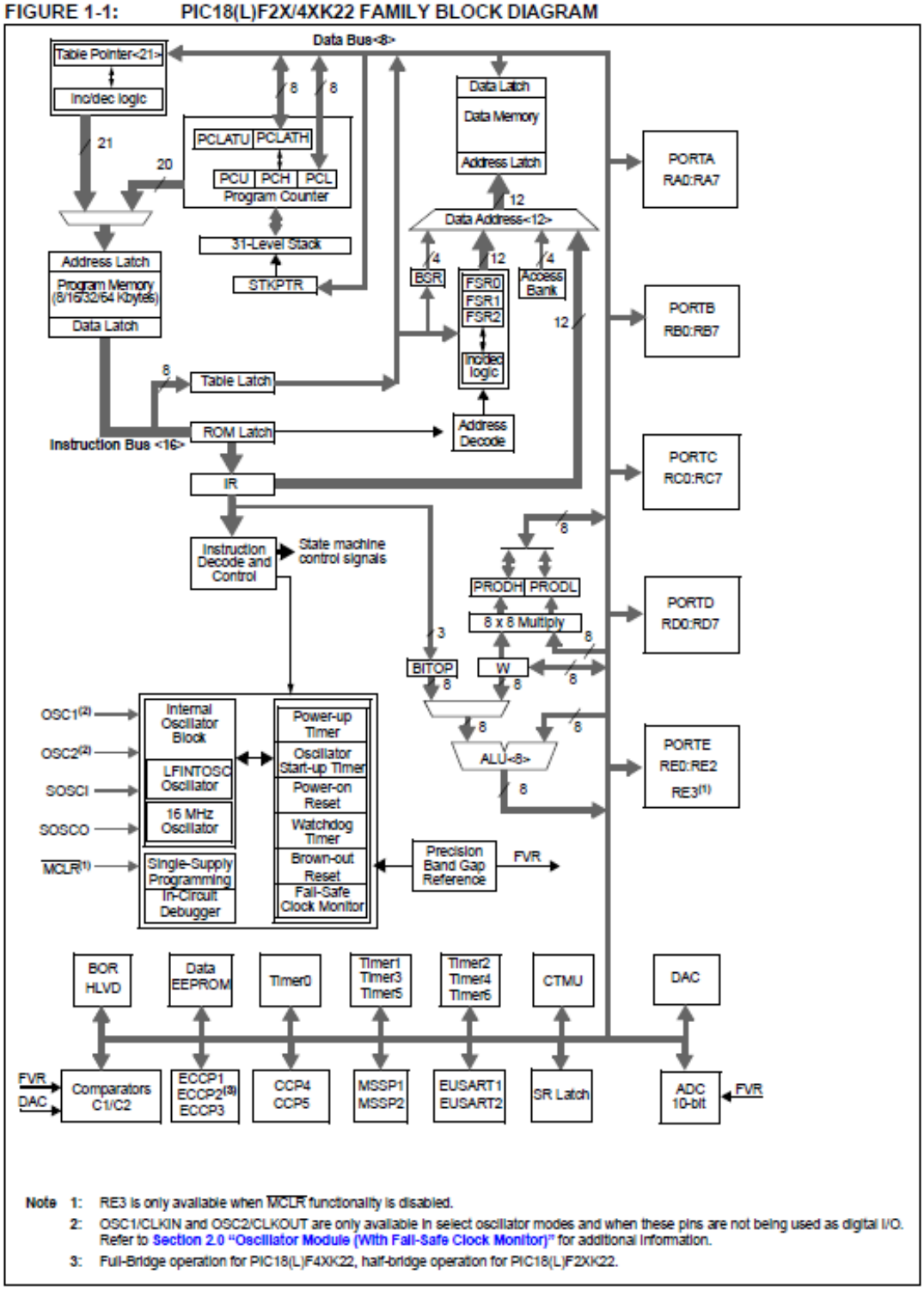
TABLE 2: PIC18(L)F4XK22 PIN SUMMARY (CONTINUED)

40-PDIP	40-UQFN	44-TQFP	44-QFN	I/O	Analog	Comparator	CTMU	SR Latch	Reference	(E)CCP	EUSART	MSSP	Timers	Interrupts	Pull-up	Basic
9	24	26	26	RE1	AN6					P3B						
10	25	27	27	RE2	AN7					CCP5						
1	16	18	18	RE3											Y	$\overline{MCLR}$ V_{PP}
11 32	7, 26	7 28	7,8 28, 29													V_{DD}
12 31	6, 27	6 29	6 30, 31													V_{SS}
—	—	12, 13 33, 34	13	NC												

Note 1: CCP2 multiplexed in fuses.
2: T3CKI multiplexed in fuses.
3: CCP3/P3A multiplexed in fuses.
4: P2B multiplexed in fuses.

PIC18(L)F2X/4XK22

FIGURE 1-1: PIC18(L)F2X/4XK22 FAMILY BLOCK DIAGRAM



Note 1: RE3 is only available when MCLR functionality is disabled.

2: OSC1/CLKIN and OSC2/CLKOUT are only available in select oscillator modes and when these pins are not being used as digital I/O. Refer to Section 2.0 "Oscillator Module (With Fail-Safe Clock Monitor)" for additional information.

3: Full-Bridge operation for PIC18(L)F4XK22, half-bridge operation for PIC18(L)F2XK22.

2.3 Register Definitions: Oscillator Control

REGISTER 2-1: OSCCON: OSCILLATOR CONTROL REGISTER

R/W-0	R/W-0	R/W-1	R/W-1	R-q	R-0	R/W-0	R/W-0
IDLEN	IRCF<2:0>			OSTS(1)	HFIOFS	SCS<1:0>	
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	q = depends on condition
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **IDLEN**: Idle Enable bit
1 = Device enters Idle mode on SLEEP instruction
0 = Device enters Sleep mode on SLEEP instruction

bit 6-4 **IRCF<2:0>**: Internal RC Oscillator Frequency Select bits(2)
111 = HFINTOSC – (16 MHz)
110 = HFINTOSC/2 – (8 MHz)
101 = HFINTOSC/4 – (4 MHz)
100 = HFINTOSC/8 – (2 MHz)
011 = HFINTOSC/16 – (1 MHz)(3)

If INTSRC = 0 and MFIOSEL = 0:
010 = HFINTOSC/32 – (500 kHz)
001 = HFINTOSC/64 – (250 kHz)
000 = LFINTOSC – (31.25 kHz)

If INTSRC = 1 and MFIOSEL = 0:
010 = HFINTOSC/32 – (500 kHz)
001 = HFINTOSC/64 – (250 kHz)
000 = HFINTOSC/512 – (31.25 kHz)

If INTSRC = 0 and MFIOSEL = 1:
010 = MFINTOSC – (500 kHz)
001 = MFINTOSC/2 – (250 kHz)
000 = LFINTOSC – (31.25 kHz)

If INTSRC = 1 and MFIOSEL = 1:
010 = MFINTOSC – (500 kHz)
001 = MFINTOSC/2 – (250 kHz)
000 = MFINTOSC/16 – (31.25 kHz)

bit 3 **OSTS**: Oscillator Start-up Time-out Status bit
1 = Device is running from the clock defined by FOSC<3:0> of the CONFIG1H register
0 = Device is running from the internal oscillator (HFINTOSC, MFINTOSC or LFINTOSC)

bit 2 **HFIOFS**: HFINTOSC Frequency Stable bit
1 = HFINTOSC frequency is stable
0 = HFINTOSC frequency is not stable

bit 1-0 **SCS<1:0>**: System Clock Select bit
1x = Internal oscillator block
01 = Secondary (SOSC) oscillator
00 = Primary clock (determined by FOSC<3:0> in CONFIG1H).

Note 1: Reset state depends on state of the IESO Configuration bit.
2: INTOSC source may be determined by the INTSRC bit in OSCTUNE and the MFIOSEL bit in OSCCON2.
3: Default output frequency of HFINTOSC on Reset.

REGISTER 2-2: OSCCON2: OSCILLATOR CONTROL REGISTER 2

R-0/0	R-0/q	U-0	R/W-0/0	R/W-0/u	R/W-1/1	R-x/u	R-0/0
PLLRDY	SOSCRUN	—	MFIOSEL	SOSCGO(1)	PRISD	MFIOFS	LFIOFS
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	q = depends on condition
'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown	
-n/n = Value at POR and BOR/Value at all other Resets			

bit 7 **PLLRDY**: PLL Run Status bit
1 = System clock comes from 4xPLL
0 = System clock comes from an oscillator, other than 4xPLL

bit 6 **SOSCRUN**: SOSC Run Status bit
1 = System clock comes from secondary SOSC
0 = System clock comes from an oscillator, other than SOSC

bit 5 **Unimplemented**: Read as '0'.

bit 4 **MFIOSEL**: MFINTOSC Select bit
1 = MFINTOSC is used in place of HFINTOSC frequencies of 500 kHz, 250 kHz and 31.25 kHz
0 = MFINTOSC is not used

bit 3 **SOSCGO**(1): Secondary Oscillator Start Control bit
1 = Secondary oscillator is enabled.
0 = Secondary oscillator is shut off if no other sources are requesting it.

bit 2 **PRISD**: Primary Oscillator Drive Circuit Shutdown bit
1 = Oscillator drive circuit on
0 = Oscillator drive circuit off (zero power)

bit 1 **MFIOFS**: MFINTOSC Frequency Stable bit
1 = MFINTOSC is stable
0 = MFINTOSC is not stable

bit 0 **LFIOFS**: LFINTOSC Frequency Stable bit
1 = LFINTOSC is stable
0 = LFINTOSC is not stable

Note 1: The SOSCGO bit is only reset on a POR Reset.

PIC18(L)F2X/4XK22

REGISTER 10-3: ANSELA – PORTA ANALOG SELECT REGISTER

U-0	U-0	R/W-1	U-0	R/W-1	R/W-1	R/W-1	R/W-1
—	—	ANSA5	—	ANSA3	ANSA2	ANSA1	ANSA0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7-6 **Unimplemented**: Read as '0'

bit 5 **ANSA5**: RA5 Analog Select bit
1 = Digital input buffer disabled
0 = Digital input buffer enabled

bit 4 **Unimplemented**: Read as '0'

bit 3-0 **ANSA<3:0>**: RA<3:0> Analog Select bit
1 = Digital input buffer disabled
0 = Digital input buffer enabled

REGISTER 10-8: TRISx: PORTx TRI-STATE REGISTER(1)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TRISx7	TRISx6	TRISx5	TRISx4	TRISx3	TRISx2	TRISx1	TRISx0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7-0 **TRISx<7:0>**: PORTx Tri-State Control bit
1 = PORTx pin configured as an input (tri-stated)
0 = PORTx pin configured as an output

Note 1: Register description for TRISA, TRISB, TRISC and TRISD.

PIC18(L)F2X/4XK22

17.3 Register Definitions: ADC Control

Note: Analog pin control is determined by the ANSELx registers (see Register 10-2)

REGISTER 17-1: ADCON0: A/D CONTROL REGISTER 0

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	CHS<4:0>					GO/$\overline{\text{DONE}}$	ADON
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **Unimplemented:** Read as '0'

bit 6-2 **CHS<4:0>: Analog Channel Select bits**
00000 = AN0
00001 = AN1
00010 = AN2
00011 = AN3
00100 = AN4
00101 = AN5(1)
00110 = AN6(1)
00111 = AN7(1)
01000 = AN8
01001 = AN9
01010 = AN10
01011 = AN11
01100 = AN12
01101 = AN13
01110 = AN14
01111 = AN15
10000 = AN16
10001 = AN17
10010 = AN18
10011 = AN19
10100 = AN20(1)
10101 = AN21(1)
10110 = AN22(1)
10111 = AN23(1)
11000 = AN24(1)
11001 = AN25(1)
11010 = AN26(1)
11011 = AN27(1)
11100 = Reserved
11101 = CTMU
11110 = DAC
11111 = FVR BUF2 (1.024V/2.048V/2.096V Volt Fixed Voltage Reference)(2)

bit 1 **GO/$\overline{\text{DONE}}$:** A/D Conversion Status bit
1 = A/D conversion cycle in progress. Setting this bit starts an A/D conversion cycle.
This bit is automatically cleared by hardware when the A/D conversion has completed.
0 = A/D conversion completed/not in progress

bit 0 **ADON:** ADC Enable bit
1 = ADC is enabled
0 = ADC is disabled and consumes no operating current

Note 1: Available on PIC18(L)F4XK22 devices only.
2: Allow greater than 15 μs acquisition time when measuring the Fixed Voltage Reference.

REGISTER 17-2: ADCON1: A/D CONTROL REGISTER 1

R/W-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
TRIGSEL	—	—	—	PVCFG<1:0>		NVCFG<1:0>	
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **TRIGSEL**: Special Trigger Select bit
1 = Selects the special trigger from CTMU
0 = Selects the special trigger from CCP5

bit 6-4 **Unimplemented**: Read as '0'

bit 3-2 **PVCFG<1:0>**: Positive Voltage Reference Configuration bits
00 = A/D VREF+ connected to internal signal, AVDD
01 = A/D VREF+ connected to external pin, VREF+
10 = A/D VREF+ connected to internal signal, FVR BUF2
11 = Reserved (by default, A/D VREF+ connected to internal signal, AVDD)

bit 1-0 **NVCFG<1:0>**: Negative Voltage Reference Configuration bits
00 = A/D VREF- connected to internal signal, AVSS
01 = A/D VREF- connected to external pin, VREF-
10 = Reserved (by default, A/D VREF- connected to internal signal, AVSS)
11 = Reserved (by default, A/D VREF- connected to internal signal, AVSS)

REGISTER 17-3: ADCON2: A/D CONTROL REGISTER 2

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	—	ACQT<2:0>			ADCS<2:0>		
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **ADFM**: A/D Conversion Result Format Select bit
1 = Right justified
0 = Left justified

bit 6 **Unimplemented**: Read as '0'

bit 5-3 **ACQT<2:0>**: A/D Acquisition time select bits. Acquisition time is the duration that the A/D charge holding capacitor remains connected to A/D channel from the instant the GO/$\overline{\text{DONE}}$ bit is set until conversions begins.
000 = 0(1)
001 = 2 TAD
010 = 4 TAD
011 = 6 TAD
100 = 8 TAD
101 = 12 TAD
110 = 16 TAD
111 = 20 TAD

bit 2-0 **ADCS<2:0>**: A/D Conversion Clock Select bits
000 = FOSC/2
001 = FOSC/8
010 = FOSC/32
011 = FRC(1) (clock derived from a dedicated internal oscillator = 600 kHz nominal)
100 = FOSC/4
101 = FOSC/16
110 = FOSC/64
111 = FRC(1) (clock derived from a dedicated internal oscillator = 600 kHz nominal)

Note 1: When the A/D clock source is selected as FRC then the start of conversion is delayed by one instruction cycle after the GO/$\overline{\text{DONE}}$ bit is set to allow the SLEEP instruction to be executed.

REGISTER 17-4: ADRESH: ADC RESULT REGISTER HIGH (ADRESH) ADFM = 0

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
ADRES<9:2>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7-0 **ADRES<9:2>**: ADC Result Register bits
Upper eight bits of 10-bit conversion result

REGISTER 17-5: ADRESL: ADC RESULT REGISTER LOW (ADRESL) ADFM = 0

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
ADRES<1:0>		—	—	—	—	—	—
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7-6 **ADRES<1:0>**: ADC Result Register bits
Lower two bits of 10-bit conversion result

bit 5-0 **Reserved**: Do not use.

REGISTER 17-6: ADRESH: ADC RESULT REGISTER HIGH (ADRESH) ADFM = 1

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	—	—	—	—	—	ADRES<9:8>	
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7-2 **Reserved**: Do not use.

bit 1-0 **ADRES<9:8>**: ADC Result Register bits
Upper two bits of 10-bit conversion result

REGISTER 17-7: ADRESL: ADC RESULT REGISTER LOW (ADRESL) ADFM = 1

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
ADRES<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7-0 **ADRES<7:0>**: ADC Result Register bits
Lower eight bits of 10-bit conversion result

11.0 TIMER0 MODULE

The Timer0 module incorporates the following features:

- Software selectable operation as a timer or counter in both 8-bit or 16-bit modes
- Readable and writable registers
- Dedicated 8-bit, software programmable prescaler
- Selectable clock source (internal or external)
- Edge select for external clock
- Interrupt-on-overflow

The T0CON register (Register 11-1) controls all aspects of the module's operation, including the prescale selection. It is both readable and writable.

A simplified block diagram of the Timer0 module in 8-bit mode is shown in Figure 11-1. Figure 11-2 shows a simplified block diagram of the Timer0 module in 16-bit mode.

