ESTUDIO Y DISEÑO DE ALGORITMOS E INTERFAZ ELECTRÓNICA DE COMUNICACIÓN ENTRE MICROCONTROLADORES SERIE PIC Y DISPOSITIVOS DE ALMACENAMIENTO USB

JUAN PABLO RUIZ ROSERO

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERIA
PROGRAMA DE INGENIERIA ELECTRONICA
SAN JUAN DE PASTO
2008

ESTUDIO Y DISEÑO DE ALGORITMOS E INTERFAZ ELECTRÓNICA DE COMUNICACIÓN ENTRE MICROCONTROLADORES SERIE PIC Y DISPOSITIVOS DE ALMACENAMIENTO USB

JUAN PABLO RUIZ ROSERO

TRABAJO DE GRADO

DIRECTOR:
ING. JUAN CARLOS CASTILLO ERASO

CODIRECTOR:
ING. DARIO FERNANDO FAJARDO FAJARDO

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERIA
PROGRAMA DE INGENIERIA ELECTRONICA
SAN JUAN DE PASTO
2008

Nota de aceptación

_____

_____

_____

_____

_____

_____

_____

Firma del presidente del jurado

_____

Firma del jurado

_____

Firma del jurado

San Juan de Pasto, Noviembre 2008

"Las ideas y conclusiones aportadas en la tesis de grado son responsabilidad exclusiva de los autores"
Artículo 1. del acuerdo No. 324 del 11 de Octubre de 1966, emanado por el Honorable Consejo Directivo de la Universidad de Nariño.

A Dios que me dio todas las herramientas y capacidad para la realización de este trabajo.

A mis padres Alvaro Ruiz y Marleny Rosero que siempre han sido un gran apoyo en todo lo que he encaminado

A Julieta por todo el amor y apoyo que me ha brindado

A todos mis familiares y amigos

Juan Pablo Ruiz Rosero

AGRADECIMIENTOS

Ing. Juan Carlos Castillo Eraso, director del trabajo de grado por sus consejos y su dedicación

Ing. Dario Fernando Fajardo Fajardo, codirector del trabajo de grado, por su ejemplo, por su dedicación, por todas sus enseñanzas y por su interés en el proyecto.

Al programa de Ingeniería Electrónica de la Universidad de Nariño, a mis compañeros, amigos y profesores y por lo enriquecedora que ha sido la carrera.

Al Aula de Informática de la Universidad de Nariño y a mis amigos monitores por conformar con todos una gran empresa durante un largo tiempo

A mi Universidad de Nariño por ser la sede de muchos de mis éxitos y trabajos.

CONTENIDO

LISTA DE CUADROS

LISTA DE FIGURAS

LISTA DE ANEXOS

GLOSARIO

A/D: conversión de una señal análoga a digital.

API: (Application Programming Interface), es el conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

BCD: (Binary Coded Decimal), sistema numérico usado en sistemas computacionales y electrónicos para codificar números enteros positivos y facilitar las operaciones aritméticas. Es un código pesado debido a que representa los dígitos con un orden específico.

D/A: conversión de una señal digital a análoga.

ECG: (Electrocardiograma), es el gráfico que se obtiene con el electrocardiógrafo para medir la actividad eléctrica del corazón en forma de cinta gráfica continua.

FAT: (File Allocation Table), tabla de asignación de archivos, es un sistema de archivos desarrollado para MS-DOS, así como el sistema de archivos principal de las ediciones no empresariales de Microsoft Windows hasta Windows Me.

FIRMWARE: bloque de instrucciones de un programa para propósitos específicos, grabado en una memoria tipo ROM, que establece la lógica de más bajo nivel que controla los circuitos electrónicos de un dispositivo de cualquier tipo. Al estar integrado en la electrónica del dispositivo es en parte hardware, pero también es software, ya que proporciona lógica y se dispone en algún tipo de lenguaje de programación. Funcionalmente, el firmware es el intermediario (interfaz) entre las órdenes externas que recibe el dispositivo y su electrónica, ya que es el encargado de controlar a ésta última para ejecutar correctamente dichas órdenes externas.

HOST: equipo anfitrión en una comunicación.

HOST USB: equipo anfitrión de dispositivos USB

HUB: dispositivo que permite centralizar el cableado de una red y poder ampliarla. Este recibe una señal y la repite emitiéndola por sus diferentes puertos.

LCD: (Liquid Crystal Display), pantalla de cristal líquido, delgada y plana formada por un número de píxeles en color o monocromos colocados delante de una fuente de luz o reflectora.

LED: (Light-Emitting Diode), diodo emisor de luz.

LSB: (Less Significant Bit), bit menos significativo, es la posición de bit en un número binario que tiene el menor valor.

MSB: (Most Significant Bit), bit mas significativo, es la posición de bit en número binario que tiene al mayor valor.

NRZI: (Non Return to Zero Inverted), variante del código NRZ, se representa únicamente los "1"por cambio en la polaridad del medio magnético, que se traduce en un impulso durante la operación de lectura. La inexistencia de señal se interpreta como un "0".

PIC: tipo de microcontrolador de Microchip

PROTOCOLO DE COMUNICACIÓN: conjunto de reglas que especifican el intercambio de datos u órdenes durante la comunicación entre las entidades que forman parte de una red.

RAM: (Random Access Memory), memoria de acceso aleatoreo, se compone de uno o más chips y se utiliza como memoria de trabajo para programas y datos. Es un tipo de memoria temporal que pierde sus datos cuando se queda sin energía (por ejemplo, al apagar la computadora), por lo cual es una memoria volátil.

RESET: puesta en condiciones iniciales de un sistema. Este puede ser mecánico, electrónico o de otro tipo.

ROM: (Read Only Memory), memoria de solo lectura, una memoria de semiconductor destinada a ser leída y no destructible, es decir, que no se puede escribir sobre ella y que conserva intacta la información almacenada, incluso en el caso de que se interrumpa la corriente (memoria no volátil).

RS-232: es una interfaz que designa una norma para el intercambio serie de datos binarios entre un DTE (Equipo terminal de datos) y un DCE (Data Communication Equipment, Equipo de Comunicación de datos), aunque existen otras situaciones en las que también se utiliza la interfaz RS-232.

SPI: (Serial Peripheral Interface), es un estándar de comunicaciones, usado principalmente para la transferencia de información entre circuitos integrados en equipos electrónicos. El bus de interfase de periféricos serie o bus SPI es un estándar para controlar casi cualquier electrónica digital que acepte un flujo de bits serie regulado por un reloj.

USART: (Universal Synchronous/Asynchronous Receiver Transmitter), periférico para la transmisión de datos en formato serie, usa técnicas de transmisión sincrónica o asincrónica, según se configure el periférico.

USB: (Universal Serial Bus), puerto que sirve para conectar periféricos en caliente a una computadora.

VDRIVE1: host usb que maneja dispositivos de almacenamiento USB.

RESUMEN

Las memorias USB son en la actualidad uno de los medios de almacenamiento de datos mas populares por su parte los microcontroladores PIC de la Microchip son de los circuitos integrados programables mas usados en la actualidad, este proyecto desarrolla un método para lograr la comunicación entre estos dos elementos con el fin de aumentar las posibilidades en las cuales se limita, por falta de memoria, un microcontrolador.

Se realiza un estudio del modulo VDRIVE1 de la empresa Vinculum el cual funciona como Host USB para dispositivos de almacenamiento USB. Se desarrolla una terminal USB a SPI para comunicarse desde el computador con el VDRIVE1 para probar su funcionamiento y comportamiento ante distintos comandos, con esto se crea una librería bien documentada para que un microcontrolador pueda comunicarse con la memoria USB.

Se implementan dos soluciones que hacen uso del modulo VDRIVE1 y de la librería. Una de ellas es una matriz de LEDS de 8 por 26 en la cual se muestran los caracteres almacenados en un archivo de texto dentro de la memoria USB. La otra solución es un electrocardiógrafo que almacena la señal ECG de una persona dentro de un archivo en una memoria USB también.

ABSTRACT

The USB memory sticks are now one of the most popular storage media data, on the other hand the PIC Microchip microcontrollers are one of those more used integrated circuits in the present, this project make a development of a method to get communication between these two elements in order to increase the functionality which is limited by the lack of memory in a microcontroller.

Here were carried out a study of the VDRIVE1 module made by the Vinculum company which works as a USB Host with a USB mass storage. The protect develops a USB to SPI terminal to communicate the computer with the VDRIVE1 to test its performance and behavior to various commands, this help to create a library well documented for the communication of the microcontroller with the USB mass storage.

Two solutions are implemented that make VDRIVE1 use the library. One is an array of LEDS, 8 by 26 in which the characters ares stored in a text file within in the USB stick. The other solution is an electrocardiograph that stores the EKG signal of a person within a file on a USB memory as.

INTRODUCCION

En el desarrollo de soluciones digitales es muy importante contar con un medio de almacenamiento masivo de datos para guardar información de señales monitorizadas, diferentes estados, configuraciones, alarmas, notificaciones, entre otros. Por ejemplo en el área de la electro medicina un equipo que monitoriza signos vitales de un paciente, al cual se le esta haciendo un seguimiento continuo, necesita de una memoria donde guardar los datos que se adquieren; al ser en muchos casos tantas las señales que pueden ser monitorizadas en lapsos muy cortos de tiempo; la memoria debe ser de una gran capacidad. En áreas también como la geofísica se hace en muchas ocasiones necesario hacer medidas a campo abierto en lugares remotos donde es muy difícil contar con un medio de comunicación para transmitir una señal.

Los dispositivos de almacenamiento USB o memorias USB han tenido un gran auge en los últimos años, son un invento novedoso que sustituyen totalmente a los disquetes; almacenando hasta mas de 1000 veces su capacidad con un tamaño muy pequeño[1]. Las memorias USB desplazan a otras memorias no volátiles como las SD debido a que no necesitan un accesorio extra para conectarse al computador, su robustez eléctrica la hace mas resistente contra accidentes y sus grandes prestaciones la hacen muy popular entre todo tipo de usuarios. Por otro lado en la implementación de soluciones digitales se hace uso muy frecuente de los microcontroladores, dispositivos programables, de los cuales son muy comunes los de la serie PIC de la Microchip gracias a su buena relación costo beneficio.

Es por esta razón que este proyecto se propone como principal objetivo explorar la comunicación entre los dispositivos de almacenamiento USB y los microcontroladores serie PIC de la Microchip, esto mediante una investigación del protocolo de comunicaciones USB, el sistema de archivos FAT32, desarrollando una interfaz eléctrica o modulo de comunicación que permita interactuar al microcontrolador con la memoria, el diseño de algoritmos de comunicación, la creación de una librería que agrupe estos algoritmos y su implementación en dos ejemplos prácticos.

[1]Revista Consumer, España Octubre de 2005

Para el desarrollo de este proyecto se hizo uso de la metodología de investigación exploratoria puesto que se estudiaron, analizaron, diseñaron y se colocaron en practica modelos de comunicación que muy poco han sido implementados en esta área, que sirven para transmitir datos entre un sistema manejado por microcontrolador y un dispositivo de almacenamiento masivo. Los resultados obtenidos han sido compilados en una librería para su ampliación en soluciones universales basadas en microcontroladores de la serie PIC de la Microchip.

1 MICROCONTROLADORES

Según Palacios Municio "Un microcontrolador es un circuito integrado programable que contiene todos los componentes necesarios para controlar el funcionamiento de una tarea determinada, como el control de una lavadora, un teclado de ordenador, una impresora, un sistema de alarma, etc."[1].

Los microcontroladores son circuitos lógicos digitales que trabajan con un programa el cual es ejecutado en forma secuencial, es decir, una instrucción tras otra. Estos cuentan para su propósito con:

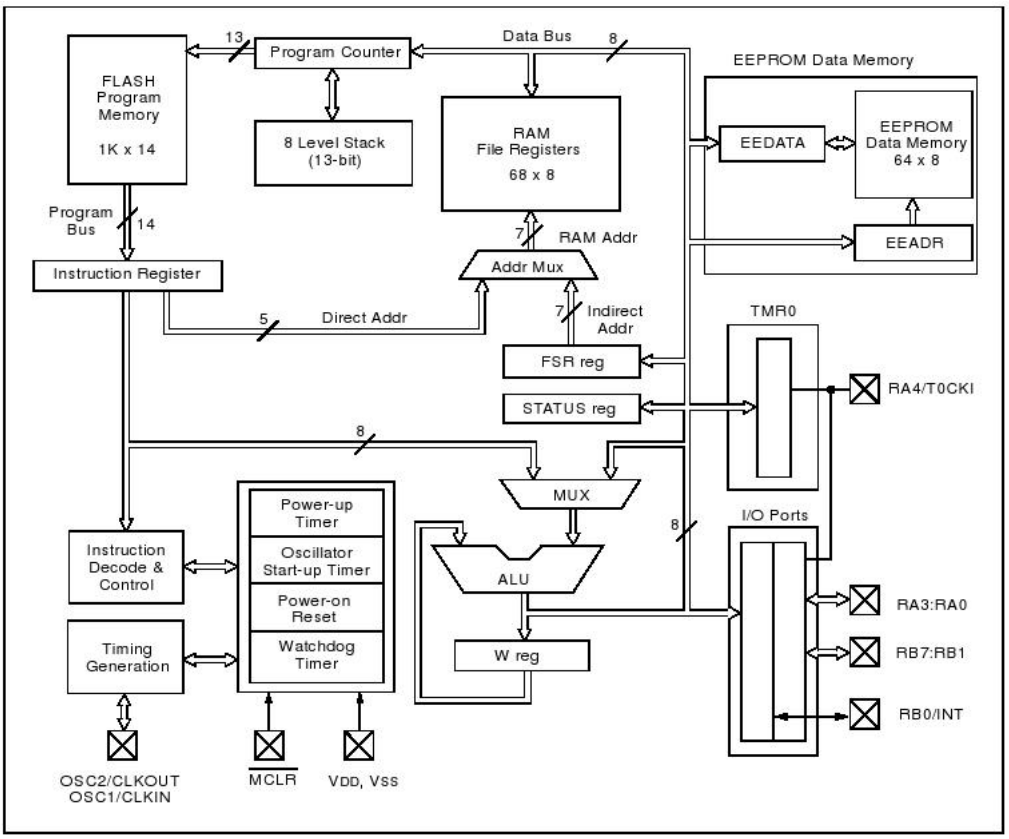
- Una memoria EPROM no volátil para almacenar las instrucciones del programa
- Puertos configurables de entrada y/o salida mediante los cuales se comunica con el exterior
- Una pequeña memoria RAM para almacenar datos de variables durante las operaciones
- Registros especiales para configurar su funcionamiento
- Una unidad aritmética lógica (ALU) y un registro de trabajo (W) con los cuales se procesan las operaciones
- Un multiplexor para distribuir los datos
- Contador rápido TIMER
- Una pequeña memoria EPROM no volátil para almacenar datos de variables que no se quieran perder
- Conversores A/D y D/A (PIC16F870, PIC18F452,...)
- Puertos de comunicación: RS-232, USB, I2C, etc. (PIC16F870, PIC18F452,...)