11.1 Register Definitions: Timer0 Control

REGISTER 11-1: T0CON: TIMER0 CONTROL REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS<2:0>		
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **TMR0ON**: Timer0 On/Off Control bit
1 = Enables Timer0
0 = Stops Timer0

bit 6 **T08BIT**: Timer0 8-bit/16-bit Control bit
1 = Timer0 is configured as an 8-bit timer/counter
0 = Timer0 is configured as a 16-bit timer/counter

bit 5 **T0CS**: Timer0 Clock Source Select bit
1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (CLKOUT)

bit 4 **T0SE**: Timer0 Source Edge Select bit
1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin

bit 3 **PSA**: Timer0 Prescaler Assignment bit
1 = TImer0 prescaler is NOT assigned. Timer0 clock input bypasses prescaler.
0 = Timer0 prescaler is assigned. Timer0 clock input comes from prescaler output.

bit 2-0 **T0PS<2:0>**: Timer0 Prescaler Select bits
111 = 1:256 prescale value
110 = 1:128 prescale value
101 = 1:64 prescale value
100 = 1:32 prescale value
011 = 1:16 prescale value
010 = 1:8 prescale value
001 = 1:4 prescale value
000 = 1:2 prescale value

12.13 Register Definitions: Timer1/3/5 Control

REGISTER 12-1: TXCON: TIMER1/3/5 CONTROL REGISTER

R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/0	R/W-0/u
TMRxCS<1:0>		TxCKPS<1:0>		TxSOSCEN	$\overline{\text{TxSYNC}}$	TxRD16	TMRxON
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	

bit 7-6 **TMRxCS<1:0>**: Timer1/3/5 Clock Source Select bits
11 = Reserved. Do not use.
10 = Timer1/3/5 clock source is pin or oscillator:
If TxSOSCEN = 0:
External clock from TxCKI pin (on the rising edge)
If TxSOSCEN = 1:
Crystal oscillator on SOSCI/SOSCO pins
01 = Timer1/3/5 clock source is system clock (FOSC)
00 = Timer1/3/5 clock source is instruction clock (FOSC/4)

bit 5-4 **TxCKPS<1:0>**: Timer1/3/5 Input Clock Prescale Select bits
11 = 1:8 Prescale value
10 = 1:4 Prescale value
01 = 1:2 Prescale value
00 = 1:1 Prescale value

bit 3 **TxSOSCEN**: Secondary Oscillator Enable Control bit
1 = Dedicated Secondary oscillator circuit enabled
0 = Dedicated Secondary oscillator circuit disabled

bit 2 **$\overline{\text{TxSYNC}}$**: Timer1/3/5 External Clock Input Synchronization Control bit
TMRxCS<1:0> = 1X
1 = Do not synchronize external clock input
0 = Synchronize external clock input with system clock (FOSC)

TMRxCS<1:0> = 0X
This bit is ignored. Timer1/3/5 uses the internal clock when TMRxCS<1:0> = 1X.

bit 1 **TxRD16**: 16-Bit Read/Write Mode Enable bit
1 = Enables register read/write of Timer1/3/5 in one 16-bit operation
0 = Enables register read/write of Timer1/3/5 in two 8-bit operation

bit 0 **TMRxON**: Timer1/3/5 On bit
1 = Enables Timer1/3/5
0 = Stops Timer1/3/5
Clears Timer1/3/5 Gate flip-flop

PIC18(L)F2X/4XK22

9.8 Register Definitions: Interrupt Control

REGISTER 9-1: INTCON: INTERRUPT CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **GIE/GIEH:** Global Interrupt Enable bit
When IPEN = 0:
1 = Enables all unmasked interrupts
0 = Disables all interrupts including peripherals
When IPEN = 1:
1 = Enables all high priority interrupts
0 = Disables all interrupts including low priority

bit 6 **PEIE/GIEL:** Peripheral Interrupt Enable bit
When IPEN = 0:
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts
When IPEN = 1:
1 = Enables all low priority interrupts
0 = Disables all low priority interrupts

bit 5 **TMR0IE:** TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 overflow interrupt
0 = Disables the TMR0 overflow interrupt

bit 4 **INT0IE:** INT0 External Interrupt Enable bit
1 = Enables the INT0 external interrupt
0 = Disables the INT0 external interrupt

bit 3 **RBIE:** Port B Interrupt-On-Change (IOCx) Interrupt Enable bit(2)
1 = Enables the IOCx port change interrupt
0 = Disables the IOCx port change interrupt

bit 2 **TMR0IF:** TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared by software)
0 = TMR0 register did not overflow

bit 1 **INT0IF:** INT0 External Interrupt Flag bit
1 = The INT0 external interrupt occurred (must be cleared by software)
0 = The INT0 external interrupt did not occur

bit 0 **RBIF:** Port B Interrupt-On-Change (IOCx) Interrupt Flag bit(1)
1 = At least one of the IOC<3:0> (RB<7:4>) pins changed state (must be cleared by software)
0 = None of the IOC<3:0> (RB<7:4>) pins have changed state

Note 1: A mismatch condition will continue to set the RBIF bit. Reading PORTB will end the mismatch condition and allow the bit to be cleared.
2: RB port change interrupts also require the individual pin IOCB enables.

Note: Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the global enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt. This feature allows for software polling.

REGISTER 9-4: PIR1: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 1

U-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
—	ADIF	RC1IF	TX1IF	SSP1IF	CCP1IF	TMR2IF	TMR1IF
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **Unimplemented:** Read as '0'.

bit 6 **ADIF:** A/D Converter Interrupt Flag bit
1 = An A/D conversion completed (must be cleared by software)
0 = The A/D conversion is not complete or has not been started

bit 5 **RC1IF:** EUSART1 Receive Interrupt Flag bit
1 = The EUSART1 receive buffer, RCREG1, is full (cleared when RCREG1 is read)
0 = The EUSART1 receive buffer is empty

bit 4 **TX1IF:** EUSART1 Transmit Interrupt Flag bit
1 = The EUSART1 transmit buffer, TXREG1, is empty (cleared when TXREG1 is written)
0 = The EUSART1 transmit buffer is full

bit 3 **SSP1IF:** Master Synchronous Serial Port 1 Interrupt Flag bit
1 = The transmission/reception is complete (must be cleared by software)
0 = Waiting to transmit/receive

bit 2 **CCP1IF:** CCP1 Interrupt Flag bit
Capture mode:
1 = A TMR1 register capture occurred (must be cleared by software)
0 = No TMR1 register capture occurred
Compare mode:
1 = A TMR1 register compare match occurred (must be cleared by software)
0 = No TMR1 register compare match occurred
PWM mode:
Unused in this mode

bit 1 **TMR2IF:** TMR2 to PR2 Match Interrupt Flag bit
1 = TMR2 to PR2 match occurred (must be cleared by software)
0 = No TMR2 to PR2 match occurred

bit 0 **TMR1IF:** TMR1 Overflow Interrupt Flag bit
1 = TMR1 register overflowed (must be cleared by software)
0 = TMR1 register did not overflow

Note 1: Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the Global Interrupt Enable bit, GIE/GIEH of the INTCON register.

Note: User software should ensure the appropriate interrupt flag bits are cleared prior to enabling an interrupt and after servicing that interrupt.

PIC18(L)F2X/4XK22

REGISTER 9-9: PIE1: PERIPHERAL INTERRUPT ENABLE (FLAG) REGISTER 1

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	ADIE	RC1IE	TX1IE	SSP1IE	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **Unimplemented:** Read as '0'.

bit 6 **ADIE:** A/D Converter Interrupt Enable bit
1 = Enables the A/D interrupt
0 = Disables the A/D interrupt

bit 5 **RC1IE:** EUSART1 Receive Interrupt Enable bit
1 = Enables the EUSART1 receive interrupt
0 = Disables the EUSART1 receive interrupt

bit 4 **TX1IE:** EUSART1 Transmit Interrupt Enable bit
1 = Enables the EUSART1 transmit interrupt
0 = Disables the EUSART1 transmit interrupt

bit 3 **SSP1IE:** Master Synchronous Serial Port 1 Interrupt Enable bit
1 = Enables the MSSP1 interrupt
0 = Disables the MSSP1 interrupt

bit 2 **CCP1IE:** CCP1 Interrupt Enable bit
1 = Enables the CCP1 interrupt
0 = Disables the CCP1 interrupt

bit 1 **TMR2IE:** TMR2 to PR2 Match Interrupt Enable bit
1 = Enables the TMR2 to PR2 match interrupt
0 = Disables the TMR2 to PR2 match interrupt

bit 0 **TMR1IE:** TMR1 Overflow Interrupt Enable bit
1 = Enables the TMR1 overflow interrupt
0 = Disables the TMR1 overflow interrupt

PIC18(L)F2X/4XK22

TABLE 25-1: OPCODE FIELD DESCRIPTIONS

Field	Description
a	RAM access bit a = 0: RAM location in Access RAM (BSR register is ignored) a = 1: RAM bank is specified by BSR register
bbb	Bit address within an 8-bit file register (0 to 7).
BSR	Bank Select Register. Used to select the current RAM bank.
C, DC, Z, OV, N	ALU Status bits: Carry, Digit Carry, Zero, Overflow, Negative.
d	Destination select bit d = 0: store result in WREG d = 1: store result in file register f
dest	Destination: either the WREG register or the specified register file location.
f	8-bit Register file address (00h to FFh) or 2-bit FSR designator (0h to 3h).
f_s	12-bit Register file address (000h to FFFh). This is the source address.
f_d	12-bit Register file address (000h to FFFh). This is the destination address.
GIE	Global Interrupt Enable bit.
k	Literal field, constant data or label (may be either an 8-bit, 12-bit or a 20-bit value).
label	Label name.
mm * *+ *- +*	The mode of the TBLPTR register for the table read and table write instructions. Only used with table read and table write instructions: No change to register (such as TBLPTR with table reads and writes) Post-Increment register (such as TBLPTR with table reads and writes) Post-Decrement register (such as TBLPTR with table reads and writes) Pre-Increment register (such as TBLPTR with table reads and writes)
n	The relative address (2's complement number) for relative branch instructions or the direct address for CALL/BRANCH and RETURN instructions.
PC	Program Counter.
PCL	Program Counter Low Byte.
PCH	Program Counter High Byte.
PCLATH	Program Counter High Byte Latch.
PCLATU	Program Counter Upper Byte Latch.
$\overline{PD}$	Power-down bit.
PRODH	Product of Multiply High Byte.
PRODL	Product of Multiply Low Byte.
s	Fast Call/Return mode select bit s = 0: do not update into/from shadow registers s = 1: certain registers loaded into/from shadow registers (Fast mode)
TBLPTR	21-bit Table Pointer (points to a Program Memory location).
TABLAT	8-bit Table Latch.
$\overline{TO}$	Time-out bit.
TOS	Top-of-Stack.
u	Unused or unchanged.
WDT	Watchdog Timer.
WREG	Working register (accumulator).
x	Don't care ('0' or '1'). The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.
z_s	7-bit offset value for indirect addressing of register files (source).
z_d	7-bit offset value for indirect addressing of register files (destination).
{ }	Optional argument.
[text]	Indicates an indexed address.
(text)	The contents of text.
[expr]<n>	Specifies bit n of the register indicated by the pointer expr.
→	Assigned to.
< >	Register bit field.
∈	In the set of.
italics	User defined term (font is Courier).

PIC18(L)F2X/4XK22

TABLE 25-2: PIC18(L)F2X/4XK22 INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word MSb			LSb	Status Affected	Notes
BYTE-ORIENTED OPERATIONS									
ADDWF	f, d, a	Add WREG and f	1	0010	01da	ffff	ffff	C, DC, Z, OV, N	1, 2
ADDWFC	f, d, a	Add WREG and CARRY bit to f	1	0010	00da	ffff	ffff	C, DC, Z, OV, N	1, 2
ANDWF	f, d, a	AND WREG with f	1	0001	01da	ffff	ffff	Z, N	1,2
CLRF	f, a	Clear f	1	0110	101a	ffff	ffff	Z	2
COMF	f, d, a	Complement f	1	0001	11da	ffff	ffff	Z, N	1, 2
CPFSEQ	f, a	Compare f with WREG, skip =	1 (2 or 3)	0110	001a	ffff	ffff	None	4
CPFSGT	f, a	Compare f with WREG, skip >	1 (2 or 3)	0110	010a	ffff	ffff	None	4
CPFSLT	f, a	Compare f with WREG, skip <	1 (2 or 3)	0110	000a	ffff	ffff	None	1, 2
DECF	f, d, a	Decrement f	1	0000	01da	ffff	ffff	C, DC, Z, OV, N	1, 2, 3, 4
DECFSZ	f, d, a	Decrement f, Skip if 0	1 (2 or 3)	0010	11da	ffff	ffff	None	1, 2, 3, 4
DCFSNZ	f, d, a	Decrement f, Skip if Not 0	1 (2 or 3)	0100	11da	ffff	ffff	None	1, 2
INCF	f, d, a	Increment f	1	0010	10da	ffff	ffff	C, DC, Z, OV, N	1, 2, 3, 4
INCFSZ	f, d, a	Increment f, Skip if 0	1 (2 or 3)	0011	11da	ffff	ffff	None	4
INFSNZ	f, d, a	Increment f, Skip if Not 0	1 (2 or 3)	0100	10da	ffff	ffff	None	1, 2
IORWF	f, d, a	Inclusive OR WREG with f	1	0001	00da	ffff	ffff	Z, N	1, 2
MOVF	f, d, a	Move f	1	0101	00da	ffff	ffff	Z, N	1
MOVFF	f_s, f_d	Move f_s (source) to 1st word f_d (destination) 2nd word	2	1100 1111	ffff ffff	ffff ffff	ffff ffff	None	
MOVWF	f, a	Move WREG to f	1	0110	111a	ffff	ffff	None	
MULWF	f, a	Multiply WREG with f	1	0000	001a	ffff	ffff	None	1, 2
NEGF	f, a	Negate f	1	0110	110a	ffff	ffff	C, DC, Z, OV, N	
RLCF	f, d, a	Rotate Left f through Carry	1	0011	01da	ffff	ffff	C, Z, N	1, 2
RLNCF	f, d, a	Rotate Left f (No Carry)	1	0100	01da	ffff	ffff	Z, N	
RRCF	f, d, a	Rotate Right f through Carry	1	0011	00da	ffff	ffff	C, Z, N	
RRNCF	f, d, a	Rotate Right f (No Carry)	1	0100	00da	ffff	ffff	Z, N	
SETF	f, a	Set f	1	0110	100a	ffff	ffff	None	1, 2
SUBFWB	f, d, a	Subtract f from WREG with borrow	1	0101	01da	ffff	ffff	C, DC, Z, OV, N	
SUBWF	f, d, a	Subtract WREG from f	1	0101	11da	ffff	ffff	C, DC, Z, OV, N	1, 2
SUBWFB	f, d, a	Subtract WREG from f with borrow	1	0101	10da	ffff	ffff	C, DC, Z, OV, N	
SWAPF	f, d, a	Swap nibbles in f	1	0011	10da	ffff	ffff	None	4
TSTFSZ	f, a	Test f, skip if 0	1 (2 or 3)	0110	011a	ffff	ffff	None	1, 2
XORWF	f, d, a	Exclusive OR WREG with f	1	0001	10da	ffff	ffff	Z, N	

Note 1: When a PORT register is modified as a function of itself (e.g., `MOVF PORTB, 1, 0`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and where applicable, 'd' = 1), the prescaler will be cleared if assigned.

3: If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a `NOP`.

4: Some instructions are two-word instructions. The second word of these instructions will be executed as a `NOP` unless the first word of the instruction retrieves the information embedded in these 16 bits. This ensures that all program memory locations have a valid instruction.

TABLE 25-2: PIC18(L)F2X/4XK22 INSTRUCTION SET (CONTINUED)

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word MSb			LSb	Status Affected	Notes
BIT-ORIENTED OPERATIONS									
BCF	f, b, a	Bit Clear f	1	1001	bbba	ffff	ffff	None	1, 2
BSF	f, b, a	Bit Set f	1	1000	bbba	ffff	ffff	None	1, 2
BTFSC	f, b, a	Bit Test f, Skip if Clear	1 (2 or 3)	1011	bbba	ffff	ffff	None	3, 4
BTFSS	f, b, a	Bit Test f, Skip if Set	1 (2 or 3)	1010	bbba	ffff	ffff	None	3, 4
BTG	f, b, a	Bit Toggle f	1	0111	bbba	ffff	ffff	None	1, 2
CONTROL OPERATIONS									
BC	n	Branch if Carry	1 (2)	1110	0010	nnnn	nnnn	None	
BN	n	Branch if Negative	1 (2)	1110	0110	nnnn	nnnn	None	
BNC	n	Branch if Not Carry	1 (2)	1110	0011	nnnn	nnnn	None	
BNN	n	Branch if Not Negative	1 (2)	1110	0111	nnnn	nnnn	None	
BNOV	n	Branch if Not Overflow	1 (2)	1110	0101	nnnn	nnnn	None	
BNZ	n	Branch if Not Zero	1 (2)	1110	0001	nnnn	nnnn	None	
BOV	n	Branch if Overflow	1 (2)	1110	0100	nnnn	nnnn	None	
BRA	n	Branch Unconditionally	2	1101	0nnn	nnnn	nnnn	None	
BZ	n	Branch if Zero	1 (2)	1110	0000	nnnn	nnnn	None	
CALL	k, s	Call subroutine 1st word	2	1110	110s	kkkk	kkkk	None	
		2nd word		1111	kkkk	kkkk	kkkk		
CLRWDT	—	Clear Watchdog Timer	1	0000	0000	0000	0100	$\overline{TO}$, $\overline{PD}$	
DAW	—	Decimal Adjust WREG	1	0000	0000	0000	0111	C	
GOTO	k	Go to address 1st word	2	1110	1111	kkkk	kkkk	None	
		2nd word		1111	kkkk	kkkk	kkkk		
NOP	—	No Operation	1	0000	0000	0000	0000	None	
NOP	—	No Operation	1	1111	xxxx	xxxx	xxxx	None	4
POP	—	Pop top of return stack (TOS)	1	0000	0000	0000	0110	None	
PUSH	—	Push top of return stack (TOS)	1	0000	0000	0000	0101	None	
RCALL	n	Relative Call	2	1101	1nnn	nnnn	nnnn	None	
RESET		Software device Reset	1	0000	0000	1111	1111	All	
RETFIE	s	Return from interrupt enable	2	0000	0000	0001	000s	GIE/GIEH, PEIE/GIEL	
RETLW	k	Return with literal in WREG	2	0000	1100	kkkk	kkkk	None	
RETURN	s	Return from Subroutine	2	0000	0000	0001	001s	None	
SLEEP	—	Go into Standby mode	1	0000	0000	0000	0011	$\overline{TO}$, $\overline{PD}$	

Note 1: When a PORT register is modified as a function of itself (e.g., `MOVF PORTB, 1, 0`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and where applicable, 'd' = 1), the prescaler will be cleared if assigned.

3: If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a `NOP`.

4: Some instructions are two-word instructions. The second word of these instructions will be executed as a `NOP` unless the first word of the instruction retrieves the information embedded in these 16 bits. This ensures that all program memory locations have a valid instruction.

PIC18(L)F2X/4XK22

TABLE 25-2: PIC18(L)F2X/4XK22 INSTRUCTION SET (CONTINUED)

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word MSb			LSb	Status Affected	Notes
LITERAL OPERATIONS									
ADDLW	k	Add literal and WREG	1	0000	1111	kkkk	kkkk	C, DC, Z, OV, N	
ANDLW	k	AND literal with WREG	1	0000	1011	kkkk	kkkk	Z, N	
IORLW	k	Inclusive OR literal with WREG	1	0000	1001	kkkk	kkkk	Z, N	
LFSR	f, k	Move literal (12-bit) 2nd word to FSR(f) 1st word	2	1110 1111	1110 0000	00ff kkkk	kkkk kkkk	None	
MOVLB	k	Move literal to BSR<3:0>	1	0000	0001	0000	kkkk	None	
MOVLW	k	Move literal to WREG	1	0000	1110	kkkk	kkkk	None	
MULLW	k	Multiply literal with WREG	1	0000	1101	kkkk	kkkk	None	
RETLW	k	Return with literal in WREG	2	0000	1100	kkkk	kkkk	None	
SUBLW	k	Subtract WREG from literal	1	0000	1000	kkkk	kkkk	C, DC, Z, OV, N	
XORLW	k	Exclusive OR literal with WREG	1	0000	1010	kkkk	kkkk	Z, N	
DATA MEMORY ↔ PROGRAM MEMORY OPERATIONS									
TBLRD*		Table Read	2	0000	0000	0000	1000	None	
TBLRD*+		Table Read with post-increment		0000	0000	0000	1001	None	
TBLRD*-		Table Read with post-decrement		0000	0000	0000	1010	None	
TBLRD+*		Table Read with pre-increment		0000	0000	0000	1011	None	
TBLWT*		Table Write	2	0000	0000	0000	1100	None	
TBLWT*+		Table Write with post-increment		0000	0000	0000	1101	None	
TBLWT*-		Table Write with post-decrement		0000	0000	0000	1110	None	
TBLWT+*		Table Write with pre-increment		0000	0000	0000	1111	None	

Note 1: When a PORT register is modified as a function of itself (e.g., `MOVF PORTB, 1, 0`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and where applicable, 'd' = 1), the prescaler will be cleared if assigned.

3: If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a `NOP`.

4: Some instructions are two-word instructions. The second word of these instructions will be executed as a `NOP` unless the first word of the instruction retrieves the information embedded in these 16 bits. This ensures that all program memory locations have a valid instruction.

25.2 Extended Instruction Set

In addition to the standard 75 instructions of the PIC18 instruction set, PIC18(L)F2X/4XK22 devices also provide an optional extension to the core CPU functionality. The added features include eight additional instructions that augment indirect and indexed addressing operations and the implementation of Indexed Literal Offset Addressing mode for many of the standard PIC18 instructions.

The additional features of the extended instruction set are disabled by default. To enable them, users must set the XINST Configuration bit.

The instructions in the extended set can all be classified as literal operations, which either manipulate the File Select Registers, or use them for indexed addressing. Two of the instructions, ADDFSR and SUBFSR, each have an additional special instantiation for using FSR2. These versions (ADDULNK and SUBULNK) allow for automatic return after execution.

The extended instructions are specifically implemented to optimize re-entrant program code (that is, code that is recursive or that uses a software stack) written in high-level languages, particularly C. Among other things, they allow users working in high-level languages to perform certain operations on data structures more efficiently. These include:

- dynamic allocation and deallocation of software stack space when entering and leaving subroutines
- function pointer invocation
- software Stack Pointer manipulation
- manipulation of variables located in a software stack

A summary of the instructions in the extended instruction set is provided in Table 25-3. Detailed descriptions are provided in **Section 25.2.2 "Extended Instruction Set"**. The opcode field descriptions in Table 25-1 apply to both the standard and extended PIC18 instruction sets.

Note:	The instruction set extension and the Indexed Literal Offset Addressing mode were designed for optimizing applications written in C; the user may likely never use these instructions directly in assembler. The syntax for these commands is provided as a reference for users who may be reviewing code that has been generated by a compiler.

25.2.1 EXTENDED INSTRUCTION SYNTAX

Most of the extended instructions use indexed arguments, using one of the File Select Registers and some offset to specify a source or destination register. When an argument for an instruction serves as part of indexed addressing, it is enclosed in square brackets ("[]"). This is done to indicate that the argument is used as an index or offset. MPASM™ Assembler will flag an error if it determines that an index or offset value is not bracketed.

When the extended instruction set is enabled, brackets are also used to indicate index arguments in byte-oriented and bit-oriented instructions. This is in addition to other changes in their syntax. For more details, see **Section 25.2.3.1 "Extended Instruction Syntax with Standard PIC18 Commands"**.

Note:	In the past, square brackets have been used to denote optional arguments in the PIC18 and earlier instruction sets. In this text and going forward, optional arguments are denoted by braces ("{}").

TABLE 25-3: EXTENSIONS TO THE PIC18 INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word MSb			LSb	Status Affected
ADDFSR	f, k	Add literal to FSR	1	1110	1000	ffkk	kkkk	None
ADDULNK	k	Add literal to FSR2 and return	2	1110	1000	11kk	kkkk	None
CALLW		Call subroutine using WREG	2	0000	0000	0001	0100	None
MOVSF	z_s, f_d	Move z_s (source) to 1st word f_d (destination) 2nd word	2	1110 1111	1011 ffff	0zzz ffff	zzzz ffff	None
MOVSS	z_s, z_d	Move z_s (source) to 1st word z_d (destination) 2nd word	2	1110 1111	1011 xxxx	1zzz xzzz	zzzz zzzz	None
PUSHL	k	Store literal at FSR2, decrement FSR2	1	1110	1010	kkkk	kkkk	None
SUBFSR	f, k	Subtract literal from FSR	1	1110	1001	ffkk	kkkk	None
SUBULNK	k	Subtract literal from FSR2 and return	2	1110	1001	11kk	kkkk	None

Winstar Display Co., LTD
華凌光電股份有限公司

住址: 407 台中市中清路 163 號
No.163 Chung Ching RD.,
Taichung, Taiwan, R.O.C

WEB: http://www.winstar.com.tw
E-mail: sales@winstar.com.tw
Tel:886-4-24262208 Fax : 886-4-24262207

SPECIFICATION

CUSTOMER : _____

MODULE NO.: WDG0151-TMI-V#N00

APPROVED BY: **(FOR CUSTOMER USE ONLY)**	**PCB VERSION:** **DATA:**

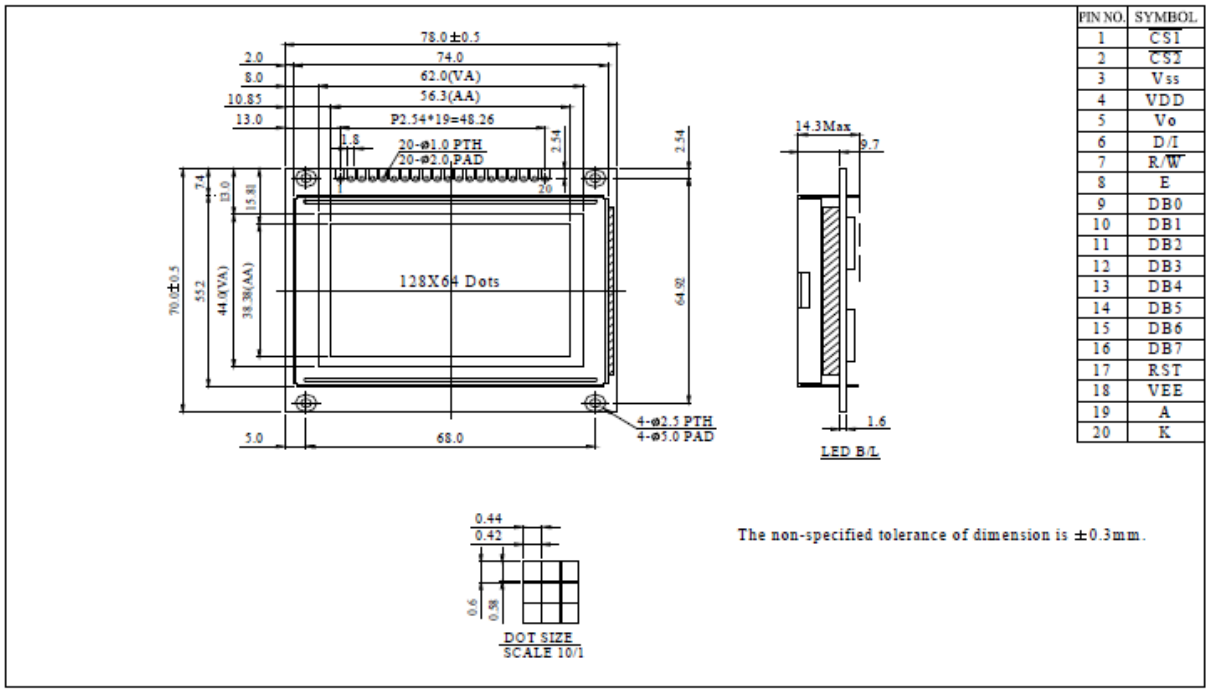
SALES BY	APPROVED BY	CHECKED BY	PREPARED BY

VERSION	DATE	REVISED PAGE NO.	SUMMARY
0	2009/11/10		First issue

7.Interface Pin Function

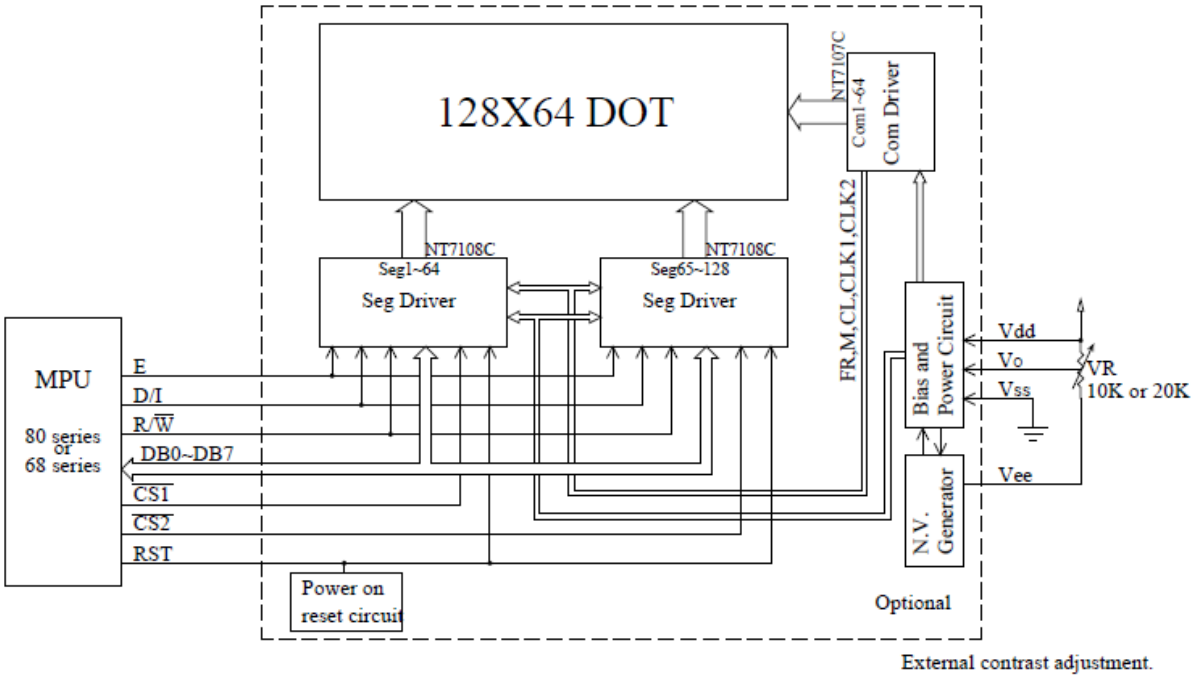
Pin No.	Symbol	Level	Description
1	$\overline{CS1}$	L	Select Segment 1 ~ Segment 64
2	$\overline{CS2}$	L	Select Segment 65 ~ Segment128
3	Vss	0V	Ground
4	V_{DD}	5.0V	Supply voltage for logic
5	V_O	(Variable)	Operating voltage for LCD
6	D/I	H/L	H: Data , L: Instruction
7	R/W	H/L	H: Read(MPU Module) , L :Write(MPU→ Module)
8	E	H	Enable signal
9	DB0	H/L	Data bus line
10	DB1	H/L	Data bus line
11	DB2	H/L	Data bus line
12	DB3	H/L	Data bus line
13	DB4	H/L	Data bus line
14	DB5	H/L	Data bus line
15	DB6	H/L	Data bus line
16	DB7	H/L	Data bus line
17	RST	L	Reset the LCM
18	VEE		Negative Voltage Output
19	A		Power supply for B/L(+)
20	K		Power supply for B/L(-)

8.Counter Drawing & Block diagram



PIN NO.	SYMBOL
1	$\overline{CS1}$
2	$\overline{CS2}$
3	Vss
4	VDD
5	Vo
6	D/I
7	R/$\overline{W}$
8	E
9	DB0
10	DB1
11	DB2
12	DB3
13	DB4
14	DB5
15	DB6
16	DB7
17	RST
18	VEE
19	A
20	K

Page 9 of 30

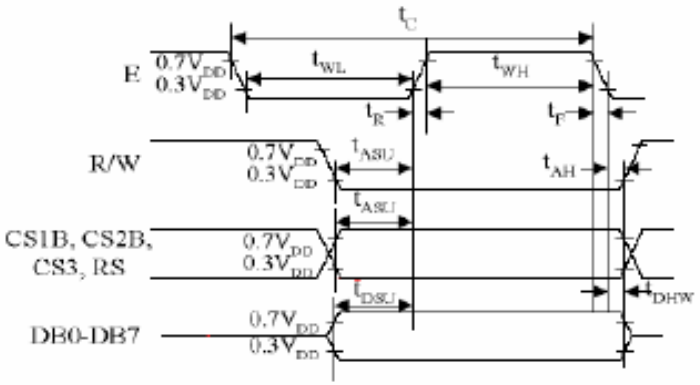

128X64 DOT
NT7107C
Com1~64
Com Driver
NT7108C
Seg1~64
Seg Driver
NT7108C
Seg65~128
Seg Driver
FR,M,CL,CLK1,CLK2
Bias and Power Circuit
N.V. Generator
Vdd
Vo
Vss
Vee
VR
10K or 20K
MPU
80 series
or
68 series
E
D/I
R/W
DB0~DB7
CS1
CS2
RST
Power on reset circuit
Optional
External contrast adjustment.

Page 10 of 30

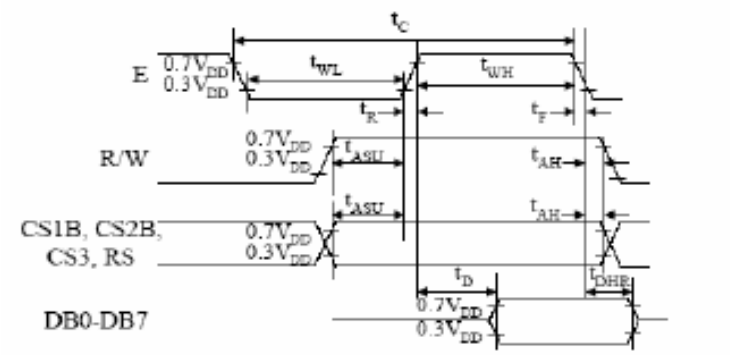
9.Timing Characteristics

MPU Interface (T=25°C, VDD=+5.0V±0.5)

Characteristic	Symbol	Min	Typ	Max	Unit
E cycle	tcyc	1000	-	-	ns
E high level width	twhE	450	-	-	ns
E low level width	twlE	450	-	-	ns
E rise time	tr	-	-	25	ns
E tall time	tf	-	-	25	ns
Address set-up time	tas	140	-	-	ns
Address hold time	tah	10	-	-	ns
Data set-up time	tdsw	140	-	-	ns
Data delay time	tddr	-	-	320	ns
Data hold time (write)	tdhw	10	-	-	ns
Data hold time (read)	tdhr	20	-	-	ns



MPU Write Timing



MPU Read Timing

10.Display Control Instruction

The display control instructions control the internal state of the NT7108. Instruction is received from MPU to NT7108 for the display control. The following table shows various instructions.

Instruction	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Function
Display on/off	L	L	L	L	H	H	H	H	H	L/H	Controls the display on or off. Internal status and display RAM data is not affected. L:OFF, H:ON
Set address (Y address)	L	L	L	H	Y address (0-63)						Sets the Y address in the Y address counter.
Set page (X address)	L	L	H	L	H	H	H	Page (0-7)			Sets the X address at the X address register.
Display Start line (Z address)	L	L	H	H	Display start line (0-63)						Indicates the display data RAM displayed at the top of the screen.
Status read	L	H	Busy	L	On/ Off	Reset	L	L	L	L	Read status. BUSY L: Ready H: In operation ON/OFF L: Display ON H: Display OFF RESET L: Normal H: Reset
Write display data	H	L	Write data								Writes data (DB0: 7) into display data RAM. After writing instruction, Y address is increased by 1 automatically.
Read display data	H	H	Read data								Reads data (DB0: 7) from display data RAM to the data bus.

11.Detailed Explanation

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	1	1	1	1	1	D

The display data appears when D is 1 and disappears when D is 0. Though the data is not on the screen with D=0, it remains in the display data RAM. Therefore, you can make it appear by changing D=0 into D=1.

SET ADDRESS (Y ADDRESS)

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0

Y address (AC0-AC5) of the display data RAM is set in the Y address counter. An address is set by instruction and increased by 1 automatically by read or write operations of display data.

SET PAGE (X ADDRESS)

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	1	0	1	1	1	AC2	AC1	AC0

X address (AC0-AC2) of the display data RAM is set in the X address register. Writing or reading to or from MPU is executed in this specified page until the next page is set.

DISPLAY START LINE (Z ADDRESS)

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	1	1	AC5	AC4	AC3	AC2	AC1	AC0

Z address (AC0-AC5) of the display data RAM is set in the display start line register and displayed at the top of the screen. When the display duty cycle is 1/64 or others (1/32-1/64), the data of total line number of LCD screen, from the line specified by display start line instruction, is displayed.

STATUS READ

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	1	BUSY	0	ON/OFF	RESET	0	0	0	0

BUSY

When BUSY is 1, the Chip is executing internal operation and no instructions are accepted.

When BUSY is 0, the Chip is ready to accept any instructions.

ON/OFF

When ON/OFF is 1, the display is OFF.

When ON/OFF is 0, the display is ON.

RESET

When RESET is 1, the system is being initialized.