[1]PALACIOS MUNICIO, Enrique, REMIRO DOMINGUEZ, Fernando y LOPEZ PEREZ, Lucas, Microcontroladores PIC16F84, Desarrollo de proyectos, Madrid: Ra-Ma, 2004. p.1

- Modulador de anchura de pulsos PWM (PIC16F870, PIC18F452,...)
- Modulo de transmisión inalámbrica (RFPIC)
- Modulo de procesamiento digital de señales (DSPIC)

En la Figura 1.1 se pueden observar las características ya mencionadas en un microcontrolador PIC16F84 de la Microchip.

Figura 1.1: Diagrama de Bloques PIC16F84



Microchip Technology Incorporated, U.S.A., 2001.

1.1. APLICACION MICROCONTROLADORES

Los microcontroladores contienen todos los componentes de un computador en una escala mas pequeña, pero a diferencia de ellos estos circuitos integrados son utilizados para

tareas determinadas. Estas tareas comprenden campos como los de el control industrial, la robótica, las telecomunicaciones, la instrumentación dentro de la física y química, la electormedicina, el diseño de soluciones en automatización, entre otros. La siguiente tabla muestra de una manera mas detallada la utilización de los microcontroladores en diferentes sectores.

- Instrumentos portátiles compactos:
  - Radio paginador numérico (beeper)
  - Planímetro electrónico
  - Nivelímetro digital
  - Identificador-probador de circuitos integra-dos
  - Tacómetro digital (desarrollado en el Labora-torio de Electrónica)
- Dispositivos periféricos:
  - Módems
  - Buffer para impresoras
  - Plotters
  - Posicionadores
- Dispositivos autonomos:
  - Fotocopiadoras
  - Máquinas de escribir
  - Selector, Codificador-decodificador de T.V.
  - Localizador de peces
  - Controlador de aspersores pararíego de jardines
  - Teléfonos de tarjeta
- Subfunciones de instrumentos:
  - Panel frontal de un osciloscopio
  - Controlador de display de cristal líquido sensible al tacto
  - Contador de microondas con interfase HP-IB
  - Multímetro multiplexado con interfaz serie a otro multímetro
  - Analizador de espectros (módulos de expansión para IR y RF comunicados vía Inter-face serie)

- Aplicaciones automotrices:
  - Control de encendido e inyección de combustible
  - Sistemas de frenado antiderrapante
  - Control dinámico de la suspensión
  - Sistemas de navegación
  - Alarmas automotrices
- Otros:
  - Controladores de CRT
  - Teléfonos celulares
  - Cerraduras electrónicas
  - Sistemas de seguridad[2].

1.2. FAMILIAS

Dentro del mercado existen muchos tipos y marcas de microcontroladores, los mas comunes son los siguientes:

- **Intel 8048:** Considerado el padre de los microcontroladores actuales, fue el primer microcontrolador de Intel, usado en sintetizadores musicales, teclados IBM entre otros. Poseía una memoria ROM interna o externa de 64-256 bytes.
- **Intel 8051:** Es uno de los mas populares microcontroladores, fue desarrollado en el año 1980 por la empresa Intel, posee un bus de 8 bits, una memoria interna de hasta 32K, puerto serial, timers y contadores internos.
- **Intel 80186 y 80188 (i80186 e i81088):** Son microcontroladores desarrollados en el año de 1982 basados en el procesador 8086, poseen buses de 8 y 16 bits, un set de instrucciones mas amplio, 1Mbyte de direccionamiento para programas y datos y una interfaz numérica directa.

[2]RINCON PASAYE, José Juan, Introducción a los microcontroladores,[monografía en linea], 1 ed, [Morelia, México]: Universidad Michoacana de San Nicolás de Hidalgo, 1997, Disponible en internet: URL: http://lc.fie.umich.mx/ jrincon/intro-microcon.pdf. p.9

- **Freescale 683XX (Motorola):** Microcontroladores diseñados para altas prestaciones, cuentan con un bus de datos de 32 bits, conversores A/D y D/A, protocolos de comunicación integrados, soporte a memorias externas SRAM, comunicación serial Profibus,HDLC,Ethernet, entre otros.
- **PIC (Microchip):** Microcontroladores muy populares hoy en día debido a su bajo costo y su gran gama de posibilidades, poseen buses de 8, 16 y 32 bits, conversores A/D y D/A, sistemas DSP(Procesamiento digital de señales) y RF (Radio Frecuencia), timers internos, sistemas de comunicación UART, RS232, I2C, módulos PWM, entre muchos otros.

1.3. MICROCONTROLADORES PIC

La empresa Microchip desarrolla los microcontroladores serie PIC los cuales poseen las siguientes características:

- Núcleos de UCP de 8/16/32 bits con Arquitectura Harvard modificada
- Memoria Flash y ROM disponible desde 256 bytes a 256 kilobytes
- Puertos de E/S (típicamente 0 a 5,5 voltios)
- Temporizadores de 8/16 bits
- Tecnología Nanowatt para modos de control de energía
- Periféricos serie síncronos y asíncronos: USART, AUSART, EUSART
- Conversores analógico/digital de 10-12 bits
- Comparadores de tensión
- Módulos de captura y comparación PWM
- Controladores LCD
- Periférico MSSP para comunicaciones I2C, SPI, y I2S
- Memoria EEPROM interna con duración de hasta un millón de ciclos de lectura/escritura

- Periféricos de control de motores
- Soporte de interfaz USB
- Soporte de controlador Ethernet
- Soporte de controlador CAN
- Soporte de controlador LIN
- Soporte de controlador Irda

1.3.1. Entorno de desarrollo. Microchip proporciona un entorno de desarrollo freeware llamado MPLAB que incluye un simulador software y un ensamblador. Otras empresas desarrollan compiladores C y BASIC. Microchip también vende compiladores para los PICs de gama alta (Ç18"para la serie F18 y Ç30"para los dsPICs) y se puede descargar una edición para estudiantes del C18 que inhabilita algunas opciones después de un tiempo de evaluación.

1.3.2. Programación. Mediante un dispositivo llamado programador se transfiere el código compilado del computador al microcontrolador. Existen programadores que se conectan a diversos puertos como el puerto serial o el puerto paralelo o nuevos programadores que hacen uso del puerto USB. Para este proyecto se emplea el programador PICkit 2, ver Figura 1.2, de la Microchip el cual se conecta por medio del puerto USB al computador, posee también la funcionalidad de escribir el código compilado desde el mismo entorno de desarrollo MPLAB.

Figura 1.2: PICkit 2

2 MODULO HOST USB VDRIVE1

2.1. PROTOCOLO USB

El puerto de bus serial universal, reconocido por sus siglas en ingles USB (Universal Serial Bus) se ha convertido en uno de los principales dentro de las comunicaciones entre distintos dispositivos con el computador gracias a aspectos como:

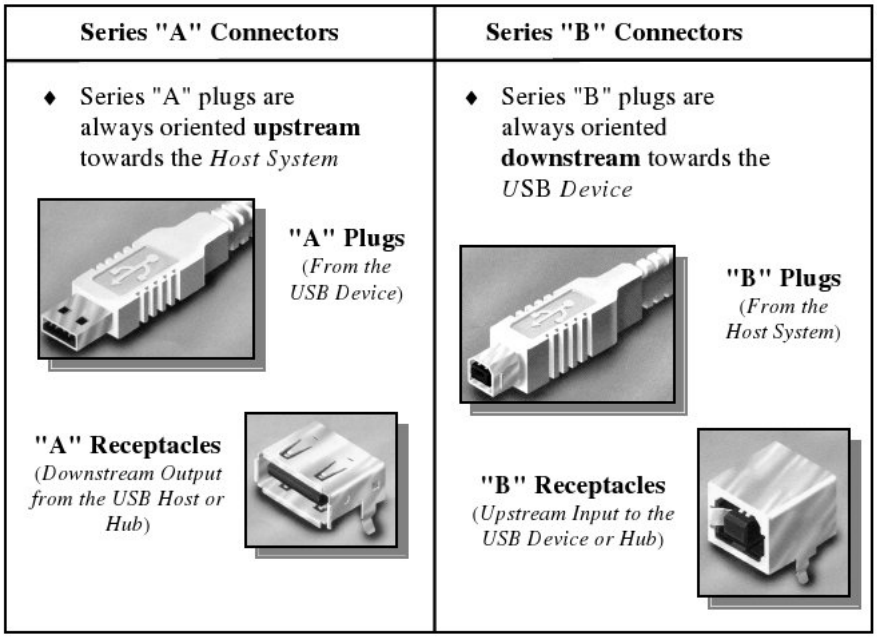
- Trae con sigo la alimentación para el dispositivo externo por lo cual este no necesita de una fuente de energía extra para funcionar.
- Mejora las características Plug and Play pues no es necesario reiniciar el equipo para conectar o desconectar un dispositivo.
- Posee un amplio rango de velocidades y modos de transmisión útiles para soportar una gran cantidad de periféricos distintos.

Su conector es universal y posee características que permiten un rápido y cómodo acople y desacople con el computador, además esta diseñado para una larga durabilidad y vulnerabilidad a altas tensiones accidentales provocadas por contacto con objetos cargados. Su estructura mecánica se puede apreciar en la Figura 2.1.

Existen diversos modos de funcionamiento los cuales permiten que el puerto USB se preste para variadas funcionalidades, estos modos se describen en el Cuadro 2.1.

2.1.1. Arquitectura USB El bus USB soporta intercambio de datos entre el computador host y un amplio rango de periféricos, estos comparten el ancho de banda USB provisto por el host con un protocolo basado en señales. Este bus permite conectar o desconectar dispositivos sin que otros que están conectados se desconecten o se vean afectados. Para conocer mas a fondo su funcionamiento se describen a continuación tres áreas que son: interconexión USB, USB Host y dispositivos USB.

Figura 2.1: Conectores USB



Universal Serial Bus Specifications, 2000.

Interconexión USB. La interconexión USB maneja el modo como los dispositivos USB se conectan y comunican con el host lo cual incluye la topologia del bus, la relaciones entre capas, los modelos de fluido de datos y las reglas provistas por el estándar USB. La topología del bus USB se basa en la topología estrella para la conexión donde cada punto central es un HUB los cuales se conectan al host, a otro HUB o a un dispositivo USB. La Figura 2.2 ilustra este tipo de conexión.

Debido a retardos de tiempo entre los HUB y los cables, el máximo número de filas permitido es siete, incluyendo la fila del host. En estas siete filas se permiten no mas de 5 HUB entre el dispositivo y el host por lo cual la fila 7 solo puede ser ocupada por dispositivos.

El puerto USB transmite los datos y la alimentación del dispositivo mediante cuatro cables como se observa en la Figura 2.3, dos de ellos transmiten las señales de forma diferencial en 3 diferentes velocidades: La USB High-Speed a 480Mb/s, USB Full-Speed a 12Mb/s y USB Low-Speed a 1,5Mb/s. El reloj es transmitido por los cables diferenciales de datos bajo la codificación NRZI (Código sin retorno a cero invertido, Non return

Cuadro 2.1: Modos del puerto USB

MODO	RENDIMIENTO	APLICACIONES	CARACTERISITCAS
LOW-SPEED	Periféricos interactivos 10 - 100 kb/s	Teclado, Mouse Game pads	Bajo costo, fácil de manejar Conexión y desconexión dinámica
FULL-SPEED	Teléfono, Audio Vídeo comprimido 500 kb/s - 10 Mb/s	Micrófonos Web Cam Broadband	Bajo costo, fácil de usar Conexión y desconexión dinámica Múltiples periféricos Garantizado ancho de banda Garantizada latencia
HIGH-SPEED	Vídeo y almacenamiento 25 - 400 Mb/s	Vídeo, Imagenes Almacenamiento Broadband	Bajo costo, fácil de usar Múltiples periféricos Garantizado ancho de banda Garantizada latencia Alto ancho de banda

to zero inverted). La alimentación la llevan los cables V_{BUS} y GND, con un voltaje de 5v y potencia máxima de 2,5w.

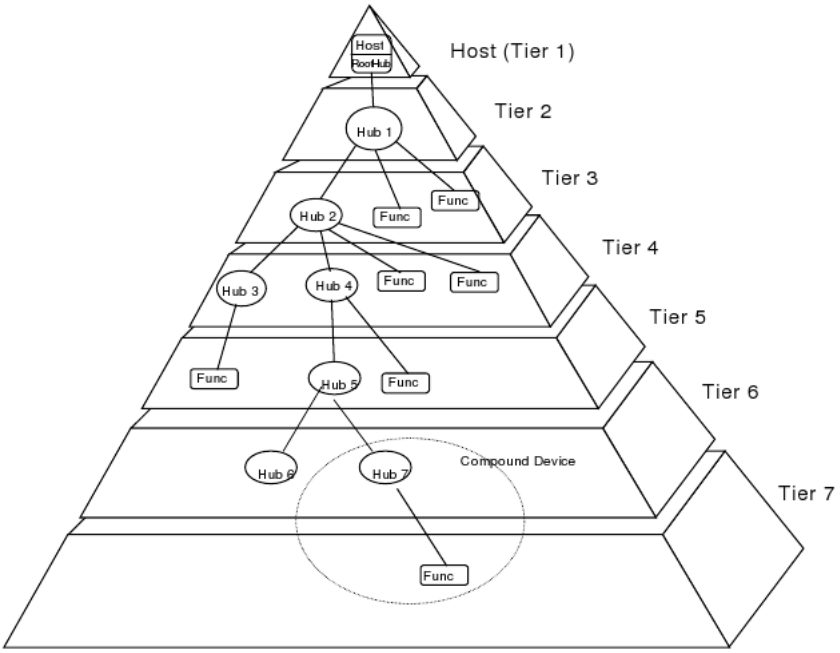
Host USB. El host USB esta conformado por hardware y por software y es quien recepciona a los dispositivos USB y a los HUBS a través del Controlador Host. En muchos casos el host USB es el computador al cual se conectan dispositivos como memorias usb, tarjetas de red o de sonido, entre otros. El host es responsable de los siguientes aspectos:

- Detectar, conectar y desconectar los dispositivos USB
- Manejar el flujo de datos entre el dispositivo usb y el host
- Recolectar datos de estado y actividades
- Provisionar la energía a los dispositivos USB conectados

Existe un software del sistema USB en el host el cual maneja la interconexión entre los dispositivos USB y el software del host. Hay cinco áreas de interconexión entre el Software del Sistema USB y el software del dispositivo:

- Enumeración y configuración del dispositivo
- Transmisión síncrona
- Transmisión asíncrona

Figura 2.2: Topología del bus USB



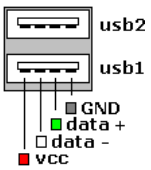
Universal Serial Bus Specifications, 2000.