In this condition, no instructions except status read can be accepted.

When RESET is 0, initializing has finished and the system is in usual operation condition.

WRITE DISPLAY DATA

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
1	0	D7	D6	D5	D4	D3	D2	D1	D0

Writes data (D0-D7) into the display data RAM. After writing instruction, Y address is increased by 1automatically.

READ DISPLAY DATA

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
1	1	D7	D6	D5	D4	D3	D2	D1	D0

Reads data (D0-D7) from the display data RAM. After reading instruction, Y address is increased by 1 automatically.

Anexo II. Código

Anexo II. Código

En este anexo se encuentra el código final para el microcontrolador PIC18F45K22 en lenguaje ASM, resultado de este proyecto. Se han añadido algunos comentarios al mismo para que resultara más sencilla su comprensión, son a titulo orientativo puesto que han servido al programador para el desarrollo del mismo.

Resaltar que la explicación de las rutinas y subrutinas se hace en las mismas y no sobre la instrucción que la inicia.

```
; *************************************************************************************************
; Titulo del Proyecto: Implementación de un sistema de control de un dispositivo táctil a partir de microcontroladores PIC en lenguaje ensamblador
; MCU: PIC18F45K22
; Modelo GLCD: WDG0151
; Controlador del GLCD NT7108 o similar
; Autor: Sergio Huarte Pulido
; Tutor: Evelio José González González
; Fecha inicio: 1/07/16
; Fecha de finalización: 10/2/2017
; *************************************************************************************************
 #include    <P18F45K22.INC>
; *************************************************************************************************
; BITS DE CONFIGURACIÓN
; *************************************************************************************************

; CONFIG1H
  CONFIG  FOSC = HSMP          ; Oscillator Selection bits (HS oscillator (medium power 4-16 MHz))
  CONFIG  PLLCFG = OFF         ; 4X PLL Enable (Oscillator used directly)
  CONFIG  PRICLKEN = ON        ; Primary clock enable bit (Primary clock is always enabled)
  CONFIG  FCMEN = OFF          ; Fail-Safe Clock Monitor Enable bit (Fail-Safe Clock Monitor disabled)
  CONFIG  IESO = OFF           ; Internal/External Oscillator Switchover bit (Oscillator Switchover mode disabled)

; CONFIG2L
  CONFIG  PWRTEN = OFF         ; Power-up Timer Enable bit (Power up timer disabled)
  CONFIG  BOREN = OFF          ; Brown-out Reset Enable bits (Brown-out Reset disabled in hardware and software)
  CONFIG  BORV = 190           ; Brown Out Reset Voltage bits (VBOR set to 1.90 V nominal)

; CONFIG2H
  CONFIG  WDTEN = OFF          ; Watchdog Timer Enable bits (Watch dog timer is always disabled. SWDTEN has no effect.)
  CONFIG  WDTPS = 32768        ; Watchdog Timer Postscale Select bits (1:32768)

; CONFIG3H
  CONFIG  CCP2MX = PORTC1      ; CCP2 MUX bit (CCP2 input/output is multiplexed with RC1)
  CONFIG  PBADEN = OFF         ; PORTB A/D Enable bit (PORTB<5:0> pins are configured as digital I/O on Reset)
  CONFIG  CCP3MX = PORTB5      ; P3A/CCP3 Mux bit (P3A/CCP3 input/output is multiplexed with RB5)
  CONFIG  HFOFST = ON          ; HFINTOSC Fast Start-up (HFINTOSC output and ready status are not delayed by the oscillator stable status)
  CONFIG  T3CMX = PORTC0       ; Timer3 Clock input mux bit (T3CKI is on RC0)
  CONFIG  P2BMX = PORTD2       ; ECCP2 B output mux bit (P2B is on RD2)
```

```
  CONFIG  MCLRE = EXTMCLR     ; MCLR Pin Enable bit (MCLR pin enabled, RE3 input pin disabled
; CONFIG4L
  CONFIG  STVREN = ON         ; Stack Full/Underflow Reset Enable bit (Stack full/underflow will cause Reset)
  CONFIG  LVP = OFF           ; Single-Supply ICSP Enable bit (Single-Supply ICSP enabled if MCLRE is also 1)
  CONFIG  XINST = OFF         ; Extended Instruction Set Enable bit (Instruction set extension and Indexed Addressing mode disabled
(Legacy mode))

; CONFIG5L
  CONFIG  CP0 = OFF           ; Code Protection Block 0 (Block 0 (000800-001FFFh) not code-protected)
  CONFIG  CP1 = OFF           ; Code Protection Block 1 (Block 1 (002000-003FFFh) not code-protected)
  CONFIG  CP2 = OFF           ; Code Protection Block 2 (Block 2 (004000-005FFFh) not code-protected)
  CONFIG  CP3 = OFF           ; Code Protection Block 3 (Block 3 (006000-007FFFh) not code-protected)

; CONFIG5H
  CONFIG  CPB = OFF           ; Boot Block Code Protection bit (Boot block (000000-0007FFh) not code-protected)
  CONFIG  CPD = OFF           ; Data EEPROM Code Protection bit (Data EEPROM not code-protected)

; CONFIG6L
  CONFIG  WRT0 = OFF          ; Write Protection Block 0 (Block 0 (000800-001FFFh) not write-protected)
  CONFIG  WRT1 = OFF          ; Write Protection Block 1 (Block 1 (002000-003FFFh) not write-protected)
  CONFIG  WRT2 = OFF          ; Write Protection Block 2 (Block 2 (004000-005FFFh) not write-protected)
  CONFIG  WRT3 = OFF          ; Write Protection Block 3 (Block 3 (006000-007FFFh) not write-protected)

; CONFIG6H
  CONFIG  WRTC = OFF          ; Configuration Register Write Protection bit (Configuration registers (300000-3000FFh) not write-
protected)
  CONFIG  WRTB = OFF          ; Boot Block Write Protection bit (Boot Block (000000-0007FFh) not write-protected)
  CONFIG  WRTD = OFF          ; Data EEPROM Write Protection bit (Data EEPROM not write-protected)

; CONFIG7L
  CONFIG  EBTR0 = OFF         ; Table Read Protection Block 0 (Block 0 (000800-001FFFh) not protected from table reads executed in
other blocks)
  CONFIG  EBTR1 = OFF         ; Table Read Protection Block 1 (Block 1 (002000-003FFFh) not protected from table reads executed in
other blocks)
  CONFIG  EBTR2 = OFF         ; Table Read Protection Block 2 (Block 2 (004000-005FFFh) not protected from table reads executed in
other blocks)
  CONFIG  EBTR3 = OFF         ; Table Read Protection Block 3 (Block 3 (006000-007FFFh) not protected from table reads executed in
other blocks)

; CONFIG7H
  CONFIG  EBTRB = OFF
```

```
;*****************************************************************************************************
; BLOQUE DECLARACIÓN DE ÁLIAS
;*****************************************************************************************************
 CBLOCK 0X00                ; Definir bloque de constantes en la SRAM
                aux         ; Guardará el número de veces que el valor tomado es correcto
                count       ; Guarda el valor de la cuenta para el Delay_5ms
                yfile       ; Lleva la cuenta de la coordenada de 'Y' en la pantalla
                xfile       ; Lleva la cuenta de la coordenada de 'X' en la pantalla
                valor       ; Guardar el valor de la pulsación en la interrupción
                pixel       ; Se irá volcando el resultado de la interpolación para hallar la posición seleccionada
                mayor       ; Guardará el resultado de 'pixel' una vez estoy seguro de que ha finalizado la interpolación

                orden       ; se usará para discernir entre realizar unas operaciones u otras en las subrutinas
                vresta      ; Guardará el valor necesario para las restas sucesivas de la división
                valorX      ; Valor correcto de la pulsación en el eje X
                valorY      ; Valor correcto de la pulsación en el eje Y
                puertod     ; Guardará el valor de la lectura de la RAM del controlador de la pantalla
                eleX        ; Registro en el que vuelco 'valorX' para operarlo en una subrutina
                eleY        ; Registro en el que vuelco 'valorY' para operarlo en una subrutina
                Dato        ; Registro necesario para obtener el valor de un número aleatorio
                Conta       ; Registro necesario para obtener el valor de un número aleatorio
                coorX       ; Variable usada para comprobar la pulsación del eje X
                coorY       ; Variable usada para comprobar la pulsación del eje Y
                maxX        ; Variable usada para comprobar el valor máximo de la seleccionen eje X
                maxY        ; Variable usada para comprobar el valor máximo de la seleccionen eje Y
                puntua0     ; Lleva la puntuación del primer número del marcador
                puntua1     ; Lleva la puntuación del segundo número del marcador
                maximoX     ; Guardará el valor máximo de X que se mostrará en la pantalla
                maximoY     ; Guardará el valor máximo de Y que se mostrará en la pantalla
                xreferencia ; Valor de refresco de X cada vez que se cambia de página
                restarvid    ; Lleva la cuenta de las "vidas" del juego
                optopo      ; Contiene el número de "topo" que se mostrará
                raiz        ; Variable que contendrá la semilla de la cuenta aleatoria

                ENDC        ; final del bloque de declaración de alias
;*****************************************************************************************************

;*****************************************************************************************************
; INICIO DEL PROGRAMA
; *****************************************************************************************************
```

```
 ORG 0x0000                    ; Directiva para definir el origen del código en dirección 0x000, aquí apuntará el PC
 GOTO MAIN
 ORG 0x0008                    ; Vector de interrupción
        BTFSC PIR1,TMR1IF      ; Si TMR1IF = 0 -> PC +4...Si TMR1IF =1 -> PC +2
        GOTO REFRESCONUMERO    ; Salta a rutina REFRESCONUMERO      PC+2
        GOTO REFRESCOTOPO      ; Salta a rutina REFRESCOTOPO        PC+4
        RETFIE                 ; Vuelve a la posición donde se produjo la interrupción


 ORG 0x0020                    ; Comienzan las líneas del programa
S_PUNTOS                       ; Subrutina encargada de indicar el pixel de la pantalla
        MOVLW .2               ; Cargo 2 en el registro de trabajo 'WREG'
        MULWF valorY           ; Multiplica 'valorY' por el contenido de 'WREG'
        MOVF PRODL,0           ; Carga el valor del registro del resultado en WREG
        ADDWF PCL              ; Se suma el WREG al PCL (Posición mas baja del PC)
         RETLW   b'10000000'   ; Devuelvo el equivalente al bit 0 en la pantalla
         RETLW   b'01000000'   ; Devuelvo el equivalente al bit 1 en la pantalla
         RETLW   b'00100000'   ; Devuelvo el equivalente al bit 2 en la pantalla
         RETLW   b'00010000'   ; Devuelvo el equivalente al bit 3 en la pantalla
         RETLW   b'00001000'   ; Devuelvo el equivalente al bit 4 en la pantalla
         RETLW   b'00000100'   ; Devuelvo el equivalente al bit 5 en la pantalla
         RETLW   b'00000010'   ; Devuelvo el equivalente al bit 6 en la pantalla
        RETLW   b'00000001'    ; Devuelvo el equivalente al bit 7 en la pantalla

MAIN:                          ;Código del programa principal, empieza con las configuraciones
        MOVLB 0xF              ; Sitúa el puntero de direcciones de la SRAM en la posición de las SFR

;*************************************************************************************************************
; CONFIGURAMOS EL PUERTO B D Y C COMO SALIDAS
;*************************************************************************************************************
        CLRF TRISB             ; Se configura este puerto como salida digital
        CLRF ANSELB            ; Se limpia para evitar conflictos
        CLRF PORTB             ; Limpiamos cualquier posible estado previo del puerto

         CLRF TRISD            ; Se configura este puerto como salida digital
         CLRF ANSELD           ; Se limpia para evitar conflictos
        CLRF PORTD             ; Limpiamos cualquier posible estado previo del puerto
```

```
        CLRF TRISC              ; Se configura este puerto como salida digital
        CLRF ANSELC             ; Se limpia para evitar conflictos
        CLRF PORTC              ; Limpiamos cualquier posible estado previo del puerto

        MOVLW 0x02              ; Cargamos el valor 2 en WREG
        MOVWF PORTC             ; Se vuelca en el puertoC seteando la salida RC1 (lectura eje Y)
;*******************************************************************************************************

;*************************************************************************
; CONFIGURAMOS EL PUERTO A0 Y A1 COMO ENTRADA ANALÓGICA, EL RESTO SALIDAS DIGITALES
;*************************************************************************
        MOVLW 0x03              ; Cargamos el valor correspondiente a los dos primeros pines de A
        MOVWF TRISA             ; Las configuramos como entradas
        MOVWF ANSELA            ; Las configuramos como analógicas
        CLRF PORTA              ; Limpiamos cualquier posible estado previo del puerto

        CLRF ADCON1             ; Se configura con las referencias Vdd y Vss
        MOVLW b'00001010'       ; Fosc/32..4 TAD..justificación a la derecha cargándolo en WREG
        MOVWF ADCON2            ; Se pasa el valor correspondiente a la configuración para ADCON2
;*******************************************************************************************************

;*******************************************************************************************************
; TEMPORIZADOR TMR0 CON INTERRUPCIÓN
;*******************************************************************************************************
        MOVLW b'00000100'       ; 16 bits/reloj interno/prescaler 1:32
        MOVWF T0CON             ; Se carga esta configuración en el registro

        MOVLW b'01100000'       ; Interrupciones periféricas/ interrupción por TMR0
        MOVWF INTCON            ; Se carga esta configuración en el registro
        MOVLW b'10000000'       ; PortB pull up disabled
        MOVWF INTCON2           ; Se carga esta configuración en el registro   *****
;*******************************************************************************************************

;*******************************************************************************************************
; TEMPORIZADOR TMR1 CON INTERRUPCIÓN
;*******************************************************************************************************
        CLRF PIR1               ; Se limpia la posible bandera activada de TMR1
        MOVLW b'01000000'       ; Reloj de sistema/ sin prescaler
        MOVWF T1CON             ; Se carga esta configuración en el registro
```

```
        BSF PIE1, TMR1IE            ; Se setea el bit correspondiente a la interrupción por TMR1
;*****************************************************************************************************
        MOVLW .105                  ; Valor correspondiente necesario para la interpolación del valor de X
        MOVWF vresta                ; Se carga en el registro que irá aplicándolo cíclicamente en la división

;*********************************************LIMPIAMOS LOS REGISTROS A UTILIZAR**********************
        CLRF aux
        CLRF valorX
        CLRF valorY
        CLRF valor
        CLRF mayor
        CLRF pixel
        CLRF orden
        CLRF puertod
        CLRF optopo
         CLRF raiz
        CLRF Dato
        CLRF Conta
;*****************************************************************************************************

INICIAR:                            ; Comienza las rutinas y subrutinas del programa

; ************************ENCENDER LA PANTALLA*****************************************
        BSF LATB,5                  ; Es necesario para que la GLCD pueda recibir instrucciones
        MOVLW 0x3F                  ; Con esta instrucción enciendo la pantalla -> 00111111
        MOVWF LATD                  ; Se carga en el bus de datos para ser enviada
        CALL S_IZQ
        CALL S_INSTRUCTION
        CALL S_DER
        CALL S_INSTRUCTION
;************************LIMPIAR LA PANTALLA*****************************************
        BCF orden,0                 ; Cuando el bit0 lo pongo a cero indico que quiero limpiar la pantalla
        CALL S_RESETY
        CALL S_RESETX
        CALL S_IZQ
        CALL S_PANTALLA
;************************MOSTRAR PORTADA*****************************************
        BSF orden,0                 ; Cuando el bit0 lo pongo a 1 indico que quiero enviar una tabla
        CALL S_RESETY
```

```
        CALL S_RESETX
        CALL S_PUNTEROPORTADA
        CALL S_IZQ
        CALL S_PANTALLA
        CALL S_LECTURA
        MOVFF valorX,raiz
;**********************MOSTRAR EL MENÚ*****************************************************
MENU:                                   ; Parte correspondiente al primer menú de selección en la pantalla
        CALL S_NUEVALECTURA
        CALL S_RESETY
        CALL S_RESETX
        CALL S_PUNTEROMENU
        CALL S_IZQ
        CALL S_PANTALLA
        CALL S_LECTURA
        GOTO ELECCIONMENU               ; Salto a la rutina de elección de las opciones del menú

ELECCIONMENU:                           ; Rutina encargada de obtener qué valor del menú se escogió

        MOVFF valorY,eleY               ; Traslado el valor final de la pulsación a eleY
        MOVLW .11                       ; Valor mínimo de 'Y' que ha de tener
        SUBWF eleY                      ; Se resta al valor de la pulsación
        BTFSS STATUS, C                 ; Si C = 1 -> PC+4 (valorY>11)..Si C=0 -> PC+2 (valorY<11)
        GOTO MENU                       ; Repito la medida porque el valorY no es de una opción del menú
        MOVLW .23                       ; Valor máximo de eleY para que este dentro de una opción
        CPFSLT eleY                     ; eley < 23 -> PC+4...eley>=23 -> PC+2
        GOTO MENU                       ; Repito la medida porque el valorY no es de una opción
        MOVFF valorX,eleX               ; Traslado el valor final de la pulsación a eleX
        BCF eleX,6                      ; Elimino el último valor tomando para que la operación sea simétrica
        MOVLW .8                        ; Valor mínimo de 'X' que ha de tener
        SUBWF eleX                      ; Se resta al valor de la pulsación
        BTFSS STATUS, C                 ; Si C = 1 -> PC+4 (valorX>8)..Si C=0 -> PC+2 (valorX<8)
        GOTO MENU                       ; Repito la medida porque el valorY no es de una opción del menú
        MOVLW .47                       ; Valor máximo de eleX para que este dentro de una opción del menú
        CPFSLT eleX                     ; eleX < 47 -> PC+4...eleX>=47 -> PC+2
        GOTO MENU                       ; Repito la medida porque el valorY no es de una opción
        BTFSS valorX ,6                 ; Al estar en posiciones simétricas decide qué lado de la pantalla se pulsó
        GOTO ESCRITURA ;                ; Lado Izquierdo....opción de Escribir libremente
        GOTO TOPOS                      ; Lado Derecho.....opción de Juego "Dale al topo"
```

```
;*****************************************************************************************************
; OPCIÓN TOPOS
;*****************************************************************************************************

TOPOS:                                  ; Rutina que marca el programa del juego de los topos
        CALL S_NUEVALECTURA
        CLRF puntua0                    ; Limpio registro de puntuaciones
        CLRF puntua1                    ; Limpio registro de puntuaciones
        MOVLW .2                        ; Inicio el contador de las vidas
        MOVWF restarvid
;*********************************MOSTRAR PANTALLA DEL JUEGO*************************************
        BSF orden,0                     ; Indico que quiero cargar una tabla en la pantalla
        CALL S_RESETY
        CALL S_RESETX
        CALL S_PUNTEROJUEGOTOPO
        CALL S_PANTALLA

        CALL S_PUNTEROMENSAJE
        CALL S_PANTALLA
        CALL S_NUEVALECTURA
        CALL S_LECTURA

        CALL S_RESETY
        CALL S_RESETX
        CALL S_PUNTEROJUEGOTOPO
         CALL S_PANTALLA


        CALL S_INICIOCUENTA
        CALL S_NUMEROALEATORIO
        CALL S_MOSTRARTOPO
;*********************************PUNTUACIONES DEL JUEGO*****************************
        GOTO TOCAR                      ; Salta a la rutina que empieza el juego
```

```
TOCAR:                          ; Aquí registra las pulsaciones e inicia el tiempo para mostrar otro topo

        BSF INTCON,7            ; Activa las interrupciones del programa
        BSF T0CON,TMR0ON        ; Activa el TMR0 ( Interrupción = Mostrar otro topo)
        CALL S_NUEVALECTURA
        CALL S_LECTURA
        BCF INTCON,7            ; Desactiva las interrupciones del programa
        BCF T0CON,TMR0ON        ; Desactiva el TMR0
        GOTO ELECCIONTOPO       ; Salta a la rutina que discierne que se ha pulsado en la pantalla

ELECCIONTOPO:                   ; Se discierne si se ha pulsado un topo o no

        MOVLW .13               ; Valor que correspondería a las opciones de menú del juego
        CPFSLT valorY           ; 'valorY' < 13 -> PC+4...'valorY'>=13 -> PC+2
        GOTO Y                  ; No ha sido una opción del menú del juego
        GOTO OPCIONES           ; Si lo ha sido y cargo los nuevos valores de comparación
OPCIONES:                       ; Rutina donde se cargan los valores de comparación del menú del juego
        MOVLW .3
        MOVWF coorY
        MOVLW .71
        MOVWF coorX
        MOVLW .51
        MOVWF maxX
        GOTO Y

Y:                              ; Mismo funcionamiento que en ELECCIONMENU solo que con variables

        MOVFF valorY,eleY
        MOVF coorY,W            ; Valor de la variable coorY al WREG
        SUBWF eleY
        BTFSS STATUS, C
        GOTO VIDAS              ; El usuario ha fallado, se salta a la rutina para restar vidas
        MOVF maxY,w             ; Valor de la variable maxY al WREG
        CPFSLT eleY
        GOTO VIDAS              ; El usuario ha fallado, se salta a la rutina para restar vidas
        GOTO X
X:
        MOVFF valorX,eleX
        MOVF coorX,W            ; Valor de la variable coorX al WREG
```

```
        SUBWF eleX
        BTFSS STATUS, C
        GOTO VIDAS              ; El usuario ha fallado, se salta a la rutina para restar vidas
        MOVF maxX,w             ; Valor de la variable MaxX al WREG
        CPFSLT eleX
        GOTO VIDAS              ; El usuario ha fallado, se salta a la rutina para restar vidas
        GOTO RESULTADO          ; La pulsación es correcta, se salta a comprobar que se pulsó

RESULTADO:                      ; Se comprueba si es una opción del menú o un topo
        MOVLW .13               ; Si el valor de Y es menor de 13 es una opción del menú
        CPFSLT valorY           ; 'valorY' < 13 -> PC+4...'valorY'>=13 -> PC+2
        GOTO MARCADOR           ; Se salta a la rutina del marcador
        MOVLW .105              ; Cual de las dos se pulsó
        CPFSLT valorX           ; 'valorX' < 105 -> PC+4...'valorX'>=105 -> PC+2
        GOTO MENU               ; Salto a la rutina menú
        GOTO TOPOS              ; Reinicio el programa



MARCADOR:                       ; Rutina que lleva la cuenta del marcador del juego
        BSF orden,0             ; Se ha de mostrar una tabla
        CALL S_PUNTERONUMERO
        CALL S_PANTALLA
        MOVLW .10               ; Compruebo que la puntuación sea inferior a 10
        CPFSLT puntua0          ; 'puntua0' < 10 -> PC+4...'puntua0'>=10 -> PC+2
        GOTO MARCADOR           ; Vuelve a saltar al marcador para modificar el otro digito
        MOVLW .1                ; Marcador de victoria
        CPFSLT puntua1          ;'puntua1' < 1 -> PC+4...'puntua1'>=1 -> PC+2
        GOTO YOUWIN             ; Rutina que da la Victoria
        GOTO TOCAR              ; Salta a tomar otra pulsación

VIDAS:                          ; Rutina que lleva la cuenta de los errores
        BCF orden,0             ; Mostrará una tabla
        DECF restarvid          ; resto una vida
        BTFSS   STATUS, C       ; Si C = 1 -> PC+4 (restarvid>0)..Si C=0 -> PC+2 (restarvid=<0)
        GOTO GAMEOVER           ; Salto a la rutina de partida perdida
        CALL S_PUNTERONUMERO
        CALL S_PANTALLA
        GOTO TOCAR              ; Salta a tomar otra pulsación
```

```
YOUWIN:                         ; Rutina que sale al final del juego si ganas
        BSF orden,0             ; Muestra una pantalla
        CALL S_PUNTEROYOUWIN
        CALL S_PANTALLA
        CALL S_NUEVALECTURA
        CALL S_LECTURA
        Goto TOPOS              ; Salta a la rutina del menú principal

GAMEOVER:                       ; Rutina que sale al final del juego si pierdes
        BSF orden,0             ; Muestra una tabla
        CALL S_PUNTEROGAMEOVER
        CALL S_PANTALLA
        CALL S_NUEVALECTURA
        CALL S_LECTURA
        GOTO TOPOS              ; Salta a la rutina del menú principal
;*****************************************************************************************************

;*****************************************************************************************************
; OPCIÓN DE ESCRITURA
;*****************************************************************************************************
ESCRITURA:
        CALL S_NUEVALECTURA
        BSF orden,0             ; Mostrar una tabla en pantalla
        CALL S_RESETY
        CALL S_RESETX
        CALL S_PUNTEROHOJADIBUJO
        CALL S_IZQ
        CALL S_PANTALLA
        GOTO IMPRIMIRoMENU

IMPRIMIRoMENU:                  ; Rutina en la que modifico las variables dependiendo del pixel pulsado
        CALL S_LECTURA
        MOVLW .14               ; Valor de Y que indica si se quiere escribir o seleccionar una opción
        CPFSGT valorY           ;'valorY' > 14 -> PC+4...'valorY'<=14 -> PC+2
        GOTO VALMENU
        GOTO VALESCRIBIR

VALMENU:                        ; Rutina que cargará los valores para la elección del menú de esta pantalla
```

```
        MOVLW .3                ; Valor mínimo de Y
        MOVWF coorY
        MOVLW .8                ; Valor máximo de Y
        MOVWF maxY
        BTFSC valorX ,6         ; Si valorX = 0 -> PC+4 Si valorX=1 -> PC+2
        GOTO VOLVER
        GOTO LIMPIEZA

VOLVER:                         ; Rutina que carga los valores correspondientes a esta opción
        MOVLW .97               ; Valor mínimo de X
        MOVWF coorX
        MOVLW .28
        MOVWF maxX              ; Valor máximo de X
        GOTO ELECCION

LIMPIEZA:                       ; Rutina que carga los valores correspondientes a esta opción
        MOVLW .2                ; Valor mínimo de X
        MOVWF coorX
        MOVLW .32               ; Valor máximo de X
        MOVWF maxX
        GOTO ELECCION

VALESCRIBIR:                    ; Rutina que carga los valores que delimitan la zona de dibujo
        MOVLW .14               ; Valor mínimo de Y
        MOVWF coorY
        MOVLW .47               ; Valor máximo de Y
        MOVWF maxY

        MOVLW .2                ; Valor mínimo de X
        MOVWF coorX
        MOVLW .123              ; Valor máximo de X
        MOVWF maxX
        GOTO ELECCION

ELECCION:                       ; Rutina que compara los valores obtenidos y actúa en consecuencia
        MOVFF valorY,eleY
        MOVF coorY,W            ; Valor de la variable coorY al WREG
        SUBWF eleY
        BTFSS STATUS, C
```

```
        GOTO IMPRIMIRoMENU          ; Vuelvo a tomar otro valor
        MOVF maxY,w                 ; Valor de la variable maxY al WREG
        CPFSLT eleY
        GOTO IMPRIMIRoMENU          ; Vuelvo a tomar otro valor

        MOVFF valorX,eleX
        MOVF coorX,W                ; Valor de la variable coorX al WREG
        SUBWF eleX
        BTFSS STATUS, C
        GOTO IMPRIMIRoMENU          ; Vuelvo a tomar otro valor
        MOVF maxX,w                 ; Valor de la variable MaxX al WREG
        CPFSLT eleX
        GOTO IMPRIMIRoMENU          ; Vuelvo a tomar otro valor
        GOTO RESULTADOIMPoMEN       ; La pulsación es correcta, se salta a comprobar que se pulsó

RESULTADOIMPoMEN:
        MOVLW .14                   ; Indicará si se trata de una opción del menú
        CPFSLT valorY               ;'valorY' < 14 -> PC+4...'valorY'>=14 -> PC+2
        GOTO IMPRIMIR
        BTFSS valorX ,6             ; Si valorX = 1 -> PC+4 Si valorX=0 -> PC+2
        GOTO ESCRITURA
        GOTO MENU

IMPRIMIR:                           ; Rutina que lee y muestra los valores de las pulsaciones
        BTFSS valorX ,6             ; Si valorX = 1 -> PC+4 Si valorX=0 -> PC+2
        CALL S_IZQ
        BTFSC valorX,6              ; Si valorX = 0 -> PC+4 Si valorX=1 -> PC+2
        CALL S_DER
        CALL S_POSICION
        GOTO IMPRIMIRoMENU          ; Vuelve a tomar otro valor
;*****************************************************************************************************

;*****************************************************************************************************
; RUTINAS DE INTERRUPCIÓN
;*****************************************************************************************************
REFRESCOTOPO:                       ; Rutina interrupción provocada por TMR0
        BCF INTCON,7                ; Desactivo interrupciones
        BCF INTCON, TMR0IF          ; Desactivo TMR0
        BCF orden,0                 ; orden =0 da la señal para borrar el topo mostrado
```

```
        CALL S_MOSTRARTOPO
        BSF orden,0                 ; orden =1 da la señal para mostrar un nuevo topo
        CALL S_NUMEROALEATORIO
        CALL S_MOSTRARTOPO
        BSF INTCON,7                ; Activo solo las interrupciones
        RETFIE                      ; Vuelvo a la posición donde se produjo la interrupción

REFRESCONUMERO:                     ; Rutina interrupción provocada por TMR1
        BCF INTCON,7                ; Desactivo interrupciones
        BCF PIR1, TMR1IF            ; Desactivo TMR1
        MOVFF mayor, valor          ; Se usará 'valor' para comprobar si es el mismo valor en la sig pulsación
        BSF INTCON,7                ; Activo solo las interrupciones
        RETFIE                      ; Vuelvo a la posición donde se produjo la interrupción
;*************************************************************************************************

; ************************************************************************************************
; SUBRUTINAS
; ************************************************************************************************

S_NUEVALECTURA                      ; Subrutina que indica si se ha pulsado la pantalla
        MOVLW 0x01                  ; Puerto RC0 (coordenada X)
        MOVWF PORTC
        MOVLW 0x03                  ; ADC canal -> AN0
        MOVWF ADCON0
        CALL CONVERSION             ; Tomo valor
        MOVLW .18                   ; Restar valor de calibrado
        SUBWF ADRESH
        BTFSS STATUS, C             ; Si C = 1 -> PC+4 (ADRESH>18)..Si C=0 -> PC+2 (ADRESH=<18)
        RETURN                      ; Vuelvo a la rutina principal
        GOTO S_NUEVALECTURA         ; Continua en bucle hasta pulsación en la pantalla

S_MOSTRARTOPO                       ; Subrutina que muestra un topo en la pantalla
        CALL S_PUNTEROTOPO
        CALL S_PANTALLA
        RETURN

S_INICIOCUENTA                      ; Subrutina que reinicia el valor del topo actual
        CLRF optopo                 ; Limpia cualquier posible valor
        BSF optopo,0                ; Marca la posición del topo 0
```

```
        RETURN

S_NUMEROALEATORIO               ; Subrutina que genera un valor aleatorio
        MOVFF raiz,Dato
        CALL RANDOM
        MOVFF Dato,raiz
        MOVF optopo,W

SELECCIONTOPO:

        RLNCF optopo
        BTFSC optopo,5
        CALL S_INICIOCUENTA
        DECF Dato
        BTFSS   STATUS, Z
        GOTO SELECCIONTOPO
        CPFSEQ optopo
        RETURN
        GOTO S_NUMEROALEATORIO
RANDOM
        BTFSC   Dato, 4
        INCF   Conta, f
        BTFSC   Dato, 3
        INCF   Conta, f
        BTFSC   Dato, 2
        INCF   Conta, f
        BTFSC   Dato, 0
        INCF   Conta, f
        RRCF   Dato, f
        BSF   Dato, 7
        BTFSS   Conta, 0
        BCF   Dato, 7
        CLRF   Conta
        RETURN

S_LECTURA                       ; Subrutina que realiza la lectura de los valores que devuelve el panel táctil
         MOVLW 0x01
        MOVWF LATC              ; Puerto RC0 (coordenada X)
READ:
```

```
        CLRF pixel                  ; Limpiamos el valor donde se hará el cálculo
        BTFSC PORTC,0               ; Si RC0 = 0 -> PC+4....Si RC0=1 -> PC+2
        MOVLW 0x03                  ; Canal del ADC AN0
        BTFSC PORTC,1               ; Si RC1 = 0 -> PC+4....Si RC1=1 -> PC+2
        MOVLW 0x07                  ; Canal del ADC AN1
        MOVWF ADCON0
        BCF INTCON,7                ; Se paran las interrupciones
        CALL CONVERSION
        BSF INTCON,7                ; Activo las interrupciones
        MOVLW .18                   ; Valor que fija la primera pulsación dentro de la pantalla
        SUBWF ADRESH                ; Se le resta al obtenido en la conversión
        BTFSS STATUS, C             ; Si C = 1 -> PC+4 (ADRESH>18)..Si C=0 -> PC+2 (ADRESH=<18)
        GOTO BORRAR
        BSF T1CON,TMR1ON            ; Empieza el TMR1 a realizar la cuenta para guardar un valor
        GOTO X64
BORRAR:                             ; Rutina que vuelve a empezar a tomar la medida por X
        BTFSS PORTC,0               ; Si PORTC = 1 -> PC+4..Si PORTC=0 -> PC+2
        CALL S_XOY
        GOTO READ                   ; Vuelvo a empezar con la medida
X64:                                ; Rutina que empieza la interpolación para hallar el pixel se pulsó
        MOVLW .64                   ; pongo en wreg el valor 64
        MULWF ADRESH                ; lo multiplico por el número puesto en el registro valor
        TSTFSZ ADRESH               ; Si ADRESH = 0 -> PC+4....Si ADRESH =/ 0 -> PC+2
        GOTO DIVISION
        GOTO COMPARO

DIVISION:                           ; Rutina donde se realiza la división a través de restas sucesivas

        INCF pixel                  ; Incrementamos en 1 la cuenta
        MOVF vresta,0               ; Le resto el valor dependiendo del eje X o Y
        SUBWF PRODL                 ; Se lo resto al menor registro
        BTFSC STATUS, C              ; Si C = 0 -> PC+4 (PRODL<0)....Si C=1 ->(PRODL>=0) PC+2
        GOTO DIVISION                ; Si no hay desbordamiento volvemos a incrementar y a restar 105
        DECF PRODH                  ; Decremento el PRODH
        BTFSS STATUS, N              ; Si N = 1 -> PC+4 (PRODH<0)..Si N=0 -> PC+2 (PRODH=>0)
        GOTO DIVISION               ; Incremento y vuelvo a realizar el proceso
        DECF pixel                  ; Elimino el pixel que sobra de la cuenta
        GOTO COMPARO                ; En caso de que si...procedo a comparar
```

```
COMPARO:
        BTFSC PORTC,0                   ; Si RC0 = 0 -> PC+4....Si RC0=1 -> PC+2
        MOVLW .128
        BTFSC PORTC,1                   ; Si RC1 = 0 -> PC+4....Si RC1=1 -> PC+2
        MOVLW .64
        CPFSLT pixel                    ; 'puntua0' < WREG -> PC+4...'puntua0'>= WREG -> PC+2
        GOTO READ
        MOVFF pixel, mayor              ; Se guarda en un registro como valor final de la operación
        MOVF valor,0                    ; Se carga el valor que se guardó en la interrupción anterior
        CPFSEQ mayor                    ; 'mayor' = valor -> PC+4...'mayor'=/ valor -> PC+2
        GOTO READ
        INCF aux                        ; Se va incrementando una cuenta con las comparaciones correctas
        MOVLW .5                        ; Tendrá que haber 5 comparaciones correctas
        CPFSEQ aux                      ; 'aux' = 5 -> PC+4...'mayor'=/ 5 -> PC+2
        GOTO ANTERIOR
        BTFSC PORTC,0                   ; Si RC0 = 0 -> PC+4....Si RC0=1 -> PC+2
        GOTO GUARDOX
        GOTO GUARDOY

ANTERIOR:               ; Rutina en la que se comprobará si la pulsación pertenece al adyacente del pixel anterior
        BTFSC PORTC,0                   ; Si RC0 = 0 -> PC+4....Si RC0=1 -> PC+2
        MOVF valorX, 0                  ; Dependiendo de la coordenada vuelco en WREG un valor u otro
        BTFSC PORTC,1                   ; Si RC1 = 0 -> PC+4....Si RC1=1 -> PC+2
        MOVF valorY, 0
        ADDLW -1                        ; Lo decremento en 1
        CPFSEQ mayor                    ; 'mayor' = valor-1 -> PC+4...'mayor'=/ valor-1 -> PC+2
        GOTO POSTERIOR                  ; En caso de no ser paso a comprobar el valor posterior
        BTFSC PORTC,0                   ; Si RC0 = 0 -> PC+4....Si RC0=1 -> PC+2
        GOTO GUARDOX
        GOTO GUARDOY

POSTERIOR:              ; Rutina en la que se comprobará si la pulsación pertenece al adyacente del pixel posterior
        BTFSC PORTC,0                   ; Si RC0 = 0 -> PC+4....Si RC0=1 -> PC+2
        MOVF valorX, 0
        BTFSC PORTC,1                   ; Si RC1 = 0 -> PC+4....Si RC1=1 -> PC+2
        MOVF valorY, 0
        ADDLW +1                        ; Le incremento en 1
        CPFSEQ mayor                    ; 'mayor' = valor+1 -> PC+4...'mayor'=/ valor+1 -> PC+2
        GOTO READ
```

```
        BTFSC PORTC,0           ; Si RC0 = 0 -> PC+4....Si RC0=1 -> PC+2
        GOTO GUARDOX
        GOTO GUARDOY

GUARDOX:                        ; Rutina donde se guarda el valor de X en la pulsación al darlo por correcto
        CLRF aux                ; Limpio la cuenta de comparaciones correctas
        MOVFF mayor, valorX     ; Pasa al registro que se usará para enviarlo a la pantalla
        BCF T1CON,TMR1ON        ; Para el TMR1
        BCF PIR1, TMR1IF        ; Limpio la bandera
        CLRF valor              ; Elimina el valor guardado con las interrupciones
        CALL S_XOY              ; Cambia a la lectura del ejeY
        GOTO READ

GUARDOY:                        ; Rutina donde se guarda el valor de Y en la pulsación al darlo por correcto
        CLRF aux                ; limpio la cuenta de comparaciones correctas
        MOVFF mayor, valorY     ; Pasa al registro que se usará para enviarlo a la pantalla
        BCF T1CON,TMR1ON        ; Para el TMR1
        BCF PIR1, TMR1IF        ; Limpio la bandera
        BCF INTCON,7            ; Desactivo las interrupciones
        CLRF valor              ; Elimina el valor guardado con las interrupciones
        CALL S_XOY              ; Cambia a la lectura del ejeX

        RETURN

ESCUCHAR                        ; Subrutina que realiza una espera hasta un evento

        BTFSC PORTB,7           ; Si RB7= 0 -> PC+4 Si RB7 =1 -> PC+2
        RETURN
        GOTO ESCUCHAR           ; Bucle

S_XOY                           ; Subrutina que alterna la lectura de 'X' e 'Y'
        CLRF valor              ; Limpio el valor guardado tras la interrupción
        BTFSC PORTC,0           ; Si RC0 = 0 -> PC+4 Si RC1 =1 -> PC+2
        GOTO LECTY
        GOTO LECTX

LECTX:                          ; Rutina de Configuración para leer el eje 'X'
        MOVLW 0x01              ; RC0 = 1 RC1/2/3..=0
        MOVWF LATC
```

```
        MOVLW .105              ; Valor necesario para la interpolación con 'X'
        MOVWF vresta
        CALL Delay_5ms          ; Retardo para evitar problemas en la lectura
        RETURN
LECTY:                          ; Rutina de configuración para leer el eje 'Y'
        MOVLW 0x02              ; RC1 = 1 RC0/RC2/RC3..=0
        MOVWF LATC
        MOVLW .88               ; Valor necesario para la interpolación con 'Y'
        MOVWF vresta
        CALL Delay_5ms          ; Retardo para evitar problemas en la lectura
        RETURN

S_PUNTEROPORTADA                ; Subrutina donde indica la posición de la tabla de la portada
        MOVLW UPPER TPORTADA    ; Devuelve el valor de los 5 primeros bits de dirección
        MOVWF TBLPTRU           ; Lo vuelca en el registro correspondiente del puntero
        MOVLW HIGH TPORTADA     ; Devuelve el valor de los 8 siguientes bits de dirección
        MOVWF TBLPTRH
        MOVLW LOW TPORTADA      ; Devuelve el valor de los 8 últimos bits de dirección
        MOVWF TBLPTRL
        RETURN

S_PUNTEROMENU                   ; Subrutina que indica la posición de la tabla del Menú
        MOVLW UPPER TMENU
        MOVWF TBLPTRU
        MOVLW HIGH TMENU
        MOVWF TBLPTRH
        MOVLW LOW TMENU
        MOVWF TBLPTRL
        RETURN

S_PUNTEROHOJADIBUJO             ; Subrutina que indica la posición de la tabla de la hoja de dibujo
        MOVLW UPPER THOJADIBUJO
        MOVWF TBLPTRU
        MOVLW HIGH THOJADIBUJO
        MOVWF TBLPTRH
        MOVLW LOW THOJADIBUJO
        MOVWF TBLPTRL
        RETURN
```

```
S_PUNTEROJUEGOTOPO              ; Subrutina que indica la posición de la tabla del Juego
        MOVLW UPPER TJUEGOTOPO
        MOVWF TBLPTRU
        MOVLW HIGH TJUEGOTOPO
        MOVWF TBLPTRH