- Manejo de la alimentación eléctrica
- Manejo de la información del dispositivo y el bus

Dispositivos USB. Los dispositivos USB se puede dividir en varias clases como: HUBS, interfaces humanas, impresoras, dispositivos de imágenes, dispositivos de almacenamiento masivo, dispositivos de comunicación entre otros. Cuando un dispositivo USB es conectado al host, este le asigna una dirección. Cada dispositivos USB soporta cuatro tipos de transferencias:

- **Control:** Son transferencias que se utilizan para leer información de los descriptores en los registros de los dispositivos (llamados endpoints), interpretarla y poder configurarlos.
- **Interrupción:** Usadas en los periféricos del tipo de los controladores de juegos, teclados y ratones, cuya comunicación es unidireccional y poco frecuente.

Figura 2.3: Pines USB Socket A



STACEY, JULIAN, USB, disponible en internet: URL: http://www.berklix.com, 2007.

- **Masiva:** Son transferencias no periódicas que precisan de todo el ancho de banda disponible. Utilizadas por las impresoras y los scanners.
- **Isócrona:** Dedicadas a las transferencias de telecomunicaciones, como voz o vídeo, que garantiza unas tasas de transferencia constantes. Se caracterizan porque el número de pulsos de reloj que transcurren entre la transmisión de dos caracteres es constante, por lo tanto, se está enviando información constantemente entre el host y el dispositivo.[1]

2.2. SISTEMA DE ARCHIVOS FAT32

El sistema de archivos FAT por sus siglas en ingles (File Allocation Table) se originó en la década de 1970 y en 1980 fue el sistema de archivos utilizado por el sistema operativo MS-DOS. Fue desarrollado para almacenar datos en disquetes de tamaños menores a 500KB, pero a lo largo del tiempo fue necesario el almacenamiento de datos de multimedia por lo cual se desarrollaron tres distintos tipos: FAT12, FAT16 y FAT32.

El sistema de archivos FAT32 maneja un direcciónamiento de 32 bits y direcciona clusters en los cuales están almacenados sectores del disco, por esta razón se pueden crear particiones de hasta 2TB. Este sistema cuenta con una tabla de localización la cual se encuentra en la parte inicial de la partición

2.2.1. Culsters. El elemento de menor tamaño que puede ser direccionado en un disco se denomina sector, este usualmente almacena 512 bytes o palabras, los sistemas de archivos FAT asignan uno o varios sectores a un elemento llamado Cluster que

[1]MARTINEZ DURA, Rafael J., BOLUDA, Jose A., PEREZ, Juan J., Estructura de Computadores y Periféricos, Valencia, RA-MA, 2001, 404p.

dependiendo de la capacidad del medio de almacenamiento varia su tamaño. Por ejemplo un disquete de 360KB tiene clusters de 2 sectores (1024 bytes) y un disco de mas de 2GB tiene clusters de 32K, es decir 64 sectores de 512 bytes.

Los clusters tienen su funcionalidad debido a que los archivos están conformados por sectores, secotres que no siempre están contiguos en el disco debido a que en ocasiones no se encuentra la cantidad suficientes de sectores libres contiguos para almacenar un archivo grande. El direccionamiento de muchos sectores en un archivo puede ser muy grande, es por esto que los sectores se agrupan en clusters.

Existe una lista de links donde se almacena la dirección de los clusters donde esta alojado un archivo, esta lista se denomina FAT (File Allocation Table). El tamaño de los clusters esta almacenado el el primer sector de la partición en una lista de números llamada Bios Parameter Block.

2.3. USO DEL MODULO VDRIVE1

Investigando las referencias bibliográficas del protocolo de comunicaciones del puerto USB basándose en el libro

3 TERMINAL USB A SPI

Se ha desarrollado una interfaz USB a SPI que permite enviar y recibir datos desde una terminal mediante un puerto serial emulado por USB a un dispositivo con conexión SPI. Esto con el fin enviar comandos de prueba al modulo VDRIVE1 y así conocer y entender mejor su funcionamiento para después de esto desarrollar las librerías que logren una excelente y eficiente comunicación. El diagrama de bloques de la Figura 3.1 describe su funcionamiento y el ANEXO A muestra el código del programa.

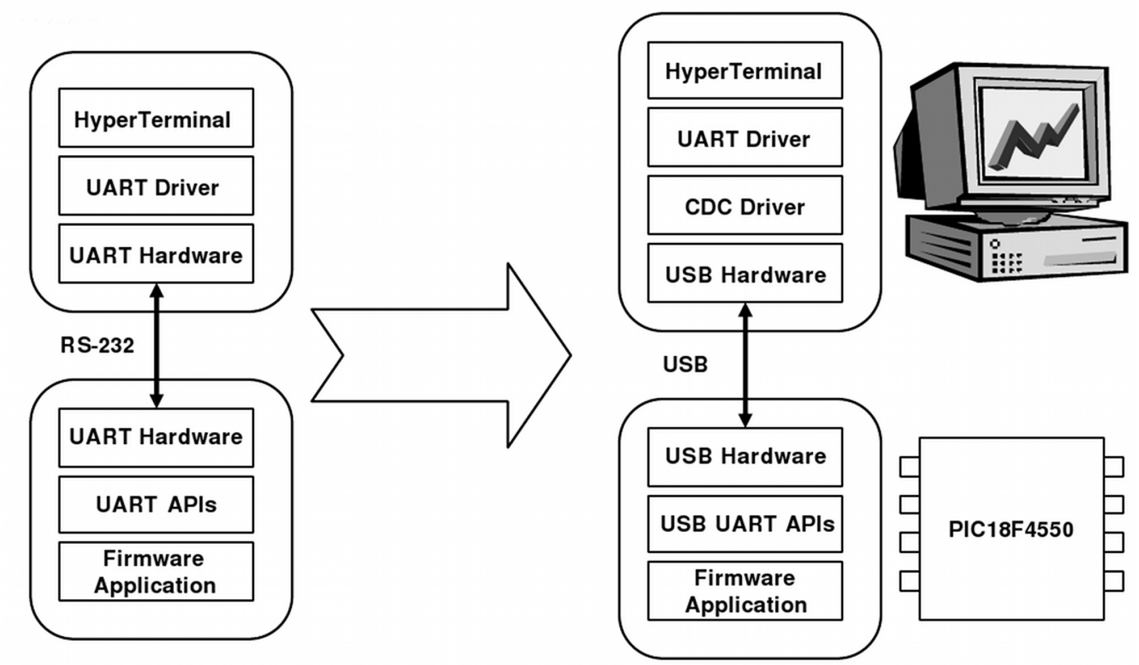
Figura 3.1: Diagrama de Interfaz Terminal USB a SPI

3.1. Emulación RS-232 bajo USB

Dado que el objetivo de esta interfaz es enviar comandos por SPI al VDRIVE1 se encontró que la manera mas sencilla de lograrlo es mediante una terminal en la cual pueden mandar datos mediante un puerto serial. Debido a que en la actualidad muchos computadores carecen de un puerto serial se utiliza en este proyecto la emulación serial RS-232 bajo USB que poseen los microcontroladores PIC18F2550 o PIC18F4550, en los cuales a su vez se le pueden programar rutinas para la comunicación SPI.

Para lograr este cometido se configura al PIC18F2550 como un dispositivo USB tipo CDC (Comunication Device Class), Dispositivo Tipo Comunicación con un reloj de 12MHz. En el gráfico de la Figura 3.2 se muestra por capas como se realiza la emulación.

Figura 3.2: Emulación RS232 bajo USB



Microchip Technology Incorporated, Emulating RS -232 Over USB with PIC18F4550, 2004.

3.1.1. Funciones para la comunicación USB USART. Las librerías proporcinadas por Microchip para la comunicación USB están escritas para funcionar en el lenguaje C18 el cual se escribe en el editor de código MPLAB de la Microchip y se compila con el software MCC18 de la misma empresa; estas herramientas son propias para trabajar con los microcontroladores serie PIC18. A continuación se estudiaran las funciones para enviar y recibir datos y una función para mantener activo el servicio de comunicación por USB.

USBTasks. Es una función que debe ser llamada constantemente para tener activos los servicios de comunicación USB y CDC. En el ejemplo de la función getsUSBUSART se muestra su uso.

mUSBUSARTIsTxTrfReady. Esta función se usa para saber si el USB esta listo para la comunicación de datos. Siempre debe usarse antes de enviar o recibir algún dato. Ejemplo:

void ejemplo1(void)

```
{
    if(mUSBUSARTIsTxTrfReady())
        putrsUSBUSART(
```

4 LIBRERÍA PARA EL MANEJO DEL VDRIVE1

Este capitulo describe la funciones implementadas en assembler para la comunicación con el modulo VDRIVE1, las cuales se realizaron con la ayuda del programa MPLAB, el compilador MPASMWIN, el debugger MPLAB SIM de la Microchip y la interfaz USB a SPI implementada en este proyecto. Estas rutinas están diseñadas para trabajar con microcontroladores serie PIC16 como el PIC16F84 o el PIC16F877. Se utiliza en todas las funciones el modo comandos cortos SCS puesto que es mas eficiente enviar y recibir datos de esta forma.

4.1. RUTINAS DE COMUNICACIÓN SPI

Estas funciones se comunican directamente con el VDRIVE1 mediante SPI usando el registro de trabajo W y realizando escritura o lectura de datos.

4.1.1. Lectura de datos VDSPIR. La función VDSPIR lee datos del VDRIVE1 por SPI, comprueba si el dato es nuevo y lo guarda en el registro de trabajo W. Coloca el Carry en 1 si el dato es viejo y en 0 si el dato es nuevo.

Ejemplo: Se desea esperar a que el modulo VDRIVE1 mande un dato nuevo y este dato mandarlo al puerto B.

```
LEER
        CALL    VDSPIR
        BTFSC   STATUS,C
        GOTO    LEER
        MOVWF   PORTB
```

4.1.2. Escritura de datos VDSPIW. La función VDSPIR toma el dato que está en el registro de trabajo W y lo envía por SPI al VDRIVE1, comprueba que el dato haya sido recibido correctamente, si no es así llama al la función VDSPIR para leer un dato y así desocupar el buffer, despues de esto vuelve a enviar el dato de entrada inicial.

Ejemplo: Se desea enviar el valor 0x0D por SPI al modulo VDRIVE1

```
MOVLW   0x0D
CALL    VDSPIW
```

4.1.3. RUTINA DE ESPERA A NO NUEVOS MENSAJES VDWNM La función VDWNM espera a que a que el modulo VDRIVE1 termine de mandar mensajes, esta rutina se usa antes de enviar comandos al VDRIVE1 para que el modulo este listo para la recepción.

4.2. RUTINAS DE CONFIGURACIÓN

Estas rutinas configuran el modo de funcionamiento del VDRIVE1, muestran información sobre el e inicializan la comunicación.

4.2.1. RUTINA DE INICIALIZACIÓN VDINIC La función VDINIC configura los pines para la conexión SPI con el VDRIVE1, manda una señal de eco para conocer si la comunicación esta establecida, configura al VDRIVE en modo SCS.

4.2.2. Monitor en datos modo ASCII VDIPAM. Esta función configura al monitor en modo datos ASCII donde los valores de entrada y salida se manejan con caracteres imprimibles ASCII. Ejemplo: se escribe en ASCII 255 lo cual quiere decir que se escribirán 255 datos en el archivo abierto.

```
CALL    VDWRF
MOVLW   '2'
CALL    VDSPIW
MOVLW   '5'
CALL    VDSPIW
MOVLW   '5'
MOVLW   0x0D
CALL    VDSPIW
```

4.2.3. Monitor en datos modo binario VDIPHM. Esta función configura al monitor en modo datos binario donde los valores de entrada y salida se manejan en binario. Ejemplo: se escribe 0x000000FF lo cual quiere decir que se escribirán 255 datos en un archivo abierto.

```
CALL    VDWRF
MOVLW   0x00
CALL    VDSPIW
MOVLW   0x00
CALL    VDSPIW
MOVLW   0x00
CALL    VDSPIW
MOVLW   0xFF
CALL    VDSPIW
MOVLW   0x0D
CALL    VDSPIW
```

4.2.4. Modo comandos cortos VDSCSM. Esta función configura al VDRIVE1 para que reciba comandos en modo SCS y responda de la misma forma

4.2.5. Mostrar la versión del firmware. Esta función le indica al VDRIVE1 que muestre la versión del firmware con la que esta trabajando.

4.3. RUTINAS PARA EL MANEJO DE ARCHIVOS

Estas rutinas permiten manejar archivos dentro de una memoria USB conectada al modulo VDRIVE1, siempre y cuando este formateada en uno de los siguientes sistemas: FAT12, FAT16 o FAT32.