        MOVLW LOW TJUEGOTOPO
        MOVWF TBLPTRL
        RETURN

S_PUNTEROTOPO                   ; Subrutina que indica la posición de la tabla del Topo
        MOVLW UPPER TTOPO
        MOVWF TBLPTRU
        MOVLW HIGH TTOPO
        MOVWF TBLPTRH
        MOVLW LOW TTOPO
        MOVWF TBLPTRL

        MOVLW .16               ; Posición en la tabla que permite borrar un topo
        BTFSS orden,0           ; Si orden = .1 -> PC+4 Si orden = .0 -> PC+2
        ADDWF TBLPTRL           ; Modifico el puntero
        BSF orden,0             ; Devuelvo a su configuración para mostrar tablas
        MOVLW .17               ; Valor máximo que tendrá cada 'Topo'
        MOVWF maxX
;*******************VALORES CORRESPONDIENTES A LA PRIMERA ILERA DE TOPOS**************************
        MOVLW .185              ; Página 1 ->  10111001
        MOVWF yfile             ; Llevará la cuenta de la página
        MOVLW .186              ; Página 2 ->  10111010
        MOVWF maximoY           ; Valor máximo de página que puede mostrar la pantalla
        MOVLW .40               ; Valor mínimo de Y que ha de tener la pulsación
        MOVWF coorY

        MOVLW .12               ; Valor máximo de Y que ha de tener la pulsación
        MOVWF maxY

        BTFSC optopo,0          ; Si optopo(0)= 0 -> PC+4 Si optopo(0) = 1 -> PC+2
        GOTO TOPO0
        BTFSC optopo, 1         ; Si optopo(1)= 0 -> PC+4 Si optopo(1) = 1 -> PC+2
```

```
            GOTO TOPO1
            BTFSC optopo, 2                 ; Si optopo(2)= 0 -> PC+4 Si optopo(2) = 1 -> PC+2
            GOTO TOPO2
;*************************VALORES CORRESPONDIENTES A LA SEGUNDA ILERA DE TOPOS***************
            MOVLW .187                      ; Página 3 -> 10111011
            MOVWF yfile
            MOVLW .188                      ; Página 4 -> 10111100
            MOVWF maximoY
            MOVLW .23                       ; Valor mínimo de 'Y' necesario en la pulsación para seleccionarlo
            MOVWF coorY

            BTFSC optopo,3                  ; Si optopo(3)= 0 -> PC+4 Si optopo(3) = 1 -> PC+2
            GOTO TOPO3
            BTFSC optopo,4                  ; Si optopo(4)= 0 -> PC+4 Si optopo(4) = 1 -> PC+2
            GOTO TOPO4

TOPO0:                                      ; Rutina que carga los valores necesarios para el 'Topo0'
            MOVLW .24                       ; Valor de X del que partirá el Topo para mostrarse
            MOVWF xfile
            MOVWF xreferencia
            MOVLW .31                       ; Valor máximo de 'X' que se mostrará en la pantalla
            MOVWF maximoX
            MOVLW .19                       ; Valor mínimo de 'X' necesario en la pulsación para seleccionarlo
            MOVWF coorX
            RETURN

TOPO1:                                      ; Rutina que carga los valores necesarios para el 'Topo1'
            MOVLW .60                       ; Valor de X del que partirá el Topo para mostrarse
            MOVWF xfile
            MOVWF xreferencia
            MOVLW .67                       ; Valor máximo de 'X' que se mostrará en la pantalla
            MOVWF maximoX
            MOVLW .55                       ; Valor mínimo de 'X' necesario en la pulsación para seleccionarlo
            MOVWF coorX
            RETURN
TOPO2:                                      ; Rutina que carga los valores necesarios para el 'Topo2'
            MOVLW .96                       ; Valor de X del que partirá el Topo para mostrarse
            MOVWF xfile
            MOVWF xreferencia
```

```
        MOVLW .103              ; Valor máximo de 'X' que se mostrará en la pantalla
        MOVWF maximoX
        MOVLW .91               ; Valor mínimo de 'X' necesario en la pulsación para seleccionarlo
        MOVWF coorX
        RETURN
TOPO3:                          ; Rutina que carga los valores necesarios para el 'Topo3'
        MOVLW .42               ; Valor de X del que partirá el Topo para mostrarse
        MOVWF xfile
        MOVWF xreferencia
        MOVLW .49               ; Valor máximo de 'X' que se mostrará en la pantalla
        MOVWF maximoX
        MOVLW .36               ; Valor mínimo de 'X' necesario en la pulsación para seleccionarlo
        MOVWF coorX
        RETURN
TOPO4:                          ; Rutina que carga los valores necesarios para el 'Topo4'
        MOVLW .78               ; Valor de X del que partirá el Topo para mostrarse
        MOVWF xfile
        MOVWF xreferencia
        MOVLW .85               ; Valor máximo de 'X' que se mostrará en la pantalla
        MOVWF maximoX
        MOVLW .72               ; Valor mínimo de 'X' necesario en la pulsación para seleccionarlo
        MOVWF coorX
        RETURN

S_PUNTERONUMERO                 ; Subrutina donde indica la posición de la tabla de los números

        MOVLW UPPER TNUMERO
        MOVWF TBLPTRU
        MOVLW HIGH TNUMERO
        MOVWF TBLPTRH
        MOVLW LOW TNUMERO
        MOVWF TBLPTRL

        MOVLW .190              ; Página 6 -> 10111110
        MOVWF yfile
        MOVLW .191              ; Página 7 -> 10111111
        MOVWF maximoY
RESTO:
        BTFSC orden,0           ; Si orden = .0 -> PC+4 Si orden = .1 -> PC+2
```

```
        GOTO PUNTUO
        GOTO TERCERO
PUNTUO:                 ; Rutina donde se decida que número se ve afectado del marcador
        MOVLW .10
        CPFSLT puntua0  ; 'puntua0' < 10 -> PC+4...'puntua0'>=10 -> PC+2
        GOTO SEGUNDO
        GOTO PRIMERO


PRIMERO:                ; Rutina que modifica primer número del marcador
        MOVLW .6        ; Cada número ocupa 6 posiciones en la tabla
        MULWF puntua0   ; Al multiplicarlo por la puntuación se obtiene la posición
        MOVF PRODL, W
        ADDWF TBLPTRL   ; Se modifica el puntero con la dirección correspondiente del número
        INCF puntua0    ; Se aumenta la puntuación del segundo número
        MOVLW .61       ; Valor de X del que partirá el número del marcador
        MOVWF xfile
        MOVWF xreferencia
        MOVLW .63       ; Valor de X máximo del número del marcador
        MOVWF maximoX
        RETURN

SEGUNDO:                ; Rutina que modifica el segundo número del marcador
        SUBWF puntua0   ; Le restamos 10 a puntua0
        MOVLW .6        ; Misma operación que en PRIMERO
        MULWF puntua1
        MOVF PRODL, W
        ADDWF TBLPTRL
        INCF puntua1    ; Se aumenta la puntuación del segundo número
        MOVLW .57       ; Valor de X del que partirá el número del marcador
        MOVWF xfile
        MOVWF xreferencia
        MOVLW .59       ; Valor de X máximo del número del marcador
        MOVWF maximoX
        RETURN

TERCERO:                ; Rutina que modifica el contador de las vidas
        BSF orden,0     ; Devuelvo a su configuración para mostrar tablas
        MOVLW .6        ; Misma operación que en PRIMERO
```

```
        MULWF restarvid
        MOVF PRODL, W
        ADDWF TBLPTRL
        MOVLW .24                   ; Valor de X del que partirá el número de 'Vidas' en pantalla
        MOVWF xfile
        MOVWF xreferencia
        MOVLW .26                   ; Valor de X máximo del número de 'Vidas' en pantalla
        MOVWF maximoX
        RETURN

S_PUNTEROGAMEOVER                   ; Subrutina donde indica la posición de la tabla de juego perdido
        MOVLW UPPER TGAMEOVER
        MOVWF TBLPTRU
        MOVLW HIGH TGAMEOVER
        MOVWF TBLPTRH
        MOVLW LOW TGAMEOVER
        MOVWF TBLPTRL

        MOVLW .185                  ; Página 1 -> 10111001
        MOVWF yfile
        MOVLW .188                  ; Página 2 -> 10111100
        MOVWF maximoY

        MOVLW .25                   ; Valor de X del que partirá del mensaje
        MOVWF xfile
        MOVWF xreferencia
        MOVLW .102                  ; Valor de X máximo del mensaje
        MOVWF maximoX
        RETURN

S_PUNTEROYOUWIN
        MOVLW UPPER TYOUWIN         ;Cargo las direcciones del puntero en el que se encuentra la tabla
        MOVWF TBLPTRU
        MOVLW HIGH TYOUWIN
        MOVWF TBLPTRH
        MOVLW LOW TYOUWIN
        MOVWF TBLPTRL

        MOVLW .185                  ; Página 1 -> 10111001
```

```
        MOVWF yfile
        MOVLW .188              ; Página 2 -> 10111100
        MOVWF maximoY

        MOVLW .25               ; Valor de X del que partirá del mensaje
        MOVWF xfile
        MOVWF xreferencia
        MOVLW .102              ; Valor de X máximo del mensaje
        MOVWF maximoX
        RETURN

S_PUNTEROMENSAJE
        MOVLW UPPER TMENSAJE    ; Cargo las direcciones del puntero en el que se encuentra la tabla
        MOVWF TBLPTRU
        MOVLW HIGH TMENSAJE
        MOVWF TBLPTRH
        MOVLW LOW TMENSAJE
        MOVWF TBLPTRL

        MOVLW .185              ; Página 1 -> 10111001
        MOVWF yfile
        MOVLW .188              ; Página 4 -> 10111100
        MOVWF maximoY

        MOVLW .25               ; Valor de X del que partirá del mensaje
        MOVWF xfile
        MOVWF xreferencia
        MOVLW .102              ; Valor de X máximo del mensaje
        MOVWF maximoX
        RETURN
S_LIMPIAR                       ; Subrutina usada para borrar la memoria de la RAM (Limpiar pantalla)
        MOVLW 0x00
        MOVWF LATD
        RETURN

S_TABLA                         ; Subrutina para leer los valores de la tabla y desplazarse por ella
        TBLRD*+                 ; Lectura de la tabla con incremento de posición
        MOVF TABLAT,W           ; Carga el valor leído en el WREG
        MOVWF PORTD
```

```
        RETURN

S_RESETX                        ; Subrutina donde se reinician los valores de referencia de la tabla (Y)
        MOVLW .127              ; Valor de X máximo a mostrar en pantalla
        MOVWF maximoX
        MOVLW .0                ; Valor de X del que se partirá para mostrar el contenido en la pantalla
        MOVWF xfile
        MOVWF xreferencia
        RETURN

S_RESETY                        ; Subrutina donde se reinician los valores de referencia de la tabla (X)
        MOVLW .184              ; Página 0 -> 10111000
        MOVWF yfile
        MOVLW .191              ; Página 1 -> 10111111
        MOVWF maximoY
        RETURN

;*****************************************************************************************************
; SUBRUTINA QUE CARGA LOS VALORES DE LAS TABLAS EN LA PANTALLA
;*****************************************************************************************************
S_PANTALLA              ; Subrutina que irá cargando los valores en la pantalla dependiendo del puntero elegido
        MOVLW .64               ; Guardamos este valor en el registro para la comparación
        CPFSLT xfile            ; 'xfile' < 64 -> PC+4...'xfile'>=64 -> PC+2
        CALL S_DER              ; Continua con los 64 primeros pixels horizontales
;*********************************************COORDENADA X**********************************
        MOVFF xfile,LATD        ; Indicamos coordenada X de la pantalla
        BSF LATD,6              ; Configuramos el puerto para que entienda que es coordenada X
        CALL S_INSTRUCTION      ; Pasamos a la rutina para cargar una instrucción
;*****************************************************************************************************
;*********************************************COORDENADA Y**********************************
        MOVf yfile,0            ; Indicamos la Página (coordenada Y)
        MOVWF LATD              ; Lo movemos al puerto D
        CALL S_INSTRUCTION      ; Pasamos a la rutina para cargar una instrucción
;*****************************************************************************************************
        BTFSC orden,0           ; Si orden (0) = 0 -> PC+4 Si orden (0) =1 -> PC+2
        CALL S_TABLA            ; En caso de no ser 0 paso a mostrar la tabla elegida
        BTFSS orden,0           ; Si orden (0) = 1 -> PC+4 Si orden (0) =0 -> PC+2
        CALL S_LIMPIAR          ; En caso de no ser 1 voy a la subrutina limpiar
        CALL S_WRITE            ; Vuelco el puerto D en la GLCD
```

```
        INCF xfile
        MOVF maximoX,W
        CPFSGT xfile                    ; 'xfile' > maximoX -> PC+4...'xfile'<=maximoX -> PC+2
        GOTO S_PANTALLA                 ; Bucle hasta completar todas las posiciones de X
        MOVFF xreferencia,xfile
        CALL S_IZQ                      ; paso la pantalla izquierda en caso de haber completado el eje y
        INCF yfile                      ; Incremento el contador de páginas
        MOVF maximoY,W
        CPFSGT yfile                    ; 'yfile' > maximoY -> PC+4...'yfile'<=maximoY -> PC+2
        GOTO S_PANTALLA                 ; Bucle hasta completar todas las posiciones de Y
        RETURN
;*************************************************************************************************************

;*************************************************************************************************************
; SUBRUTINA QUE CARGA LOS VALORES DEL PANEL TÁCTIL EN LA PANTALLA
;*************************************************************************************************************
S_POSICION                              ; indico el pixel que quiero encender
;*****************************SELECCIÓN DE PÁGINA*************************************
        MOVLW 0xB8                      ; selecciono la primera línea de pixels de la pantalla
        MOVWF LATD                      ; Envío la trama a la pantalla
        MOVLW .32                       ; Valor necesario para dividir el registro valorY en dos partes
        MULWF valorY                    ; Quedará el valor de la página en el registro PRODH
        BTG PRODH,0                     ; Invierto las posiciones debido a la configuración inversa de la pantalla
        BTG PRODH,1
        BTG PRODH,2
        MOVF PRODH,0
        ADDWF LATD                      ; Se lo sumo al puerto ya configurado para enviar una página
        CALL S_INSTRUCTION              ; Cargo la instrucción en la pantalla
;*****************************SELECIÓN DE POSICIÓN DE X*******************************
        BCF valorX, 7                   ; Se configura valor X para indicar que es una coordenada de X
        BSF valorX, 6
        MOVFF valorX,LATD               ; Carga el valor en el puerto D
        CALL S_INSTRUCTION              ; Envía el valor al GLCD
        CALL S_READ                     ; Leo en la posición indicada que pixels están encendidos
        MOVFF valorX,LATD               ; Vuelve a cargar la posición de X porque READ la modifica
        CALL S_INSTRUCTION
;*****************************SELECIÓN DEL PIXEL******************************************
        MOVLW 0x07                      ; Valor correspondiente a los 3 primeros bits
        ANDWF valorY                    ; Con la and solo me quedo con el valor de esos 3 bits
```

```
        CALL S_PUNTOS
        IORWF puertod               ; Conociendo el valor en la pos seleccionada sumo el de la pulsación
        MOVFF puertod, LATD         ; Lo cargo en la trama para enviarlo al GLCD
        CALL S_WRITE                ; lo escribo en el byte de la RAM seleccionado
        RETURN
;*************************************************************************************************************

S_IZQ                               ; Subrutina de selección del controlador de la izquierda de la pantalla
        BCF LATB,0                  ; CS1 = 0
        BSF LATB,1                  ; CS2 = 1
        RETURN

S_DER                               ; Subrutina de selección del controlador de la derecha de la pantalla
        BSF LATB,0                  ; CS1 = 1
        BCF LATB,1                  ; CS2 = 0
        RETURN

S_INSTRUCTION                       ; Subrutina que envía la información del puertoD al controlador elegido
                        ; APAGADO/ENCENDIDO...ESPECIFICAR: PÁGINA, COORDENADA X
        CLRF TRISD                  ; Todo el puerto D como SALIDA
        BCF LATB,2                  ; D/I = 0
        BCF LATB,3                  ; R/W = 0
        BSF LATB,4                  ; E = 1
        CALL Delay_10us
        BCF LATB,4                  ; E = 0
        CALL Delay_10us
        RETURN

S_WRITE                             ; Subrutina que carga el puertoD en el Byte de RAM elegido
        CLRF TRISD                  ; Todo el puerto D como SALIDA
        BSF LATB,2                  ; D/I = 1
        BCF LATB,3                  ; R/W = 0
        BSF LATB,4                  ; E = 1
        CALL Delay_10us
        BCF LATB,4                  ; E = 0
        CALL Delay_10us
        RETURN

S_READ                              ; Subrutina de lectura de los datos de la posición de RAM escogida
```

```
        SETF TRISD              ; Todo el puerto D como ENTRADA
        BSF LATB,2              ; D/I = 1
        BSF LATB,3              ; R/W = 1
        BSF LATB,4              ; E = 1
        CALL Delay_10us
        BCF LATB,4              ; E = 0
        CALL Delay_10us
        BSF LATB,4              ; E = 1
        CALL Delay_10us
        MOVFF PORTD, puertod
        BCF LATB,4              ; E = 0
        CALL Delay_10us
        CLRF TRISD              ; Configuro de nuevo el puertoD como salida
        RETURN



CONVERSION                      ; Subrutina que se encarga de realizar la conversión del módulo ADC

        BTFSC ADCON0,1          ; Si ADCON0(1) = 0 -> PC+4...Si ADCON0(1) =1 -> PC+2
        GOTO CONVERSION
        BTFSC PORTC,0           ; Si PORTC(0) = 0 -> PC+4...Si PORTC(0) =1 -> PC+2
        RETURN
        RRNCF ADRESH            ; Como se trata de una medida de Y le quito un bit al resultado
        BCF ADRESH, 7
        RETURN                  ; Volvemos al código principal

Delay_10us                      ; +2 Subrutina de retardo
CUENTA10us:
        MOVLW d'1'              ; Tiempo de ejecución de la instrucción +1
        ADDLW -1                ; +1
        BTFSS STATUS, Z         ; +2
        GOTO CUENTA10us         ; +2
        RETURN                  ; +2

Delay_5ms                       ; Subrutina de retado
        MOVLW d'45'             ; Valor de la cuenta que irá disminuyendo
        MOVWF count
```

```
        GOTO    CUENTA


CUENTA:
        ADDLW   -1                      ; Resto 1
        BTFSS   STATUS, Z               ; Si Z = 1 -> PC+4 (WREG<0)..Si Z=0 -> PC+2 (WREG=>0)
        GOTO    CUENTA
        MOVLW d'45'                     ; vuelvo a cargar el valor
        DECFSZ count, f                 ; Resta 1 y si count = 0 -> PC+4 ..Si count /=0 -> PC+2
        GOTO CUENTA
        RETURN


  ORG 0x1000
TPORTADA
  db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
  db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
  db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xE0, 0xF0, 0xF0, 0xF0, 0xF0, 0x70, 0x00, 0x00, 0x00
  db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xE0, 0xF0, 0xF0, 0xF0, 0xF0, 0x00, 0x00, 0x00, 0xF0, 0xF0
  db 0xF0, 0xF0, 0xF0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xF0, 0xF0, 0xF0
  db 0xF0, 0xF0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
  db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
  db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

  db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
  db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
  db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFE, 0xFF, 0xFF, 0xFF, 0xFF, 0x03, 0x00, 0x00, 0x00, 0x00
  db 0x00, 0x00, 0x00, 0x00, 0x00, 0xF0, 0xFF, 0xFF, 0xFF, 0xFF, 0x1F, 0x00, 0x00, 0x00, 0xFF, 0xFF
  db 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF
  db 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
  db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
  db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

  db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
  db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
  db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1F, 0xFF, 0xFF, 0xFF, 0xFF, 0xF0, 0xC0, 0x00, 0x00, 0x00
  db 0x00, 0x80, 0xE0, 0xF0, 0xFE, 0xFF, 0xFF, 0xFF, 0xFF, 0x7F, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF
  db 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x80, 0x80, 0x80, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF
  db 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x80, 0x80, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
  db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
  db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
```