4.3.1. Función para listar archivos VDDIR. Lista los archivos y carpetas del directorio actual

4.3.2. Función listar un archivo VDDIRF. Lista el archivo especificado y su tamaño. Ejemplo: Lista el archivo A.TXT con su tamaño.

```
CALL    VDDIRF
MOVLW   'A'
CALL    VDSPIW
MOVLW   '.'
CALL    VDSPIW
MOVLW   'T'
CALL    VDSPIW   Devuelve:A.TXT tttt
MOVLW   'X'
CALL    VDSPIW
MOVLW   'T'
CALL    VDSPIW
MOVLW   0x0D
CALL    VDSPIW
```

Donde tttt, una variable de 32 bits que muestra el tamaño en bytes

4.3.3. Función cambio de carpeta VDCDF. Esta función cambia de carpeta. Ejemplo: Accede a la carpeta DATA

```
CALL    VDCDF
MOVLW   'D'
CALL    VDSPIW
MOVLW   'A'
CALL    VDSPIW
MOVLW   'T'
CALL    VDSPIW
MOVLW   'A'
CALL    VDSPIW
MOVLW   0x0D
CALL    VDSPIW
```

4.3.4. Función subir carpeta VDCDF. Esta función cambia hacia arriba una carpeta

4.3.5. Función leer archivo completo VDRD Esta función lee por completo un archivo especificado. Ejemplo: Lee el contenido del archivo B.TXT y lo manda al puerto B.

```
        CALL    VDRD
        MOVLW   'B'
        CALL    VDSPIW
        MOVLW   '.'
        CALL    VDSPIW
        MOVLW   'T'
        CALL    VDSPIW
        MOVLW   'X'
        CALL    VDSPIW
        MOVLW   'T'
        CALL    VDSPIW
        MOVLW   0x0D
        CALL    VDSPIW
ESTALEYENDO
        CALL    VDSPIR
        MOVWF   PORTB
        BTFSS   STATUS,C
        GOTO    ESTALEYENDO
        SLEEP
```

4.3.6. Función borrar carpeta VDDLD. Borra una carpeta vacía especificada. Ejemplo: Borra la carpeta DATA

```
        CALL    VDDLD
        MOVLW   'D'
        CALL    VDSPIW
        MOVLW   'A'
        CALL    VDSPIW
        MOVLW   'T'
        CALL    VDSPIW
        MOVLW   'A'
        CALL    VDSPIW
        MOVLW   0x0D
        CALL    VDSPIW
```

4.3.7. Función crear carpeta VDMKD. Crea una nueva carpeta, también opcionalmente se puede definir la fecha y hora de su creación en formato 32 bits FAT. Ejemplo: Crea una carpeta llamada DATA con la fecha 2008-11-11 00:00:00

```
CALL    VDMKD
MOVLW   'D'
CALL    VDSPIW
MOVLW   'A'
CALL    VDSPIW
MOVLW   'T'
CALL    VDSPIW
MOVLW   'A'
CALL    VDSPIW
MOVLW   0x20
CALL    VDSPIW
MOVLW   0x38
CALL    VDSPIW
MOVLW   0x5B
CALL    VDSPIW
MOVLW   0x00
CALL    VDSPIW
MOVLW   0x00
CALL    VDSPIW
MOVLW   0x0D
CALL    VDSPIW
```

4.3.8. Función borrar archivo VDDLF. Borra el archivo especificado. Ejemplo: Borrar el archivo B.TXT

```
CALL    VDDDLF
MOVLW   'B'
CALL    VDSPIW
MOVLW   '.'
CALL    VDSPIW
MOVLW   'T'
CALL    VDSPIW
MOVLW   'X'
CALL    VDSPIW
MOVLW   'T'
CALL    VDSPIW
MOVLW   0x0D
CALL    VDSPIW
```

4.3.9. Función escribir en archivo VDWRF. Escribe un numero de bytes especificados en el archivo que se encuentre abierto, el numero de bytes debe especificarse en forma binaria con una longitud de 32bits (4bytes). Ejemplo: Escribe las letras JP dentro del archivo abierto

```
CALL    VDWRF
MOVLW   0x00
CALL    VDSPIW
MOVLW   0x00
CALL    VDSPIW
MOVLW   0x00
CALL    VDSPIW
MOVLW   0x02
CALL    VDSPIW
MOVLW   0x0D
CALL    VDSPIW
MOVLW   'J'
CALL    VDSPIW
MOVLW   'P'
CALL    VDSPIW
```

4.3.10. Función abrir archivo para escritura VDOPW. ; Abre un archivo especificado para escribir o crea un archivo nuevo, opcional se puede también especificar la fecha y hora de creación. Ejemplo: abre el archivo C.DAT para escribir en el.

```
CALL    VDOPW
MOVLW   'C'
CALL    VDSPIW
MOVLW   '.'
CALL    VDSPIW
MOVLW   'D'
CALL    VDSPIW
MOVLW   'A'
CALL    VDSPIW
MOVLW   'T'
CALL    VDSPIW
MOVLW   0x0D
CALL    VDSPIW
```

4.3.11. Función cerrar archivo VDCLF. Cierra el archivo que se encuentra abierto, debe especificarse el archivo. Ejemplo: cierra el archivo C.DAT

```
CALL    VDCLF
MOVLW   'C'
CALL    VDSPIW
MOVLW   '.'
CALL    VDSPIW
MOVLW   'D'
CALL    VDSPIW
MOVLW   'A'
CALL    VDSPIW
MOVLW   'T'
CALL    VDSPIW
MOVLW   0x0D
CALL    VDSPIW
```

4.3.12. Función leer archivo VDRDF. . Lee el numero de bytes especificados del archivo que se encuentra abierto el numero de bytes debe especificarse en forma binaria con una longitud de 32bits (4bytes). Ejemplo: lee dos bytes del archivo abierto, el primero lo saca por el puerto B y el segundo lo saca por el puerto D.

```
CALL    VDRDF
MOVLW   0x00
CALL    VDSPIW
MOVLW   0x00
CALL    VDSPIW
MOVLW   0x00
CALL    VDSPIW
MOVLW   0x02
CALL    VDSPIW
MOVLW   0x0D
CALL    VDSPIW
CALL    VDSPIR
MOVWF   PORTB
CALL    VDSPIR
MOVWF   PORTD
```

4.3.13. Función renombrar VDREN. Esta función renombra un archivo o una carpeta especificado. Ejemplo: renombra el archivo A.TXT por B.TXT

```
CALL    VDREN
MOVLW   'A'
CALL    VDSPIW
MOVLW   '.'
CALL    VDSPIW
MOVLW   'T'
CALL    VDSPIW
MOVLW   'X'
CALL    VDSPIW
MOVLW   'T'
CALL    VDSPIW
MOVLW   0x20
CALL    VDSPIW
MOVLW   'B'
CALL    VDSPIW
MOVLW   '.'
CALL    VDSPIW
MOVLW   'T'
CALL    VDSPIW
MOVLW   'X'
CALL    VDSPIW
MOVLW   'T'
CALL    VDSPIW
MOVLW   0x0D
CALL    VDSPIW
```

4.3.14. Función abrir archivo para lectura VDOPR. Abre un archivo para su lectura, opcional se puede especificar la fecha de acceso en formato 16 bits FAT. Ejemplo: se abre el archivo A.TXT y se le coloca como fecha de acceso 0x355B (2008-11-11)

```
CALL    VDOPR
MOVLW   'A'
CALL    VDSPIW
MOVLW   '.'
CALL    VDSPIW
MOVLW   'T'
CALL    VDSPIW
MOVLW   'X'
CALL    VDSPIW
MOVLW   'T'
CALL    VDSPIW
MOVLW   0x20
CALL    VDSPIW
MOVLW   0x38
CALL    VDSPIW
MOVLW   0x5B
CALL    VDSPIW
MOVLW   0x0D
CALL    VDSPIW
```

4.3.15. Función mover puntero VDSEK. Esta función mueve el puntero a la posición especificada en el archivo que se encuentra abierto, la posición del puntero debe especificarse en forma binaria con una longitud de 32bits (4bytes). Ejemplo: Mueve el puntero a la posición 0xFF (255) del archivo abierto

```
CALL    VDSEK
MOVLW   0x00
CALL    VDSPIW
MOVLW   0x00
CALL    VDSPIW
MOVLW   0x00
CALL    VDSPIW
MOVLW   0xFF
CALL    VDSPIW
MOVLW   0x0D
CALL    VDSPIW
```

4.3.16. Función espacio libre VDFS y VDFSE. Estas funciones muestran el espacio libre de una memoria USB, la función VDFS se usa para memorias con espacio libre menor o igual a 4GB, devuelve un valor de 32 bits que indica el espacio libre en bytes. La función VDFSE se usa cuando el espacio libre puede ser mayor a 4GB, devuelve un valor de 48 bits que indica el espacio libre en bytes. Ejemplo:

CALL VDFS
Devuelve:tttt

Donde tttt es un dato de 32bits que indica el espacio libre en bytes

CALL VDFSE
Devuelve:tttttt

Donde tttttt es un dato de 48bits que indica el espacio libre en bytes

4.4. RUTINAS PARA EL MANEJO DE LA ENERGÍA

Estas funciones permiten manejar la energía del dispositivo USB.

4.4.1. Función suspender disco VDSUD. Esta función coloca a la memoria USB en modo de bajo consumo, suele usarse para también como una forma de extracción segura de ella.

4.4.2. Función despertar disco VDWKD. Esta función devuelve a la memoria USB del modo bajo consumo al modo normal para poder operar nuevamente en ella.

4.5. RUTINAS PARA DE COMPROBACIÓN DE RESPUESTA

Estas rutinas comprueban estados en el VDRIVE1 o la respuesta a determinado tipo de comando.

4.5.1. Función leer respuesta de comando VDRRC. Esta función lee la respuesta dada por un comando, coloca al Carry en 0 si el comando fue aceptado corectamente y en 1 si no lo fue.

4.5.2. Función leer comando fallido VDRFC Esta función lee la salida de datos del VDRIVE1 y la compara con la secuencia de respuesta de comando fallido, es muy útil para conocer cuando un archivo ha llegado a su fin. Coloca al Carry en 0 si la

respuesta es comando fallido y en 1 si no lo es. Ejemplo: Se leerá los datos de un pequeño archivo abierto que contiene el texto "1234567890"de 3 en 3 caracteres donde se desea conocer cuando el archivo ha llegado su fin.

```
LECTURA
      CALL    VDRDF       ; Llama a la función leer datos de archivo especificando
      MOVLW   0x00        ; que leerá 3 bytes.
      CALL    VDSPIW
      MOVLW   0x00
      CALL    VDSPIW
      MOVLW   0x00
      CALL    VDSPIW
      MOVLW   0x03
      CALL    VDSPIW
      MOVLW   0x0D
      CALL    VDSPIW
      MOVLW   0x03
      MOVWF   CONTCAR     ; Inicializa al contador de caracteres CONTCAR en 3
LECTURA1
      CALL    VDSPIRN     ; Llama a la función leer dato nuevo
      CALL    MLETRA      ; Manda el dato a la matriz de LEDS
      DECFSZ  CONTCAR,F   ; Decrementa a CONTCAR para saber cuantos caracteres
      GOTO    LECTURA1    ; se han leído, salta a LECTURA1 para continuar
      CALL    VDRCF       ; Si acabo de leer los 3 caracteres detecta si el
      BTFSC   STATUS,C    ; archivo llego a su fin.
      GOTO    LECTURA     ; Si no ha llegado a su fin salta a LECTURA
                          ; para continuar con los siguientes 3 caracteres
      GOTO    FINARCHIVO  ; Si termino de leer el archivo salta a FINARCHIVO
```

4.5.3. Función detectar presencia memoria USB VDUSBP. Esta función detecta si existe una memoria USB conectada o no al VDRIVE1, devuelve al Carry en 1 si la memoria no fue detectada y en 0 si lo fue.