db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

db 0x00, 0x00, 0x00, 0x00, 0x80, 0x80, 0x80, 0x80, 0x81, 0x87, 0x8F, 0x8F, 0x9F, 0x9F, 0x9E, 0x9E

db 0x9F, 0x9F, 0x8F, 0x8F, 0x9F, 0x9F, 0x9F, 0x9F, 0x9F, 0x80, 0x80, 0x80, 0x80, 0x80, 0x9F, 0x9F

db 0x9F, 0x9F, 0x9F, 0x9F, 0x9F, 0x9F, 0x9F, 0x9F, 0x8F, 0x8F, 0x8F, 0x80, 0x80, 0x9F, 0x9F, 0x9F

db 0x9F, 0x9F, 0x9F, 0x9F, 0x9F, 0x9F, 0x9F, 0x8F, 0x8F, 0x8F, 0x80, 0x80, 0x00, 0x00, 0x00, 0x00

db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

db 0x00, 0x00, 0x00, 0x00, 0xC0, 0x40, 0x40, 0x40, 0x00, 0xC0, 0x40, 0x40, 0x40, 0x00, 0xC0, 0x40

db 0x40, 0x40, 0x00, 0xC0, 0x03, 0x03, 0xC3, 0x03, 0xC3, 0x43, 0x43, 0x43, 0x03, 0xC3, 0x03, 0x03

db 0x03, 0x83, 0x43, 0x43, 0x83, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0xC3, 0x43, 0x43, 0x43, 0x03

db 0xC3, 0x03, 0x03, 0xC3, 0x03, 0xC3, 0x43, 0x43, 0x83, 0x03, 0xC3, 0x43, 0x43, 0x43, 0x03, 0xC3

db 0x43, 0x43, 0x83, 0x03, 0xC3, 0x03, 0xC3, 0x43, 0xC3, 0x03, 0xC3, 0x43, 0x40, 0x80, 0x00, 0x00

db 0x00, 0x00, 0x00, 0x00, 0xC0, 0x40, 0x40, 0x80, 0x00, 0xC0, 0x40, 0x40, 0x40, 0x00, 0x00, 0x00

db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x00, 0xC0, 0x80

db 0x00, 0x00, 0xC0, 0x00, 0xC7, 0x45, 0x45, 0x44, 0x00, 0xC5, 0x45, 0x45, 0x47, 0x00, 0xC7, 0x84

db 0x04, 0x04, 0xC0, 0x07, 0xC4, 0x04, 0xC7, 0x40, 0x47, 0x45, 0x05, 0xC4, 0x40, 0x47, 0x84, 0x04

db 0xC0, 0x07, 0x81, 0x41, 0x47, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x05, 0x05, 0x45, 0x87, 0x00

db 0x87, 0x44, 0x04, 0x07, 0x00, 0x07, 0x01, 0x41, 0x40, 0xC0, 0x47, 0x45, 0x05, 0xC4, 0x40, 0x47

db 0x42, 0x02, 0xC5, 0x40, 0x47, 0x40, 0x07, 0xC4, 0x87, 0x00, 0x07, 0xC2, 0x02, 0xC5, 0x40, 0xC0

db 0x00, 0xC0, 0x00, 0x00, 0x07, 0xC4, 0x44, 0xC3, 0x00, 0xC7, 0x45, 0x45, 0x44, 0x00, 0xC0, 0x00

db 0x80, 0x40, 0x40, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0x00, 0x07, 0x00

db 0x01, 0x02, 0x07, 0x00, 0x07, 0x04, 0x05, 0x07, 0x00, 0x07, 0x05, 0x05, 0x04, 0x00, 0x07, 0x00

db 0x01, 0x02, 0x07, 0x00, 0x07, 0x00, 0x07, 0x05, 0x05, 0x04, 0x00, 0x07, 0x02, 0x02, 0x05, 0x00

db 0x07, 0x00, 0x07, 0x01, 0x01, 0x07, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07

db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0x00, 0x00, 0x00, 0x07, 0x05, 0x05

db 0x04, 0x00, 0x07, 0x04, 0x04, 0x04, 0x00, 0x07, 0x00, 0x01, 0x02, 0x07, 0x00, 0x07, 0x04, 0x07

db 0x00, 0x07, 0x04, 0x04, 0x00, 0x07, 0x04, 0x07, 0x00, 0x07, 0x04, 0x05, 0x07, 0x00, 0x07, 0x00

db 0x07, 0x01, 0x01, 0x07, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

```
db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

TMENU
db 0xFF, 0x01, 0xFD, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05
db 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05
db 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05
db 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05
db 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05
db 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05
db 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05
db 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0xFD, 0x01, 0xFF

db 0xFF, 0x00, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x40, 0x40, 0x40, 0x00
db 0xC0, 0x00, 0x00, 0x00, 0xC0, 0x00, 0xC0, 0x40, 0x40, 0x40, 0x00, 0xC0, 0x40, 0x40, 0x40, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x00, 0x00, 0xC0, 0x00, 0xC0, 0x80, 0x00, 0x00, 0xC0, 0x00
db 0x80, 0x40, 0x40, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x40, 0xC0, 0x00, 0xC0, 0x40
db 0x40, 0x80, 0x00, 0xC0, 0x40, 0x40, 0x40, 0x00, 0xC0, 0x00, 0xC0, 0x40, 0xC0, 0x00, 0xC0, 0x80
db 0x00, 0x00, 0xC0, 0x00, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF

db 0xFF, 0x00, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0x05, 0x05, 0x04, 0x00
db 0x07, 0x04, 0x04, 0x00, 0x07, 0x00, 0x07, 0x04, 0x05, 0x07, 0x00, 0x07, 0x05, 0x05, 0x04, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0x04, 0x04, 0x07, 0x00, 0x07, 0x00, 0x01, 0x02, 0x07, 0x00
db 0x07, 0x01, 0x01, 0x07, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0x04, 0x07, 0x00, 0x07, 0x01
db 0x01, 0x00, 0x00, 0x07, 0x04, 0x04, 0x04, 0x00, 0x07, 0x00, 0x07, 0x04, 0x07, 0x00, 0x07, 0x00
db 0x01, 0x02, 0x07, 0x00, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF

db 0xFF, 0x00, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0xE0, 0x20, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0
db 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0
db 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0
db 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0x20, 0xE0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
```

```
db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xE0, 0x20, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0
db 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0
db 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0
db 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0x20, 0xE0, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF

db 0xFF, 0x00, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00
db 0x80, 0x80, 0x80, 0x00, 0x00, 0x80, 0x00, 0x80, 0x80, 0x80, 0x00, 0x00, 0x80, 0x00, 0x00, 0x80
db 0x00, 0x00, 0x00, 0x80, 0x00, 0x00, 0x80, 0x80, 0x00, 0x00, 0x80, 0x80, 0x80, 0x00, 0x00, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00
db 0x00, 0xF8, 0x88, 0x88, 0x70, 0x00, 0xF0, 0x28, 0x28, 0xF0, 0x00, 0xF8, 0x80, 0x80, 0x00, 0xF8
db 0xA8, 0xA8, 0x88, 0x00, 0x00, 0x00, 0x00, 0xF0, 0x28, 0x28, 0xF0, 0x00, 0xF8, 0x80, 0x80, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF

db 0xFF, 0x00, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00
db 0x0F, 0x08, 0x08, 0x07, 0x00, 0x0F, 0x00, 0x0F, 0x0A, 0x0A, 0x05, 0x00, 0x0F, 0x08, 0x08, 0x0F
db 0x00, 0x08, 0x08, 0x0F, 0x00, 0x0F, 0x02, 0x02, 0x0F, 0x00, 0x0F, 0x04, 0x04, 0x0B, 0x00, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x04, 0x7C, 0x04, 0x00, 0x7C, 0x44, 0x7C, 0x00
db 0x7C, 0x14, 0x14, 0x08, 0x00, 0x7C, 0x44, 0x7C, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF

db 0xFF, 0x00, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1F, 0x10, 0x17, 0x14, 0x14, 0x14, 0x14, 0x14
db 0x14, 0x14, 0x14, 0x14, 0x14, 0x14, 0x14, 0x14, 0x14, 0x14, 0x14, 0x14, 0x14, 0x14, 0x14, 0x14
db 0x14, 0x14, 0x14, 0x14, 0x14, 0x14, 0x14, 0x14, 0x14, 0x14, 0x14, 0x14, 0x14, 0x14, 0x14, 0x14
db 0x14, 0x14, 0x14, 0x14, 0x14, 0x17, 0x10, 0x1F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1F, 0x10, 0x17, 0x14, 0x14, 0x14, 0x14, 0x14
db 0x14, 0x14, 0x14, 0x14, 0x14, 0x14, 0x14, 0x14, 0x14, 0x14, 0x14, 0x14, 0x14, 0x14, 0x14, 0x14
db 0x14, 0x14, 0x14, 0x14, 0x14, 0x14, 0x14, 0x14, 0x14, 0x14, 0x14, 0x14, 0x14, 0x14, 0x14, 0x14
db 0x14, 0x14, 0x14, 0x14, 0x14, 0x17, 0x10, 0x1F, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF

db 0xFF, 0x80, 0xBF, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0
db 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0
db 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0
db 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0
db 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0
db 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0
db 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0
```

```
db 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xBF, 0x80, 0xFF
```

```
TLABERINTO
;****************************************PÁGINA 0******************************************************
db 0xFF, 0x01, 0x01, 0x01, 0x01, 0xF9, 0x09, 0x09, 0x09, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89
db 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89
db 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89
db 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x09, 0x09, 0x09, 0x89, 0x09, 0x09, 0x09, 0xF9, 0x09, 0x09
db 0x09, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89
db 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89
db 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89
db 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x89, 0x09, 0x09, 0x09, 0xF9, 0x01, 0xFF
;****************************************PÁGINA 1******************************************************
db 0xFF, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xF8, 0x08, 0x08
db 0x08, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88
db 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88
db 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x80, 0x80, 0x80, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00
db 0x00, 0xF8, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xF8, 0x00, 0x00, 0x00, 0xF8, 0x00, 0x00
db 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x08, 0x08, 0x08, 0x88, 0x88, 0x88
db 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88
db 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0xFF, 0x00, 0xFF
;****************************************PÁGINA 2******************************************************
db 0xFF, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00
db 0x00, 0xFF, 0x00, 0x00, 0x00, 0xF8, 0x00, 0x00, 0x00, 0xF8, 0x08, 0x08, 0x08, 0x8F, 0x00, 0x00
db 0x00, 0xF8, 0x08, 0x08, 0x08, 0xF8, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xF8, 0x00, 0x00
db 0x00, 0xFF, 0x00, 0x00, 0x00, 0xF8, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00
db 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00
db 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x80, 0x80
db 0x80, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88
db 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0xFF, 0x00, 0xFF
;****************************************PÁGINA 3******************************************************
db 0xFF, 0x00, 0x00, 0x00, 0x00, 0x3F, 0x20, 0x20, 0x20, 0x3F, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00
db 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00
db 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00
db 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00
db 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00
db 0x00, 0x7F, 0x00, 0x00, 0x00, 0xFF, 0x10, 0x10, 0x10, 0x11, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00
db 0x00, 0xF8, 0x08, 0x08, 0x08, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88
db 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0xF8, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF
```

```
;*************************************PÁGINA 4*************************************************
db 0xFF, 0x00, 0x00, 0x00, 0x00, 0xFE, 0x02, 0x02, 0x02, 0xE2, 0x22, 0x22, 0x22, 0x23, 0x20, 0x20
db 0x20, 0x3F, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00
db 0x00, 0x1F, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00
db 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0x7F, 0x40, 0x40, 0x40, 0x7F, 0x00, 0x00
db 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xC7, 0x04, 0x04
db 0x04, 0xFC, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00
db 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x08, 0x08, 0x08, 0xF8, 0x08, 0xC8, 0x88, 0x08, 0x88, 0xC8
db 0x08, 0xC8, 0x48, 0x40, 0x00, 0x40, 0xC8, 0x48, 0x08, 0xC8, 0x48, 0xC8, 0x08, 0xFF, 0x00, 0xFF
;*************************************PÁGINA 5*************************************************
db 0xFF, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFE, 0x02, 0x02
db 0x02, 0xE2, 0x22, 0x22, 0x22, 0xE3, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x01, 0x01
db 0x01, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xE3, 0x22, 0x22
db 0x22, 0x23, 0x20, 0x20, 0x20, 0x23, 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0x22
db 0x22, 0x23, 0x02, 0x02, 0x02, 0x3F, 0x00, 0x00, 0x00, 0x3F, 0x20, 0x20, 0x20, 0x3F, 0x20, 0x20
db 0x20, 0x3F, 0x20, 0x20, 0x20, 0x23, 0x20, 0x20, 0x20, 0x3F, 0x20, 0x20, 0x20, 0x21, 0x20, 0x20
db 0x20, 0x3F, 0x00, 0x00, 0x00, 0x3F, 0x20, 0x20, 0x20, 0x3F, 0x20, 0x27, 0x20, 0x21, 0x20, 0x27
db 0x20, 0x27, 0x25, 0x24, 0x20, 0x20, 0x27, 0x20, 0x20, 0x27, 0x21, 0x27, 0x20, 0xFF, 0x00, 0xFF
;*************************************PÁGINA 6*************************************************
db 0xFF, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00
db 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00
db 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0x3F, 0x20, 0x20, 0x20, 0x23, 0x22, 0x22
db 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0xE2, 0x22, 0x22, 0x22, 0x22, 0x22
db 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0x22
db 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0x22
db 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0x22
db 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0x22, 0x02, 0x02, 0x02, 0xFF, 0x00, 0xFF
;*************************************PÁGINA 7*************************************************
db 0xFF, 0x80, 0x80, 0x80, 0x80, 0xBF, 0xA0, 0xA0, 0xA0, 0xA3, 0xA2, 0xA2, 0xA2, 0xA3, 0xA0, 0xA0
db 0xA0, 0xBF, 0xA0, 0xA0, 0xA0, 0xA3, 0xA0, 0xA0, 0xA0, 0xA3, 0xA0, 0xA0, 0xA0, 0xBF, 0xA0, 0xA0
db 0xA0, 0xA3, 0xA0, 0xA0, 0xA0, 0xA3, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2
db 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA3, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2
db 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA0, 0xA0, 0xA0, 0xBE
db 0xA0, 0xA0, 0xA0, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2
db 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2
db 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xA2, 0xBF, 0x80, 0xFF
```