4.6. CODIGO DE LA LIBRERIA

A continuación se muestra el código que compone las funciones de la librería que maneja al VDRIVE1, en el se indica como se debe configurar el modulo para que funcione correctamente con el microcontrolador, comentarios amplios en las instrucciones usadas para su fácil entendimiento y funciones acompañadas con ejemplos prácticos para su fácil aplicación.

```
;************************************ Librería "VDRIVE.INC" ************************************
; Esta librería esta diseñada para comunicarse con el modulo VDRIVE1 que cuenta con el chip
; Vinculum VNC1L-1A mediante SPI. Maneja las siguientes funciones:
;
; VDINIC: Inicializa la comunicación con el VDRIVE1
; VDSPIW: Escribe datos en el VDRIVE1
; VDSPIR: Lee un dato del VDRIVE1
; VDSPIRN: Lee un dato nuevo del VDRIVE1
; VDWNM: Espera a que el VDRIVE1 termine de mandar datos
; VDWWNM: Espera varias veces a que el VDRIVE termine de mandar datos
; VDIPAM: Monitor en datos modo ASCII
; VDIPHM: Monitor en datos modo binario
; VDSCSM: Modo comandos cortos
; VDFWV: Muestra la versión del firmware del VDRIVE1
; VDDIR: Lista los archivos y carpetas del directorio actual
; VDDIRF: Lista el archivo especificado y su tamaño
; VDCDF: Cambia de carpeta
; VDCDA: Cambia hacia arriba una carpeta
; VDRD: Lee por completo un archivo
; VDDLD: Borra una carpeta
; VDMKD: Crea una nueva carpeta
; VDDLF: Borra un archivo
; VDWRF: Escribe un numero de bytes especificados en el archivo que se encuentre abierto
; VDOPW: Abre un archivo para escribir o crea un archivo nuevo
; VDCLF: Cierra el archivo que se encuentra abierto
; VDRDF: Lee el numero de bytes especificados del archivo que se encuentra abierto
; VDREN: Renombra un archivo o una carpeta
; VDOPR: Abre un archivo para su lectura
; VDSEK: Mueve el puntero a la posición especificada en el archivo que se encuentra abierto
; VDFS: Muestra el espacio libre en bytes del disco si este es menor a 4GB
; VDFSE: Muestra el espacio libre en bytes del disco
; VDSUD: Coloca a la memoria USB en suspensión
; VDWKD: Saca a la memoria USB de la suspensión
; VDRRC: Comprueba respuesta de comando
; VDRCF: Comprueba comando fallido
; VDUSBP: Detecta la presencia de la memoria USB
; VDRET: Realiza un retardo de 16 ciclos
;
; Estas rutinas están diseñadas para trabajar en microcontroladores serie PIC con un reloj de hasta
; 96MHz donde las rutinas necesiten 4 ciclos de reloj para su ejecución. Si el reloj es superior
; es necesario colocar retardos en las rutinas VDSPIR y VDSPIW para aumentar el tiempo de la onda
; de clock a un mínimo de 83ns.
;
; Para el manejo de entrada y salida en archivos el firmware requiere que se los abra usando las
; funciones VDOPW o VDOPR escribiendo en ellos con la función VDWRF o leyendo con la función VDRDF
; y luego cerrándolo con la función VDCLF. Solo un archivo puede ser abierto al tiempo. Si un
; archivo es abierto para escritura, se puede escribir o leer de el. Si un archivo es abierto para
; lectura solo se podrá leer de el.
;
; El puntero en el archivo se mantiene desde el lugar donde fue leído o escrito por ultima vez. La
; función VDSEK puede ser usada para mover el puntero dentro de un archivo. Cuando se usa la función
; VDOPW sobre un archivo existente la función VDWRF empezara a escribir a partir del último dato.
;
; Mientras se escriben datos en el disco, el firmware puede demorase un cierto tiempo para encontrar
; un cluster vacío debido a la fragmentación de datos, esto suele ocurrir cuando se usan las
; funciones VDWRF para operaciones de escritura o las funciones VDOPW o VDMKD.
;
; Si no se especifica la fecha y hora en la creación de una carpeta o en un archivo el firmware
; colocara por defecto el dato 0x31940000 (2004.12.04 00:00:00). Si no se especifica la fecha de
; lectura al usar la función VDOPR se mantendrá la fecha de la última lectura.
;
; Nombres largos en sistemas de archivo FAT32 no son soportados, los nombres para los archivos
; pueden tener de 1 a 8 caracteres y opcional seguidos de un punto con una extensión de archivo
; de 3 caracteres, deben estar en mayúsculas, si se los inserta en minúsculas el firmware los
; pasara a mayúsculas usando el código OEM437. Se pueden entrar letras, números y los siguientes
```

```
; caracteres:
; $ % ‘ - _ @ ~ ! ( ) { } ^ # &
;
; Ejemplo: "TEXTO_01.TXT" o "JUAN.WAV"
;
;
; Conexión del modulo VDRIVE1 al microcontrolador PIC16F877A
;
;           -------
;                  \                 ----------------------
;                   \               |                      |
;             V      \              |                      |
;             D   6 -|-- CSE ----|Pin18-RC3      PIC    |
;             R   5 -|-- CLK ----|Pin17-RC2      16F    |
;             I   4 -|-- SDI ----|Pin15-RC0     877A    |
;             V   3 -|-- VCC     |                      |
;             E   2 -|-- SDO ----|Pin16-RC1             |
;             1   1 -|-- GND     |                      |
;                    /           |                      |
;                   /            |______________________|
;           -------/
;
;
;
;     ================================================================================
;     ||                              PROYECTO DE GRADO                             ||
;     ||                                                                            ||
;     ||     ESTUDIO Y DISEÑO DE ALGORITMOS E INTERFAZ ELECTRÓNICA DE COMUNICACIÓN      ||
;     ||    ENTRE MICROCONTROLADORES SERIE PIC Y DISPOSITIVOS DE ALMACENAMIENTO USB     ||
;     ||                                                                            ||
;     ||                           Juan Pablo Ruiz Rosero                           ||
;     ||                           Ingeniería Electrónica                           ||
;     ||                            Universidad de Nariño                           ||
;     ||                                Octubre 2008                                ||
;     ================================================================================
;
;
;*************************************************************************************************


; ZONA DE DATOS ***********************************************************************************
;*************************************************************************************************
;*************************************************************************************************

        CBLOCK
        VDSPICONT               ; Define direcciones en memoria para las variables
        VDSPIDATO
        VDSPIDATO2
        VDINICC
        VDRDSCONT
        VDSRFL
        VDWNMCONT
        VDRETCONT
        ENDC

#DEFINE VDSPIO  PORTC,0         ; Define el pin para la salida de datos SPI (SPI OUT)
#DEFINE VDSPII  PORTC,1         ; Define el pin para la entrada de datos SPI (SPI IN)
#DEFINE VDSPIC  PORTC,2         ; Define el pin para la salida del clock del SPI (SPI CLOCK)
#DEFINE VDSPIE  PORTC,3         ; Define el pin para la habilitación del dispositivo SPI (SPI ENABLE)
#DEFINE VDSRN   VDSRFL,0        ; Define una bandera para conocer
#DEFINE VDSWA   VDSRFL,1        ; Define una bandera


; ZONA DE FUNCIONES *******************************************************************************
```

```
;*************************************************************************************************
;*************************************************************************************************

;FUNCION DE INICIALIZACION*********************************************************************
; Esta función configura los pines para la conexión SPI con el VDRIVE1, manda una señal de eco
; para conocer si la comunicación esta establecida, configura al VDRIVE en modo SCS y devuelve
; un 0 si se logro establecer la comunicación y un 1 si no se logro.
VDINIC
        BSF     STATUS,RP0              ; Pone a 1 el bit 5 del STATUS. Acceso al Banco 1.
        BCF     VDSPIO                  ; Coloca a SPI OUT como salida
        BSF     VDSPII                  ; Coloca a SPI IN como entrada
        BCF     VDSPIC                  ; Coloca a SPI CLOCK como salida
        BCF     VDSPIE                  ; Coloca a SPI ENABLE como salida
        BCF     STATUS,RP0              ; Acceso al banco 0

        BCF     VDSPIO                  ; Coloca en cero las salidas
        BCF     VDSPIC
        BCF     VDSPIE

        CALL    Retardo_100ms           ; Llama un retardo para que el modulo VDRIVE1 este listo


VDINIC1
        CALL    VDWNM
VDINIC2
        MOVLW   'E'                     ; Manda el comando de eco "E" al VDRIVE1
        CALL    VDSPIW
        MOVLW   0x0D
        CALL    VDSPIW

VDINIC3
        CALL    VDSPIR                  ; Lee los datos del VDRIVE1
        BTFSC   STATUS,C                ; Comprueba que el dato sea nuevo
        GOTO    VDINIC3                 ; El dato no es nuevo, vuelve a leer
        SUBLW   'E'                     ; El dato es nuevo, comprueba que sea igual a "E"
        BTFSS   STATUS,Z
        GOTO    VDINIC2                 ; El dato no es igual a "E" vuelve a mandar la señal de eco
        CALL    VDSCSM                  ; El dato si es igual a "E", coloca el monitor en modo comandos
        CALL    VDRRC
        BCF     STATUS,C                ; cortos, coloca el 0 el Carry para informar que se logro
        RETURN                          ; establecer la comunicación con el VDRIVE1 y termina la función.

; FIN FUNCION DE INICIALIZACION ****************************************************************


;FUNCION DE ESCRITURA SPI **********************************************************************
; Esta función toma el dato que esta en W y lo envía por SPI al VDRIVE1

VDSPIW
        MOVWF   VDSPIDATO2              ; Almacena el dato que esta W en el registro VDSPIDATO2
VDSPIWR
        MOVF    VDSPIDATO2,W            ; Lee el dato del registro VDSPIDATO2 y lo pasa al registro
        MOVWF   VDSPIDATO               ; VDSPIDATO

        BCF     VDSPIC                  ; Se cerciora de que la señal de reloj empiece en cero

; CLOCK 1
        BSF     VDSPIE                  ; Coloca en 1 a SPI ENABLE para habilitar el modulo VDRIVE1
        BSF     VDSPIO                  ; Coloca en 1 a SPI OUT para indicar el bit de arranque

        BSF     VDSPIC                  ; Manda un pulso al reloj
        BCF     VDSPIC

; CLOCK 2
        BCF     VDSPIO                  ; Coloca en 0 a SPI OUT para indicar que realizará escritura
```

```
                                        ; de datos
        BSF     VDSPIC                  ; Manda un pulso al reloj
        BCF     VDSPIC

; CLOCK 3
        BCF     VDSPIO                  ; Coloca en 0 a SPI OUT para indicar que escribirá en el
                                        ; registro de datos
        BSF     VDSPIC                  ; Manda un pulso al reloj
        BCF     VDSPIC

; CLOCK 4 A 11
        MOVLW   0x08                    ; Inicializa a VDSPICONT en 0x08 para contar los 8 bits
        MOVWF   VDSPICONT               ; de datos que va a leer
VDSPIW1
        RLF     VDSPIDATO,F             ; Rota a la izquierda el registro VDSPIDATO a través del
                                        ; Carry para colocar al MSB en el Carry
        BTFSC   STATUS,C                ; Testea el valor del Carry
        GOTO    VDSPIWUNO               ; Como el Carry esta en uno salta a VDSPIWUNO
        BCF     VDSPIO                  ; Como el Carry esta en 0 coloca en 0 a SPI OUT
        GOTO    VDSPIW2                 ; Salta a VDSPIW2
VDSPIWUNO
        BSF     VDSPIO                  ; Coloca a SPI OUT en 1
VDSPIW2

        BSF     VDSPIC                  ; Manda un pulso al reloj
        BCF     VDSPIC

        DECFSZ  VDSPICONT,F             ; Decrementa el contador VDSPICONT y testea si esta en cero
        GOTO    VDSPIW1                 ; Si no esta en cero continua con la escritura del siguiente bit

; CLOCK 12
        BCF     VDSWA                   ; Coloca la bandera de escritura aceptada en 0
        BTFSC   VDSPII                  ; Testea la señal SPI IN
        BSF     VDSWA                   ; Si SPI IN esta en 1 coloca la bandera de escritura aceptada en
                                        ; 1 y si no la deja en 0

        BSF     VDSPIC                  ; Manda un pulso al reloj
        BCF     VDSPIC

; CLOCK 13
        BCF     VDSPIO                  ; Coloca la salida SPI OUT en cero
        BCF     VDSPIE                  ; Coloca en 0 a SPI ENABLE para deshabilitar el modulo VDRIVE1

        BSF     VDSPIC                  ; Manda un pulso al reloj
        BCF     VDSPIC

        BTFSS   VDSWA                   ; Testea la bandera de escritura aceptada
        RETURN                          ; Si esta en cero termina la escritura
        CALL    VDSPIR                  ; Si esta en uno la lectura no fue aceptada, lee datos del
        GOTO    VDSPIWR                 ; VDRIVE1 para desocupar el buffer e intenta escribir
                                        ; nuevamente el dato.
;FIN FUNCION DE ESCRITURA SPI ***************************************************************


;FUNCION DE LECTURA *************************************************************************
;Esta función lee datos del VDRIVE1 por SPI, comprueba si el dato es nuevo y lo guarda en W
;Coloca el Carry así:
;Carry = 1  Si el dato es viejo
;Carry = 0  Si el dato es nuevo

VDSPIR

        BCF     VDSPIC                  ; Se cerciora de que la señal de reloj empiece en cero

; CLOCK 1
```

```
        BSF     VDSPIE                  ; Coloca en 1 a SPI ENABLE para habilitar el modulo VDRIVE1
        BSF     VDSPIO                  ; Coloca en 1 a SPI OUT para indicar el bit de arranque

        BSF     VDSPIC                  ; Manda un pulso al reloj
        BCF     VDSPIC

; CLOCK 2
        BSF     VDSPIO                  ; Coloca en 1 a SPI OUT para indicar que realizará una lectura
                                        ; de datos
        BSF     VDSPIC                  ; Manda un pulso al reloj
        BCF     VDSPIC

; CLOCK 3
        BCF     VDSPIO                  ; Coloca en 0 a SPI OUT para indicar que leerá del registro
                                        ; de datos

        BSF     VDSPIC                  ; Manda un pulso al reloj
        BCF     VDSPIC

; CLOCK 4 A 11
        CLRF    VDSPIDATO               ; Limpia el registro donde se guardará el dato leído
        MOVLW   0x08                    ; Inicializa a VDSPICONT en 0x08 para contar los 8 bits
        MOVWF   VDSPICONT               ; de datos que va a leer
VDSPIR1
        BCF     STATUS,C                ; Supone que el dato de entrada es cero por lo cual coloca en
                                        ; cero al bit de Carry
        BTFSC   VDSPII                  ; Testea si la señal de entrada SPI IN esta en uno o en cero
        BSF     STATUS,C                ; Como la señal de entrada esta en uno coloca al Carry en uno
        RLF     VDSPIDATO,F             ; Como la señal de entrada esta en cero salta y deja al Carry
                                        ; en cero y lo rota a la izquierda a través del registro
                                        ; VDSPIDATO, con MSB primero
        BSF     VDSPIC                  ; Manda un pulso al reloj
        BCF     VDSPIC

        DECFSZ  VDSPICONT,F             ; Decrementa el contador VDSPICONT y testea si esta en cero
        GOTO    VDSPIR1                 ; Si no esta en cero continua con la lectura del siguiente bit

; CLOCK 12                              ; Si esta en cero sigue con el CLOCK12
        BCF     VDSRN                   ; Coloca la bandera de nueva respuesta en 0
        BTFSC   VDSPII                  ; Testea la señal de entrada SPI IN
        BSF     VDSRN                   ; Si SPI IN esta en 1 coloca la bandera de nueva respuesta en
                                        ; 1 y si no la deja en 0

        BSF     VDSPIC                  ; Manda un pulso al reloj
        BCF     VDSPIC

; CLOCK 13
        BCF     VDSPIO                  ; Coloca la salida SPI OUT en cero
        BCF     VDSPIE                  ; Coloca en 0 a SPI ENABLE para deshabilitar el modulo VDRIVE1

        BSF     VDSPIC                  ; Manda un pulso al reloj
        BCF     VDSPIC

        MOVF    VDSPIDATO,W             ; Coloca el dato leído en el registro de trabajo W
        BCF     STATUS,C                ; Coloca el dato de la bandera VDSRN en el bit de Carry
        BTFSC   VDSRN
        BSF     STATUS,C
        RETURN                          ; Termina la lectura
;FIN DE LA FUNCION DE LECTURA SPI ***********************************************************

;FUNCION DE LECTURA DE DATOS NUEVOS *********************************************************
; Esta función lee datos del VDRIVE1 por SPI y espera a que el dato sea nuevo finalizar. Guarda
; el dato leído en el registro de trabajo W.
VDSPIRN
        CALL    VDSPIR          ; Lee un dato del VDRIVE1
```

```
        BTFSC   STATUS,C        ; Testea si el dato es nuevo o viejo
        GOTO    VDSPIRN         ; Si es viejo salta a VDSPIRN
        RETURN                  ; Si es nuevo termina la función


;FUNCION ESPERA QUE EL VDRIVE TERMINE DE MANDAR MENSAJES ******************************************
; Esta función lee la salida del VDRIVE1 hasta que el no tenga mensajes nuevos por enviar
VDWNM
        CALL    VDSPIR                  ; Realiza la lectura del VDRIVE1
        BTFSS   STATUS,C                ; Testea si el dato es nuevo
        GOTO    VDWNM                   ; Si el dato es nuevo sigue leyendo
        RETURN                          ; Si el dato no es nuevo termina la función

;FUNCION ESPERA VARIAS VECES A QUE EL VDRIVE TERMINE DE MANDAR MENSAJES*****************************
; Cuando se inserta un dispositivo nuevo en el VDRIVE1 este suele demorace un poco en responder
; es por eso que es necesario esperar varias veces a que el modulo este listo para recivir nuevas
; instrucciones, esta función espera 100 veces a que el modulo este listo.
VDWWNM
        CALL    VDRET                   ; Llama a un retardo de 16 ciclos
        MOVLW   d'100'                  ; Inicializa en contador VDWNMCONT en 100
        MOVWF   VDWNMCONT
VDWWNM1
        CALL    VDRET                   ; Llama a un retardo de 16 ciclos
        CALL    VDSPIR                  ; Realiza la lectura del VDRIVE1
        BTFSS   STATUS,C                ; Testea si el dato es nuevo
        GOTO    VDWWNM                  ; Si el dato es nuevo sigue leyendo
        DECFSZ  VDWNMCONT,F             ; Si el dato es viejo decrementa al contador
        GOTO    VDWWNM1                 ; Si no ha llegado a cero salta a VDWWNM1
        RETURN                          ; Si ha llegado a cero termina la función


;FUNCION VDIPHM, MONITOR EN DATOS MODO BINARIO ***************************************************
; Configura al monitor en modo datos binario donde los valores de entrada y salida se manejan en binario
; Ejemplo:
;       CALL    VDWRF
;       MOVLW   0x00            ; Se escribe 0x000000FF lo cual quiere decir que
;       CALL    VDSPIW          ; se escribirán 255 datos en el archivo abierto
;       MOVLW   0x00
;       CALL    VDSPIW
;       MOVLW   0x00
;       CALL    VDSPIW
;       MOVLW   0xFF
;       CALL    VDSPIW
;       MOVLW   0x0D
;       CALL    VDSPIW
;*****************************************************************************************************
VDIPHM
        CALL    VDWNM
        MOVLW   0x91
        CALL    VDSPIW
        MOVLW   0x0D
        CALL    VDSPIW
        RETURN


;FUNCION VDIPAM, MONITOR EN DATOS MODO ASCII *****************************************************
; Configura al monitor en modo datos ASCII donde los valores de entrada y salida se manejan con
; caracteres imprimibles ASCII
; Ejemplo:
;       CALL    VDWRF
;       MOVLW   '2'             ; Se escribe en ASCII 255 lo cual quiere decir que
;       CALL    VDSPIW          ; se escribirán 255 datos en el archivo abierto
;       MOVLW   '5'
;       CALL    VDSPIW
```

```
;       MOVLW   '5'
;       CALL    VDSPIW
;       MOVLW   0x0D
;       CALL    VDSPIW
;*****************************************************************************************************
VDIPAM
        CALL    VDWNM
        MOVLW   0x90
        CALL    VDSPIW
        MOVLW   0x0D
        CALL    VDSPIW
        RETURN


;MONITOR EN MODO COMANDOS CORTOS ************************************************************************
; Por defecto el VDRIVE1 asume comandos extendidos, con esta función se cambia al modo comandos
; cortos para que las rutinas siguientes sean mas cortas
VDSCSM
        CALL    VDWNM
        MOVLW   'S'
        CALL    VDSPIW
        MOVLW   'C'
        CALL    VDSPIW
        MOVLW   'S'
        CALL    VDSPIW
        MOVLW   0x0D
        CALL    VDSPIW
        RETURN


;COMANDO FMV ********************************************************************************************
; Muestra la versión del Frimware del VDRIVE1
; Ejemplo
;       CALL    VDFWV
;       Devuelve:
;       MAIN 03.65VDAPF
;       RPRG 1.00R
;*****************************************************************************************************
VDFWV
        CALL    VDWNM
        MOVLW   0x13
        CALL    VDSPIW
        MOVLW   0x0D
        CALL    VDSPIW
        RETURN


;COMANDO DIR ********************************************************************************************
; Lista los archivos y carpetas del directorio actual
VDDIR
        CALL    VDWNM
        MOVLW   0x01
        CALL    VDSPIW
        MOVLW   0x0D
        CALL    VDSPIW
        RETURN


;COMANDO DIR SOBRE ARCHIVO ******************************************************************************
; Lista el archivo especificado y su tamaño
; Ejemplo: Lista el archivo A.TXT con su tamaño
        ;       CALL    VDDIRF
        ;       MOVLW   'A'
        ;       CALL    VDSPIW
        ;       MOVLW   '.'
```

```
        ;       CALL    VDSPIW
        ;       MOVLW   'T'
        ;       CALL    VDSPIW
        ;       MOVLW   'X'
        ;       CALL    VDSPIW
        ;       MOVLW   'T'
        ;       CALL    VDSPIW
        ;       MOVLW   0x0D
        ;       CALL    VDSPIW
        ;       Devuelve:A.TXT tttt
        ;       Donde tttt, una variable de 32 bits que muestra el tamaño en bytes
;*****************************************************************************************************
VDDIRF
        CALL    VDWNM
        MOVLW   0x01
        CALL    VDSPIW
        MOVLW   0x20
        CALL    VDSPIW
        RETURN


;COMANDO CD SOBRE CARPETA*************************************************************************
; Cambia de carpeta
; Ejemplo: Accede a la carpeta DATA
;       CALL    VDCDF
;       MOVLW   'D'
;       CALL    VDSPIW
;       MOVLW   'A'
;       CALL    VDSPIW
;       MOVLW   'T'
;       CALL    VDSPIW
;       MOVLW   'A'
;       CALL    VDSPIW
;       MOVLW   0x0D
;       CALL    VDSPIW
;*****************************************************************************************************
VDCDF
        CALL    VDWNM
        MOVLW   0x02
        CALL    VDSPIW
        MOVLW   0x20
        CALL    VDSPIW
        RETURN


;COMANDO CD.. ************************************************************************************
; Cambia hacia arriba una carpeta
VDCDA
        CALL    VDWNM
        MOVLW   0x02
        CALL    VDSPIW
        MOVLW   0x20
        CALL    VDSPIW
        MOVLW   0x2E
        CALL    VDSPIW
        MOVLW   0x2E
        CALL    VDSPIW
        MOVLW   0x0D
        CALL    VDSPIW
        RETURN


;COMANDO RD **************************************************************************************
; Lee por completo un archivo
; Ejemplo: Lee el contenido del archivo B.TXT y lo manda al puerto B
```

```
;       CALL    VDRD
;       MOVLW   'B'
;       CALL    VDSPIW
;       MOVLW   '.'
;       CALL    VDSPIW
;       MOVLW   'T'
;       CALL    VDSPIW
;       MOVLW   'X'
;       CALL    VDSPIW
;       MOVLW   'T'
;       CALL    VDSPIW
;       MOVLW   0x0D
;       CALL    VDSPIW
;ESTALEYENDO
;       CALL    VDSPIR
;       MOVWF   PORTB
;       BTFSS   STATUS,C
;       GOTO    ESTALEYENDO
;       SLEEP
;
;       Devuelve byte por byte los datos que contiene el archivo B.TXT hasta llegar a su fin
;       se usa la función VDSPIR para obtener estos datos y se esta pendiente del Carry para
;       conocer cuando llegue el fin del archivo.
;*************************************************************************************************
VDRD
        CALL    VDWNM
        MOVLW   0x04
        CALL    VDSPIW
        MOVLW   0x20
        CALL    VDSPIW
        RETURN


;COMANDO DLD ************************************************************************************
; Borra una carpeta
; Ejemplo: Borra la carpeta DATA
;       CALL    VDDLD
;       MOVLW   'D'
;       CALL    VDSPIW
;       MOVLW   'A'
;       CALL    VDSPIW
;       MOVLW   'T'
;       CALL    VDSPIW
;       MOVLW   'A'
;       CALL    VDSPIW
;       MOVLW   0x0D
;       CALL    VDSPIW
;*************************************************************************************************
VDDLD
        CALL    VDWNM
        MOVLW   0x05
        CALL    VDSPIW
        MOVLW   0x20
        CALL    VDSPIW
        RETURN


;COMANDO MKD ************************************************************************************
; Crea una nueva carpeta, también se puede definir la fecha y hora de su creación en formato 32 bits FAT
; Ejemplo: Crea una carpeta llamada DATA con la fecha 2008-11-11 00:00:00
;       CALL    VDMKD
;       MOVLW   'D'
;       CALL    VDSPIW
;       MOVLW   'A'
;       CALL    VDSPIW
```

```
;       MOVLW   'T'
;       CALL    VDSPIW
;       MOVLW   'A'
;       CALL    VDSPIW
;       MOVLW   0x20
;       CALL    VDSPIW
;       MOVLW   0x38
;       CALL    VDSPIW
;       MOVLW   0x5B
;       CALL    VDSPIW
;       MOVLW   0x00
;       CALL    VDSPIW
;       MOVLW   0x00
;       CALL    VDSPIW
;       MOVLW   0x0D
;       CALL    VDSPIW

;*************************************************************************************************
VDMKD
        CALL    VDWNM
        MOVLW   0x06
        CALL    VDSPIW
        MOVLW   0x20
        CALL    VDSPIW
        RETURN


;COMANDO DLF ************************************************************************************
; Borra un archivo
; Ejemplo: Borrar el archivo B.TXT
;       CALL    VDDLF
;       MOVLW   'B'
;       CALL    VDSPIW
;       MOVLW   '.'
;       CALL    VDSPIW
;       MOVLW   'T'
;       CALL    VDSPIW
;       MOVLW   'X'
;       CALL    VDSPIW
;       MOVLW   'T'
;       CALL    VDSPIW
;       MOVLW   0x0D
;       CALL    VDSPIW
;*************************************************************************************************
VDDLF
        CALL    VDWNM
        MOVLW   0x07
        CALL    VDSPIW
        MOVLW   0x20
        CALL    VDSPIW
        RETURN


;COMANDO WRF ************************************************************************************
; Escribe un numero de bytes especificados en el archivo que se encuentre abierto
; el numero de bytes debe especificarse en forma binaria con una longitud de 32bits (4bytes)
; Ejemplo: Escribe las letras JP dentro del archivo abierto
;       CALL    VDWRF
;       MOVLW   0x00
;       CALL    VDSPIW
;       MOVLW   0x00
;       CALL    VDSPIW
;       MOVLW   0x00
;       CALL    VDSPIW
;       MOVLW   0x02
```

```
;       CALL    VDSPIW
;       MOVLW   0x0D
;       CALL    VDSPIW
;       MOVLW   'J'
;       CALL    VDSPIW
;       MOVLW   'P'
;       CALL    VDSPIW
;
;*****************************************************************************************************
VDWRF
        CALL    VDWNM
        MOVLW   0x08
        CALL    VDSPIW
        MOVLW   0x20
        CALL    VDSPIW
        RETURN

;COMANDO OPW ********************************************************************************************
; Abre un archivo para escribir o crea un archivo nuevo, se puede también especificar la fecha y
; hora de creación.
; Ejemplo: Abre el archivo C.DAT para escribir en el
;       CALL    VDOPW
;       MOVLW   'C'
;       CALL    VDSPIW
;       MOVLW   '.'
;       CALL    VDSPIW
;       MOVLW   'D'
;       CALL    VDSPIW
;       MOVLW   'A'
;       CALL    VDSPIW
;       MOVLW   'T'
;       CALL    VDSPIW
;       MOVLW   0x0D
;       CALL    VDSPIW
;*****************************************************************************************************
VDOPW
        CALL    VDWNM
        MOVLW   0x09
        CALL    VDSPIW
        MOVLW   0x20
        CALL    VDSPIW
        RETURN


;COMANDO CLF ********************************************************************************************
; Cierra el archivo que se encuentra abierto, debe especificarse el archivo
; Ejemplo: Cierra el archivo C.DAT
;       CALL    VDCLF
;       MOVLW   'C'
;       CALL    VDSPIW
;       MOVLW   '.'
;       CALL    VDSPIW
;       MOVLW   'D'
;       CALL    VDSPIW
;       MOVLW   'A'
;       CALL    VDSPIW
;       MOVLW   'T'
;       CALL    VDSPIW
;       MOVLW   0x0D
;       CALL    VDSPIW
;*****************************************************************************************************
VDCLF
        CALL    VDWNM
        MOVLW   0x0A
        CALL    VDSPIW
```

```
        MOVLW   0x20
        CALL    VDSPIW
        RETURN

;COMANDO RDF ************************************************************************************
; Lee el numero de bytes especificados del archivo que se encuentra abierto
; el numero de bytes debe especificarse en forma binaria con una longitud de 32bits (4bytes)
; Ejemplo: Lee dos bytes del archivo abierto, el primero lo saca por el puerto B y el segundo
; lo saca por el puerto D.
;       CALL    VDRDF
;       MOVLW   0x00
;       CALL    VDSPIW
;       MOVLW   0x00
;       CALL    VDSPIW
;       MOVLW   0x00
;       CALL    VDSPIW
;       MOVLW   0x02
;       CALL    VDSPIW
;       MOVLW   0x0D
;       CALL    VDSPIW
;       CALL    VDSPIR
;       MOVWF   PORTB
;       CALL    VDSPIR
;       MOVWF   PORTD
;************************************************************************************************
VDRDF
        CALL    VDWNM
        MOVLW   0x0B
        CALL    VDSPIW
        MOVLW   0x20
        CALL    VDSPIW
        RETURN


;COMANDO REN ************************************************************************************
;Renombra un archivo o una carpeta
; Ejemplo: Renombra el archivo A.TXT por B.TXT
;       CALL    VDREN
;       MOVLW   'A'
;       CALL    VDSPIW
;       MOVLW   '.'
;       CALL    VDSPIW
;       MOVLW   'T'
;       CALL    VDSPIW
;       MOVLW   'X'
;       CALL    VDSPIW
;       MOVLW   'T'
;       CALL    VDSPIW
;       MOVLW   0x20
;       CALL    VDSPIW
;       MOVLW   'B'
;       CALL    VDSPIW
;       MOVLW   '.'
;       CALL    VDSPIW
;       MOVLW   'T'
;       CALL    VDSPIW
;       MOVLW   'X'
;       CALL    VDSPIW
;       MOVLW   'T'
;       CALL    VDSPIW
;       MOVLW   0x0D
;       CALL    VDSPIW
;************************************************************************************************
VDREN
        CALL    VDWNM
```

```
        MOVLW   0x0C
        CALL    VDSPIW
        MOVLW   0x20
        CALL    VDSPIW
        RETURN


;COMANDO OPR ***************************************************************************************
; Abre un archivo para su lectura, se puede especificar la fecha de acceso en formato 16 bits FAT
; Ejemplo: Se abre el archivo A.TXT y se le coloca como fecha de acceso 0x355B (2008-11-11)
;       CALL    VDOPR
;       MOVLW   'A'
;       CALL    VDSPIW
;       MOVLW   '.'
;       CALL    VDSPIW
;       MOVLW   'T'
;       CALL    VDSPIW
;       MOVLW   'X'
;       CALL    VDSPIW
;       MOVLW   'T'
;       CALL    VDSPIW
;       MOVLW   0x20
;       CALL    VDSPIW
;       MOVLW   0x38
;       CALL    VDSPIW
;       MOVLW   0x5B
;       CALL    VDSPIW
;       MOVLW   0x0D
;       CALL    VDSPIW
;*****************************************************************************************************
VDOPR
        CALL    VDWNM
        MOVLW   0x0E
        CALL    VDSPIW
        MOVLW   0x20
        CALL    VDSPIW
        RETURN

;COMANDO SEK ***************************************************************************************
; Mueve el puntero a la posición especificada en el archivo que se encuentra abierto
; la posición del puntero debe especificarse en forma binaria con una longitud de 32bits (4bytes)
; Ejemplo: Mueve el puntero a la posición 0xFF (255) del archivo abierto
;       CALL    VDSEK
;       MOVLW   0x00
;       CALL    VDSPIW
;       MOVLW   0x00
;       CALL    VDSPIW
;       MOVLW   0x00
;       CALL    VDSPIW
;       MOVLW   0xFF
;       CALL    VDSPIW
;       MOVLW   0x0D
;       CALL    VDSPIW
;*****************************************************************************************************
VDSEK
        CALL    VDWNM
        MOVLW   0x28
        CALL    VDSPIW
        MOVLW   0x20
        CALL    VDSPIW
        RETURN


;COMANDO FS ****************************************************************************************
; Muestra el espacio libre en bytes con longitud de 32bits (4bytes) del disco si este es menor a 4GB
```

```
; Ejemplo:
;       CALL    VDFS
;       Devuelve:tttt
; Donde tttt es un dato de 32bits que indica el espacio libre en bytes
VDFS
        CALL    VDWNM
        MOVLW   0x12
        CALL    VDSPIW
        MOVLW   0x0D
        CALL    VDSPIW
        RETURN

;COMANDO FSE ********************************************************************************
; Muestra el espacio libre en bytes con longitud de 48bits (6bytes) en el disco
; Ejemplo:
;       CALL    VDFS
;       Devuelve:tttttt
; Donde tttttt es un dato de 48bits que indica el espacio libre en bytes
VDFSE
        CALL    VDWNM
        MOVLW   0x93
        CALL    VDSPIW
        MOVLW   0x0D
        CALL    VDSPIW
        RETURN


;COMANDO SUD ********************************************************************************
; Coloca a la memoria USB en modo de bajo consumo
VDSUD
        CALL    VDWNM
        MOVLW   0x15
        CALL    VDSPIW
        MOVLW   0x0D
        CALL    VDSPIW
        RETURN

;COMANDO WKD ********************************************************************************
; Saca a la memoria del modo de bajo consumo
VDWKD
        CALL    VDWNM
        MOVLW   0x16
        CALL    VDSPIW
        MOVLW   0x0D
        CALL    VDSPIW
        RETURN

;FUNCION LEER RESPUESTA DE COMANDO **********************************************************
; Esta función lee la respuesta dada por un comando, coloca al Carry en 0 si el comando fue
; aceptado y en 1 si no lo fue
VDRRC
        CALL    VDSPIRN                 ; Realiza la lectura de dato nuevo del VDRIVE1
        SUBLW   0x3E                    ; Compara el primer dato con 0x3E
        BSF     STATUS,C                ; Supone que no es igual y coloca el Carry en 1
        BTFSS   STATUS,Z                ; Testea si el dato es igual
        RETURN                          ; Si no lo es retorna de la función

        CALL    VDSPIRN                 ; Si es igual lee el siguiente dato
        SUBLW   0x0D                    ; Compara el segundo dato con 0x0D
        BSF     STATUS,C                ; Supone que no es igual y coloca el Carry en 1
        BTFSS   STATUS,Z                ; Testea si el dato es igual
        RETURN                          ; Si no lo es retorna de la función
        BCF     STATUS,C                ; Si es igual coloca al Carry en 1
        RETURN                          ; y retorna de la función
```

```
;FUNCION LEER COMANDO FALLIDO ***************************************************************************
; Esta función lee la salida de datos del VDRIVE1 y la compara con la secuencia de respuesta de
; comando fallido, es muy útil para conocer cuando un archivo ha llegado a su fin. Coloca al Carry
; en 0 si la respuesta es comando fallido y en 1 si no lo es.
; Ejemplo: Se leerá los datos de un pequeño archivo abierto que contiene el texto "1234567890" de 3
; en 3 caracteres.
;
; LECTURA
;       CALL    VDRDF           ; Llama a la función leer datos de archivo especificando
;       MOVLW   0x00            ; que leerá 3 bytes.
;       CALL    VDSPIW
;       MOVLW   0x00
;       CALL    VDSPIW
;       MOVLW   0x00
;       CALL    VDSPIW
;       MOVLW   0x03
;       CALL    VDSPIW
;       MOVLW   0x0D
;       CALL    VDSPIW
;
;       MOVLW   0x03
;       MOVWF   CONTCAR         ; Inicializa al contador de caracteres CONTCAR en 3
;
; LECTURA1
;       CALL    VDSPIRN         ; Llama a la función leer dato nuevo
;       CALL    MLETRA          ; Manda el dato a la matriz de leds
;       DECFSZ  CONTCAR,F       ; Decrementa a CONTCAR para saber cuantos caracteres
;       GOTO    LECTURA1        ; se han leído, salta a LECTURA1 para continuar.
;       CALL    VDRCF           ; Si acabo de leer los 3 caracteres detecta si el
;       BTFSC   STATUS,C        ; archivo llego a su fin.
;       GOTO    LECTURA         ; Si no ha llegado a su fin salta a LECTURA
;                               ; para continuar con los siguientes 3 caracteres
;       GOTO    FINARCHIVO      ; Si termino de leer el archivo salta a FINARCHIVO
;
;*************************************************************************************************************
VDRCF
        CALL    VDSPIRN
        SUBLW   0x43
        BSF     STATUS,C
        BTFSS   STATUS,Z
        RETURN
        CALL    VDSPIRN
        SUBLW   0x46
        BSF     STATUS,C
        BTFSS   STATUS,Z
        RETURN
        CALL    VDSPIRN
        SUBLW   0x0D
        BSF     STATUS,C
        BTFSS   STATUS,Z
        RETURN
        BCF     STATUS,C
        RETURN


;FUNCION DETECTAR PRESENCIA DE MEMORIA USB ***********************************************************
; Esta función detecta si existe una memoria USB conectada o no al VDRIVE1, devuelve al Carry en 1 si
; la memoria no fue detectada y en 0 si lo fue.
VDUSBP
        CALL    VDWWNM          ; Espera varias veces a que el modulo este listo
        MOVLW   0x0D            ; Manda un retorno de carril para descartar otros mensajes
        CALL    VDSPIW
        CALL    VDWWNM          ; Espera varias vecs a que el modulo este listo
```

```
        MOVLW   0x0D            ; Manda un retorno de carril para que el modulo responda con
        CALL    VDSPIW          ; un 0x3E,0x0D en el caso de existir una memoria conectada

        CALL    VDSPIRN         ; Lee una nueva entrada
        SUBLW   0x3E            ; La compara con 0x3E
        BSF     STATUS,C        ; Supone que no es igual y coloca al Carry en 1
        BTFSS   STATUS,Z
        RETURN                  ; Si no es igual termina la función
        CALL    VDSPIRN         ; Como es igual lee una entrada nueva
        SUBLW   0x0D            ; Compara el segundo dato con 0x0D
        BSF     STATUS,C        ; Supone que no es igual y coloca al Carry en 1
        BTFSS   STATUS,Z
        RETURN                  ; Si no es igual termina la función
        BCF     STATUS,C        ; Si es igual coloca al Carry en 0 y
        RETURN                  ; termina la función


;FUNCION DE RETARDO SPI ********************************************************************************
; Genera un retardo de 16 ciclos necesario para algunas instrucciones de esta librería
VDRET
        MOVLW   0x0F            ; Inicializa el contador VDRETCONT en 0x0F
        MOVWF   VDRETCONT
VDRET1
        NOP                     ; Realiza un nop, No Operation
        DECFSZ  VDRETCONT,F     ; Decrementa el contador
        GOTO    VDRET1          ; Si no ha llegado a 0 salta a VRET1
        RETURN                  ; Si llegó a cero termina la función
```