TJUEGOTOPO

```
db 0xFF, 0x01, 0xFD, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05
db 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05
db 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05
db 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05
db 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05
db 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05
db 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05
db 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0xFD, 0x01, 0xFF

db 0xFF, 0x00, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x80, 0x40, 0x20, 0x20, 0x20, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x20
db 0x20, 0x20, 0x40, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x40, 0x20, 0x20, 0x20, 0x10, 0x10, 0x10
db 0x10, 0x10, 0x10, 0x20, 0x20, 0x20, 0x40, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x40, 0x20, 0x20
db 0x20, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x20, 0x20, 0x20, 0x40, 0x80, 0x00, 0x00, 0x00, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF

db 0xFF, 0x00, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
db 0x00, 0x00, 0x00, 0x1E, 0x31, 0x40, 0x40, 0x80, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80
db 0x80, 0x40, 0x40, 0x31, 0x1E, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1E, 0x31, 0x40, 0x40, 0x80, 0x80, 0x00, 0x00, 0x00
db 0x00, 0x00, 0x00, 0x80, 0x80, 0x40, 0x40, 0x31, 0x1E, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1E, 0x31, 0x40, 0x40, 0x80
db 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x80, 0x40, 0x40, 0x31, 0x1E, 0x00, 0x00, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF

db 0xFF, 0x00, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x40, 0x20, 0x20, 0x20, 0x10, 0x10, 0x10, 0x10, 0x10
db 0x10, 0x20, 0x20, 0x20, 0x40, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x01, 0x01
db 0x01, 0x01, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x40, 0x20, 0x20, 0x20, 0x10
db 0x10, 0x10, 0x10, 0x10, 0x10, 0x20, 0x20, 0x20, 0x40, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
db 0x00, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF

db 0xFF, 0x00, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0x1E, 0x31, 0x40, 0x40, 0x80, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00
```

```
db 0x00, 0x80, 0x80, 0x40, 0x40, 0x31, 0x1E, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1E, 0x31, 0x40, 0x40, 0x80, 0x80, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x80, 0x40, 0x40, 0x31, 0x1E, 0x00, 0x00, 0x00, 0x00, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF

db 0xFF, 0x00, 0xFF, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80
db 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80
db 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x81, 0x81, 0x81, 0x81, 0x81
db 0x81, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80
db 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x81
db 0x81, 0x81, 0x81, 0x81, 0x81, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80
db 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80
db 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0xFF, 0x00, 0xFF

db 0xFF, 0x00, 0xFC, 0x04, 0xE4, 0x04, 0xE4, 0x04, 0xE4, 0x04, 0xE4, 0x24, 0xC4, 0x04, 0xC4, 0xA4
db 0xC4, 0x04, 0xE4, 0xA4, 0xA4, 0x04, 0x44, 0x04, 0x14, 0x94, 0xF4, 0x04, 0xFC, 0x04, 0xE4, 0xA4
db 0x44, 0x04, 0xE4, 0x04, 0xE4, 0x04, 0xE4, 0x44, 0x84, 0xE4, 0x04, 0x24, 0xE4, 0x24, 0x04, 0xE4
db 0x24, 0xE4, 0x04, 0xE4, 0xA4, 0xA4, 0x04, 0x44, 0x04, 0xF4, 0x14, 0xF4, 0x04, 0xF4, 0x14, 0xF4
db 0x04, 0xFC, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFE, 0x02, 0xE2, 0x22, 0xC2, 0x02, 0xE2, 0xA2, 0x22
db 0x02, 0xE2, 0x02, 0xE2, 0x42, 0x82, 0xE2, 0x02, 0xE2, 0x02, 0xE2, 0x22, 0x22, 0x02, 0xE2, 0x02
db 0xC2, 0xA2, 0xC2, 0x02, 0xE2, 0x22, 0xC2, 0x02, 0xFE, 0x02, 0xE2, 0x22, 0x42, 0x22, 0xE2, 0x02
db 0xE2, 0xA2, 0x22, 0x02, 0xE2, 0x42, 0x82, 0xE2, 0x02, 0xE2, 0x02, 0xE2, 0x02, 0xFE, 0x00, 0xFF

db 0xFF, 0x80, 0x9F, 0x90, 0x91, 0x92, 0x91, 0x90, 0x93, 0x90, 0x93, 0x92, 0x91, 0x90, 0x93, 0x90
db 0x93, 0x90, 0x92, 0x92, 0x93, 0x90, 0x91, 0x90, 0x94, 0x94, 0x97, 0x90, 0x9F, 0x90, 0x93, 0x90
db 0x90, 0x90, 0x93, 0x92, 0x93, 0x90, 0x93, 0x90, 0x90, 0x93, 0x90, 0x90, 0x93, 0x90, 0x90, 0x93
db 0x92, 0x93, 0x90, 0x92, 0x92, 0x93, 0x90, 0x91, 0x90, 0x97, 0x94, 0x97, 0x90, 0x97, 0x94, 0x97
db 0x90, 0x9F, 0x80, 0x80, 0x80, 0x80, 0x80, 0xBF, 0xA0, 0xA3, 0xA1, 0xA2, 0xA0, 0xA3, 0xA2, 0xA2
db 0xA0, 0xA3, 0xA0, 0xA3, 0xA0, 0xA0, 0xA3, 0xA0, 0xA3, 0xA0, 0xA3, 0xA2, 0xA2, 0xA0, 0xA3, 0xA0
db 0xA3, 0xA0, 0xA3, 0xA0, 0xA3, 0xA1, 0xA2, 0xA0, 0xBF, 0xA0, 0xA3, 0xA0, 0xA0, 0xA0, 0xA3, 0xA0
db 0xA3, 0xA2, 0xA2, 0xA0, 0xA3, 0xA0, 0xA0, 0xA3, 0xA0, 0xA3, 0xA2, 0xA3, 0xA0, 0xBF, 0x80, 0xFF
```

TNUMERO

```
db 0x04, 0x24, 0xF4, 0x90, 0x90, 0x97, 0x94, 0x94, 0xF4, 0x97, 0x94, 0x94, 0x14, 0x94, 0xF4, 0x94
db 0x94, 0x97, 0xF4, 0x84, 0xE4, 0x90, 0x90, 0x97, 0xF4, 0x94, 0x94, 0x94, 0x94, 0x97, 0xF4, 0x84
db 0x84, 0x97, 0x94, 0x97, 0x14, 0x14, 0xF4, 0x90, 0x90, 0x97 ,0xF4, 0x94, 0xF4, 0x97, 0x94, 0x97
db 0xF4, 0x94, 0xF4, 0x90, 0x90, 0x97, 0xF4, 0x14, 0xF4, 0x97, 0x94, 0x97
```

```
TMENSAJE
    db 0xF8, 0x08, 0xE8, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28
    db 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28
    db 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28
    db 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28
    db 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0xE8, 0x08, 0xF8, 0xFF, 0x00
    db 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x80, 0x80, 0x00, 0x00, 0x1F, 0x85, 0x85, 0x02
    db 0x00, 0x9F, 0x09, 0x09, 0x16, 0x80, 0x1F, 0x95, 0x95, 0x91, 0x00, 0x17, 0x95, 0x95, 0x1D, 0x00
    db 0x9F, 0x00, 0x1F, 0x11, 0x9F, 0x00, 0x1F, 0x02, 0x04, 0x88, 0x9F, 0x00, 0x1E, 0x05, 0x05, 0x1E
    db 0x80, 0x80, 0x80, 0x00, 0x00, 0x1F, 0x90, 0x90, 0x00, 0x1E, 0x85, 0x85, 0x9E, 0x00, 0x00, 0x00
    db 0x80, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF, 0xFF, 0x00, 0xFF, 0x00
    db 0x00, 0x00, 0x00, 0x00, 0x00, 0x0F, 0x02, 0x02, 0x01, 0x00, 0x0F, 0x02, 0x02, 0x0F, 0x00, 0xCF
    db 0x41, 0x42, 0x44, 0x0F, 0xC0, 0x40, 0xCF, 0x00, 0xC0, 0x4F, 0x82, 0x42, 0xCF, 0x00, 0xCF, 0x48
    db 0x48, 0x40, 0x0F, 0xC8, 0x88, 0x00, 0x0F, 0xC2, 0x02, 0x4F, 0x40, 0xC0, 0x00, 0x80, 0x4F, 0x42
    db 0x82, 0x01, 0xC0, 0x4F, 0x42, 0x82, 0x0F, 0x00, 0x0F, 0x04, 0x04, 0x0B, 0x00, 0x0F, 0x02, 0x02
    db 0x0F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF, 0xFF, 0x80, 0xBF, 0xA0, 0xA0, 0xA0
    db 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA7, 0xA4, 0xA4
    db 0xA4, 0xA0, 0xA7, 0xA4, 0xA7, 0xA0, 0xA7, 0xA0, 0xA0, 0xA0, 0xA7, 0xA0, 0xA7, 0xA5, 0xA5, 0xA4
    db 0xA0, 0xA7, 0xA0, 0xA1, 0xA2, 0xA7, 0xA0, 0xA6, 0xA5, 0xA4, 0xA0, 0xA7, 0xA1, 0xA1, 0xA7, 0xA0
    db 0xA7, 0xA2, 0xA2, 0xA5, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0
    db 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xBF, 0x80, 0xFF
TGAMEOVER
    db 0xF8, 0x08, 0xE8, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28
    db 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28
    db 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28
    db 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28
    db 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0xE8, 0x08, 0xF8, 0xFF, 0x00
    db 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x80
    db 0x80, 0x80, 0x00, 0x00, 0x80, 0x80, 0x00, 0x00, 0x80, 0x80, 0x00, 0x80, 0x80, 0x00, 0x80, 0x80
    db 0x80, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x80, 0x80, 0x00, 0x80, 0x00, 0x80, 0x00

    db 0x80, 0x80, 0x80, 0x80, 0x00, 0x80, 0x80, 0x80, 0x00, 0x00, 0x00, 0xC0, 0x00, 0x00, 0x00, 0x00
    db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF, 0xFF, 0x00, 0xFF, 0x00
    db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0F, 0x08, 0x0A, 0x0E
    db 0x00, 0x0F, 0x02, 0x02, 0x0F, 0x00, 0x0F, 0x00, 0x01, 0x00, 0x0F, 0x00, 0x0F, 0x0A, 0x0A, 0x08
    db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0F, 0x08, 0x0F, 0x00, 0x07, 0x08, 0x07, 0x00, 0x0F, 0x0A
    db 0x0A, 0x08, 0x00, 0x0F, 0x04, 0x04, 0x0B, 0x00, 0x00, 0x0B, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF, 0xFF, 0x80, 0xBF, 0xA0, 0xA0, 0xA0
    db 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0
```

```
    db 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0
    db 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0
    db 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0
    db 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xBF, 0x80, 0xFF

TYOUWIN
    db 0xF8, 0x08, 0xE8, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28
    db 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28
    db 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28
    db 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28
    db 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0x28, 0xE8, 0x08, 0xF8, 0xFF, 0x00
    db 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    db 0x00, 0x00, 0x00, 0x00, 0x80, 0x00, 0x00, 0x00, 0x80, 0x00, 0x80, 0x80, 0x80, 0x00, 0x80, 0x00
    db 0x00, 0x80, 0x00, 0x00, 0x00, 0x00, 0x80, 0x00, 0x00, 0x00, 0x80, 0x00, 0x00, 0x80, 0x00, 0x80
    db 0x00, 0x00, 0x00, 0x80, 0x00, 0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF, 0xFF, 0x00, 0xFF, 0x00
    db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    db 0x00, 0x00, 0x01, 0x02, 0x0C, 0x02, 0x01, 0x00, 0x0F, 0x08, 0x0F, 0x00, 0x0F, 0x08, 0x08, 0x0F
    db 0x00, 0x00, 0x00, 0x00, 0x07, 0x08, 0x06, 0x08, 0x07, 0x00, 0x00, 0x0F, 0x00, 0x0F, 0x01, 0x02
    db 0x04, 0x0F, 0x00, 0x0B, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF, 0xFF, 0x80, 0xBF, 0xA0, 0xA0, 0xA0
    db 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0
    db 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0
    db 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0
    db 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xA0
    db 0xA0, 0xA0, 0xA0, 0xA0, 0xA0, 0xBF, 0x80, 0xFF

THOJADIBUJO
    db 0xFF, 0x01, 0xFD, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05
    db 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05
    db 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05
    db 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05
    db 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05
    db 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05
    db 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05
    db 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0xFD, 0x01, 0xFF

    db 0xFF, 0x00, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
```

db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF

db 0xFF, 0x00, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF

db 0xFF, 0x00, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF

db 0xFF, 0x00, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF

db 0xFF, 0x00, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

```
db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x00, 0xFF

db 0xFF, 0x00, 0xF3, 0x12, 0xD2, 0x12, 0x12, 0x12, 0xD2, 0x12, 0xD2, 0x52, 0x92, 0x52, 0xD2, 0x12
db 0xD2, 0x52, 0x52, 0x92, 0x12, 0xD2, 0x12, 0x92, 0x52, 0x52, 0x92, 0x12, 0xD2, 0x52, 0x52, 0x92
db 0x12, 0x12, 0xF2, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02
db 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02
db 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02
db 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02
db 0x02, 0xF2, 0x12, 0xD2, 0x12, 0xD2, 0x12, 0xD2, 0x52, 0xD2, 0x12, 0xD2, 0x12, 0x12, 0x12, 0xD2
db 0x12, 0xD2, 0x12, 0xD2, 0x52, 0x52, 0x52, 0x12, 0xD2, 0x52, 0x52, 0x92, 0x12, 0xF3, 0x00, 0xFF

db 0xFF, 0x80, 0x9F, 0x90, 0x97, 0x94, 0x94, 0x90, 0x97, 0x90, 0x97, 0x90, 0x90, 0x90, 0x97, 0x90
db 0x97, 0x91, 0x91, 0x90, 0x90, 0x97, 0x90, 0x97, 0x91, 0x91, 0x97, 0x90, 0x97, 0x92, 0x92, 0x95
db 0x90, 0x90, 0x9F, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80
db 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80
db 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80
db 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80
db 0x80, 0x9F, 0x90, 0x93, 0x94, 0x93, 0x90, 0x97, 0x94, 0x97, 0x90, 0x97, 0x94, 0x94, 0x90, 0x93
db 0x94, 0x93, 0x90, 0x97, 0x95, 0x95, 0x94, 0x90, 0x97, 0x92, 0x92, 0x95, 0x90, 0x9F, 0x80, 0xFF
TTOPO
db 0xE0, 0x10, 0xC8, 0x08, 0x08, 0xC8, 0x10, 0xE0, 0xFF, 0x00, 0x09, 0x14, 0x14, 0x09, 0x00, 0xFF
db 0x20, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x20, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80
END
```