5 APLICACIONES

Los alcances de este proyecto se ven reflejados en dos nuevas aplicaciones que trabajan con la lectura y la escritura de archivos dentro de un dispositivo de almacenamiento USB; una matriz de LEDS que muestra los carácteres contenidos dentro de un archivo denominado TEXTO.TXT y un electrocardiógrafo que graba la señal del corazón en formato digital dentro de un fichero. Estas aplicaciones se realizan con el fin de demostrar la funcionalidad de la librería para el manejo del VDRIVE1 implementada, también están estructuradas para que sirvan como ejemplos prácticos para aquellas personas que deseen realizar nuevas soluciones.

5.1. MATRIZ DE LEDS USB

Uno de los ejemplos prácticos de la implementación de la comunicación entre microcontroladores serie PIC y memorias USB es una matriz de LEDS que presenta los caracteres guardados en un archivo de texto. El proyecto a desarrollado un sistema en el cual un microcontrolador PIC16F877A programado en assembler y conectado al modulo VDRIVE1 es capaz de leer un archivo llamado TEXTO.TXT y mostrar su contenido en una matriz de LEDS que funciona con registros de desplazamiento, la velocidad de desplazamiento del texto es configurada por un potenciometro que esta conectado a una entrada análoga del microcontrolador.

5.1.1. Matriz de LEDS. La matriz de LEDS esta conformada por 26 columnas y 8 filas de LEDS, donde los datos se van desplazando de fila en fila de derecha a izquierda para formar caracteres o figuras. Funciona con 26 registros de desplazamiento 74LS374, donde cada uno se encarga de una columna. El dato llega a la entrada del registro y este lo desplaza a la salida al recibir un pulso ascendente por la entrada de clock. Para entender mejor su funcionamiento el esquema del ANEXO C muestra la disposición de los elementos circuitales.

5.1.2. Librería MATRIZL.INC. Esta librería realizada en assembler contiene las rutinas para controlar la matriz de LEDS. Posee dos funciones principales, una de inicialización y otra para imprimir caracteres. El código de la librería se encuentra en el ANEXO D.

Función de inicialización MDLINIC. Esta función configura los pines utilizados para el manejo de la matriz de LEDS, también configura el conversor análogo digital para entrar los datos del potenciómetro que controla la velocidad de desplazamiento del texto.

Función enviar letra MLETRA. Esta función toma el dato que esta en el registro de trabajo W y lo manda como carácter a la matriz de LEDS, si no es un carácter imprimible manda un espacio, se basa en la codificación de caracteres latin-1 o ISO 8859-1.

Ejemplo: Manda las palabra UDENAR a la matriz de LEDS.

```
MOVLW   'U'
CALL    MLETRA
MOVLW   'D'
CALL    MLETRA
MOVLW   'E'
CALL    MLETRA
MOVLW   'N'
CALL    MLETRA
MOVLW   'A'
CALL    MLETRA
MOVLW   'R'
CALL    MLETRA
```

Para lograr mandar un carácter a la matriz de LEDS esta función toma el dato de entrada y compara si es un comando, un carácter imprimible ASCII, un carácter especial o no es imprimible. Si es un comando o un carácter no imprimible manda un espacio en blanco a la matriz de LEDS, si es un carácter imprimible ASCII lo manada a una tabla que salta a la posición donde se le colocan los valores correspondientes a los registros de salida para conformar la figura del carácter introducido, si es un carácter especial es comparado con los caracteres especiales del idioma español, como vocales con tildes, la eñe o la diéresis, si es una de esta salta a la posición donde están los valores correspondientes y si no es uno de estos manda un espacio en blanco. El diagrama de flujo de la Figura 5.1 explica de forma mas detallada el proceso.

5.1.3. Programa principal. El funcionamiento del programa principal se rige por el diagrama de flujo mostrado en la Figura 5.2, su código se encuentra en el ANEXO A.

Este algoritmo esta diseñado para leer forma continua y repetitiva el archivo TEXTO.TXT codificado en latin1 o ISO 8859-1. Si se deja precionado el botón 1 durante mas de 3 caracteres la lectura se interrumpe, en este momento se puede iniciar la lectura desde el principio del archivo TEXTO.TXT pulsando nuevamente el botón 1, se puede extraer de forma segura la memoria y luego apagar la unidad o se puede cambiar la memoria por otra que tenga un archivo llamado TEXTO.TXT, que posea un mensaje que se quiera mostrar, se espera unos 3 segundos para que el modulo reconozca la memoria y se pulsa el botón 1 para iniciar la lectura.

5.2. ELECTROCARDIOGRAFO

Se ha implementado un electrocardiógrafo que es capaz de tomar la señal del corazón y guardarla dentro de un archivo en una memoria USB. El diagrama de bloques de la Figura 5.3 muestra sus componentes y funcionamiento.

5.2.1. Electrocardiograma ECG. "Todo organo que trabaja es productor de electricidad. El corazón, cuando se contrae, es, pues, sede de variaciones de potencial eléctrico." [1] El potencial eléctrico de corazón puede ser medido colocando un electrodo en en la punta y uno en la base, pero gracias a que los tejidos orgánicos son buenos conductores se puede colocar estos electrodos en la superficie del cuerpo, este método de registro se denomina método indirecto.

Las derivaciones estándar de Einthoven. Einthoven pensó que, siendo el corazón un generador de corriente y el cuerpo humano un buen conductor, podría construirse imaginariamente un triángulo, formado por las raíces de los miembros, sobre cuyos lados se proyectarían las fuerzas eléctricas emanadas del músculo cardíaco. Dado que el corazón se inclina dentro del pecho hacia la izquierda, y como los brazos y piernas son prolongaciones de sus respectivas raíces, en la práctica empleamos los miembros superiores y el inferior izquierdo para construir el triángulo. En este proyecto se trabaja con la derivación del brazo izquierdo como la parte positiva de la señal, la del brazo

[1] CARATINI, Roger, Argos Enciclopedia temática, Medicina I, Barcelona: Argos, 1970. p.63

derecho como la negativa y la de la pierna izquierda como tierra tal como se ve en la Figura 5.4.

Onda PQRST. El reflejo del potencial eléctrico del corazón sobre el eje del tiempo conforma una onda denominada PQRST, ver Figura 5.5, esta onda describe distintos instantes del pulso cardiaco:

- La onda P indica que las aurículas (las dos cavidades superiores del corazón) son estimuladas en forma eléctrica para bombear la sangre hacia los ventrículos.
- El complejo QRS indica que los ventrículos (las dos cavidades inferiores del corazón) se están estimulado eléctricamente para bombear la sangre hacia fuera.
- El segmento ST indica la cantidad de tiempo que transcurre desde el final de una contracción de los ventrículos hasta el comienzo del período de reposo
- La onda T indica el período de recuperación de los ventrículos

Frecuencia de muestreo señal ECG. Según GUTIERREZ, Rafael M. y CERQUERA, Edwin A. los datos tomados que contienen información de un comportamiento dinámico de una señal cardiaca no varian en frecuencias de muestreo de 300Hz o 600Hz en cambio para un muestreo 100Hz se observan cambios cuantitavos mas no cualitativos que empiezan a afectar el sistema para cierto tipo de análisis; por esta razón en esta aplicación se usa una frecuencia de muestreo igual a 400Hz con la cual no se perderá ninguna detalle importante de la señal.

5.2.2. Acondicionamiento de la señal. El acondicionamiento de la señal eléctrica obtenida por los electrodos sobre la piel se realiza con amplificadores operacionales de instrumentación TL084CN los cuales tienen como característica principal una gran impedancia de entrada, del orden de los $10^{12}\Omega$. Basándose en el circuito amplificador de señales cardiacas diseñado por Jhon Nozum's de West Virgina University se ha implementado el esquema circuital del ANEXO E.

Despues de esta etapa se implementa un filtro anti alias de orden 8 tipo Butterworth con una frecuencia de corte de 200Hz para el muestreo de 400Hz, el esquema del circuito se puede ver en el ANEXO F.

A continuación viene una etapa en la que se le da la ganancia final a la señal y un nivel de offset, el esquema del circuito se encuentra en el ANEXO G.

5.2.3. Conversión Análoga a Digital A/D. Los microcontroladores PIC16F87X cuentan con un modulo para conversión Análoga a Digital que tiene una resolución de 10 bits. La conversión se realiza mediante aproximaciones sucesivas, el diagrama de este arreglo se encuentra en la Figura 5.6. Supongamos que la entrada análoga es de 1 voltio y el voltaje de referencia es de 5 voltios en un conversor de 4 bits de resolución, es decir que si la entrada es de 0V la salida digital sera de 0000 en binario y si la entrada análoga es de 5 voltios la salida será de 1111 en binario.

La entrada análoga llega a la parte positiva de un comparador donde la parte negativa inicia en la mitad del voltaje de referencia (Vref) en este caso 2.5V, la salida del comparador nos dice si el dato es mas grande que el 2.5V o menor, esto representa el MSB del dato que deseamos obtener, en este caso es 0, esta dato llega al registro de aproximaciones susecivas (SAR) el cual ahora sabe que el dato se encuentra entre 0 y 2.5V por lo cual manda el valor en binario de 0100 al DAC para que saque 1.25V este valor entra en la parte negativa del comparador y es comparado con la entrada de 1V, como es menor sale un 0 que entra al SAR el cual ya sabe que la entrada se encuentra entre 0 y 1.25V y coloca a ese cero en el bit 2, este manda el dato 0010 que equivale a 0.625 voltios, los cuales van al comparador donde 1V es mayor, este manda un 1 al SAR el cual ahora sabe que el dato se encuentra entre 0.625 y 1.25V y coloca ese 1 en el bit 1, el SAR manda ahora el valor 0011 al ADC el cual lo convierte en 0.9375V los cuales son comparados con el 1V de entrada, el comparador saca un 1 que entra al SAR el cual ahora sabe que el dato se encuentra entre 0.9375V y 1.25V, coloca ese 1 en el bit 0 de dato de salida obteniendo 0011 como respuesta, manda una señal informando que la conversión fue terminada para que un registro de desplazamiento (SRG) publique el dato. El diagrama de bloques de la Figura 5.7 resume este funcionamiento.

5.2.4. Programa principal. El programa del microcontrolador PIC16F877A realizado en assembler graba de forma digital una señal análoga de ECG entre 0 a 5v con un muestreo de 400Hz a una resolución de 10bits en un archivo ECG_XX.DAT donde XX es el numero del archivo que se puede seleccionar mediante dos displays de 7 segmentos y dos bonotes de incrementar y decrementar.

Al iniciar al programa, este le pide al usuario que escoja en que archivo desea guardar los datos, el numero del archivo es indicado por dos display de 7 segmentos, este numero se puede cambiar con los botones de incrementar y decrementar. Un LED denominado archivo libre indica que no existe un archivo con ese numero y que el lugar esta libre para escribir uno nuevo. Un LED denominado archivo ocupado indica que si existe un archivo con ese nombre y que no se puede escribir en el. Uno de estos LEDS se enciende cada vez que el usuario se posiciona en un archivo.

Después de escoger el archivo se pulsa el botón grabar/finalizar para entrar en una etapa de calibración de la señal. Se tienen 4 LEDS, dos rojos exteriores de limites alto y bajo y dos verdes interiores de umbral alto y bajo. Para calibrar la entrada hay que ajustar los potenciómetros de offset y el de ganancia de tal forma que parpadeen solo los LEDS verdes de umbral, si parpadean los LEDS rojos de limite alto o bajo quiere decir que se esta acercando mucho a los limites de lectura del digitalizador.

Una vez realizada la calibración se pulsa el botón grabar/finalizar para iniciar con la grabación, en este momento empiezan a guardar los datos en la memoria USB en el archivo seleccionado, se puede seguir observando los LEDS de calibración para ver como se comparta la entrada. Para finalizar la grabación se pulsa el botón grabar/finalizar, los display de 7 segmentos muestran un doble cero indicando que el proceso a terminado. En este momento se puede extraer la memoria de forma segura,se puede cambiarla por otra y/o se puede pulsar el botón grabar/finalizar para grabar en un diferente archivo. El diagrama de flujo de la Figura 5.8 resume el funcionamiento de este programa y su código se encuentra en el ANEXO H.

Figura 5.1: Diagrama de flujo función MLETRA

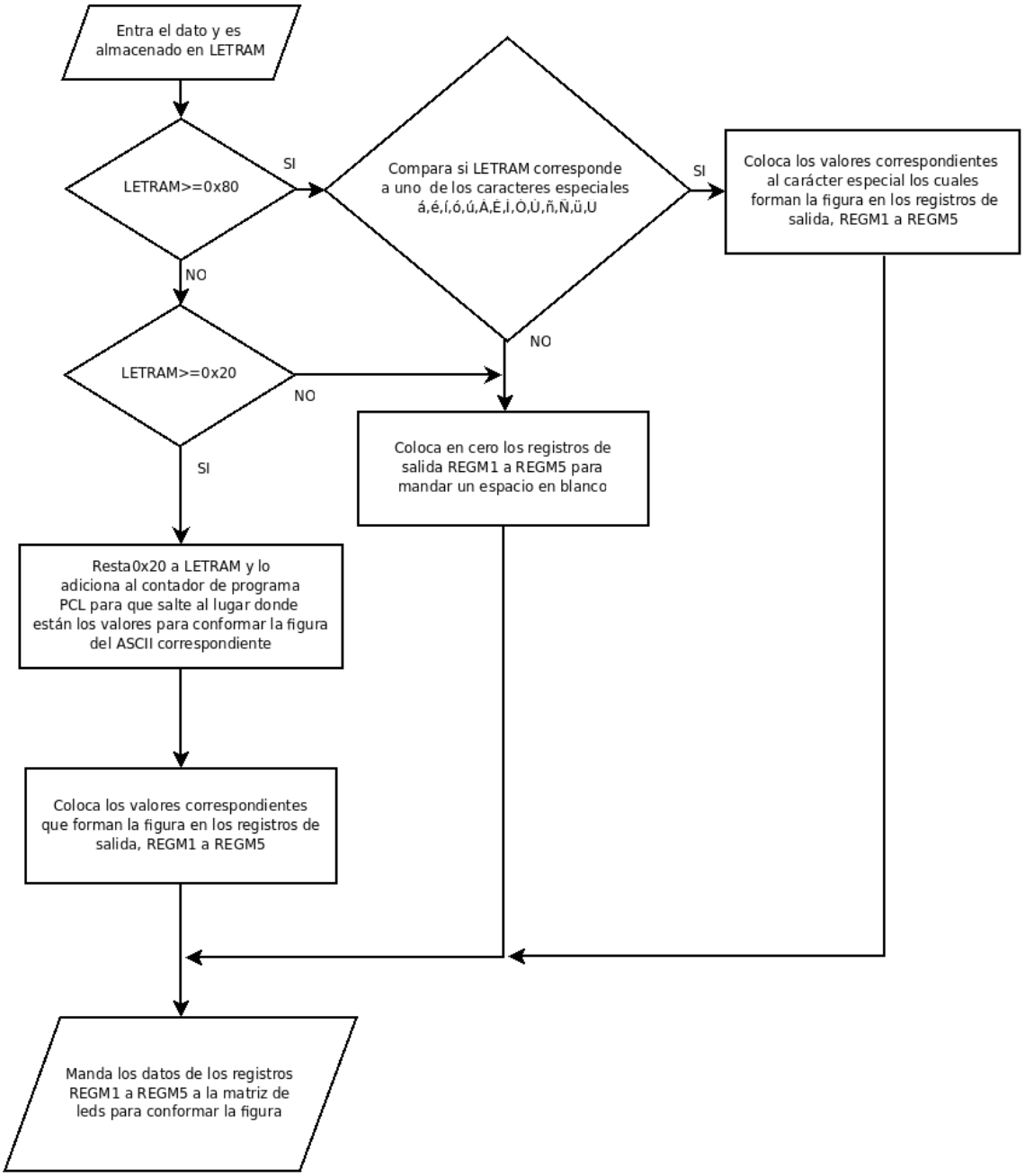


Figura 5.2: Diagrama de flujo Matriz de LEDS USB

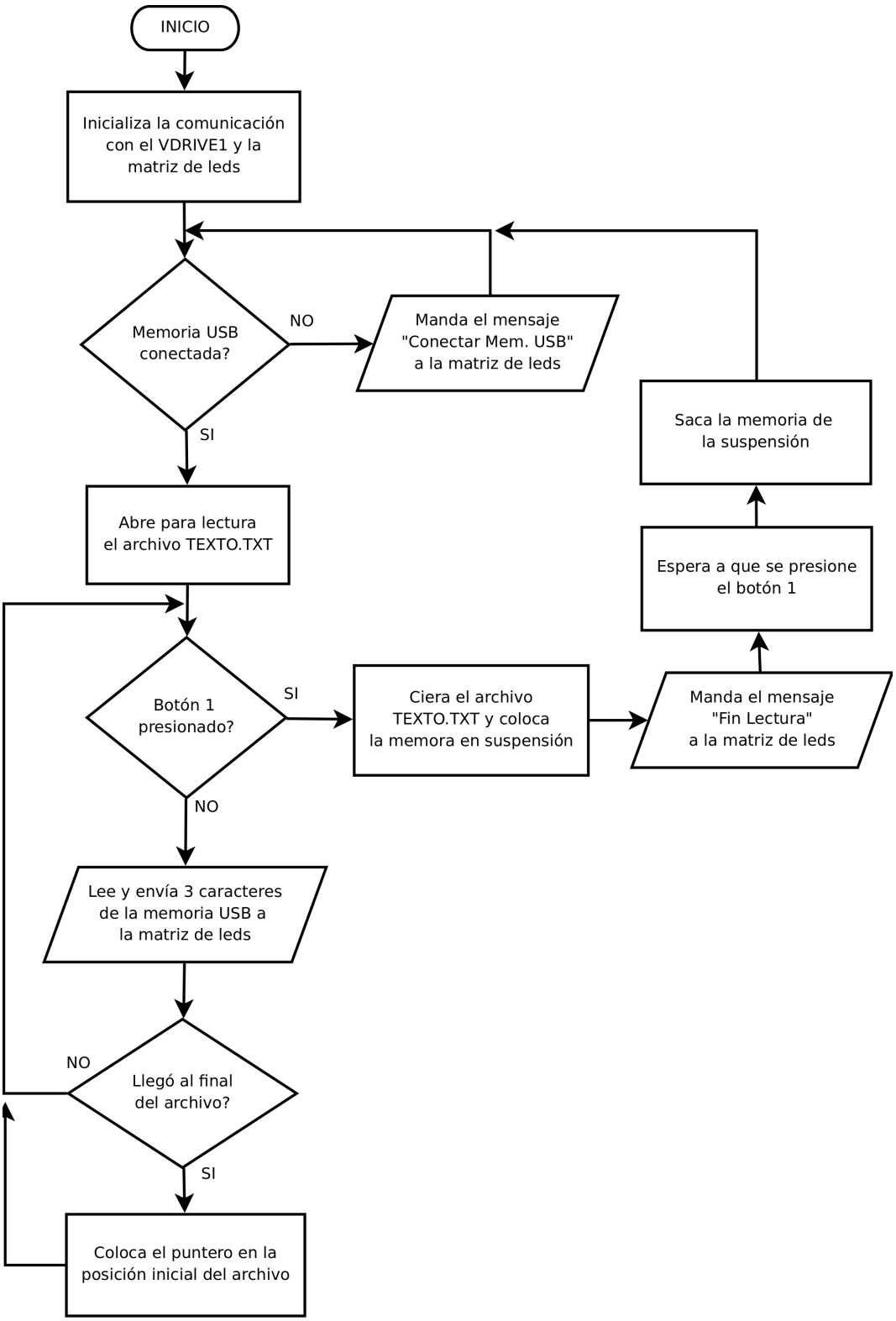


Figura 5.3: Electrocariógrafo

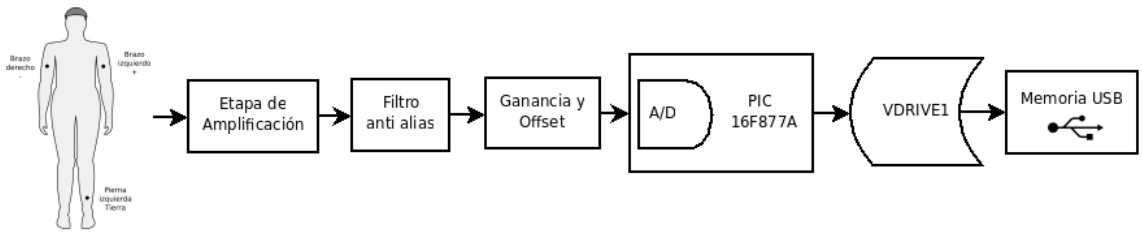


Figura 5.4: Derivaciones bipolares

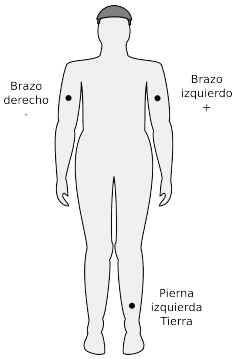
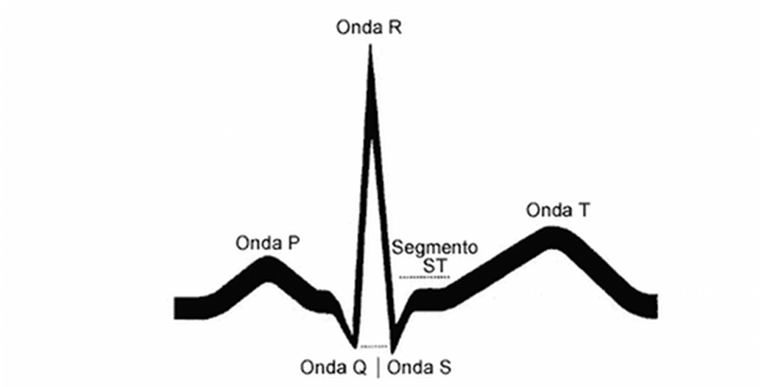


Figura 5.5: Onda PQRST



University of Virginia, disponible en internet:
http://www.healthsystem.virginia.edu/uvahealth/peds_cardiac_sp/ekgecg.cfm, 2008.

Figura 5.6: Conversor ADC SAR

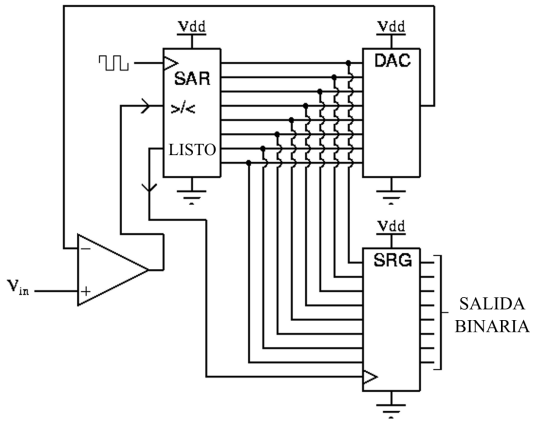


Figura 5.7: Diagrama de flujo ADC SAR

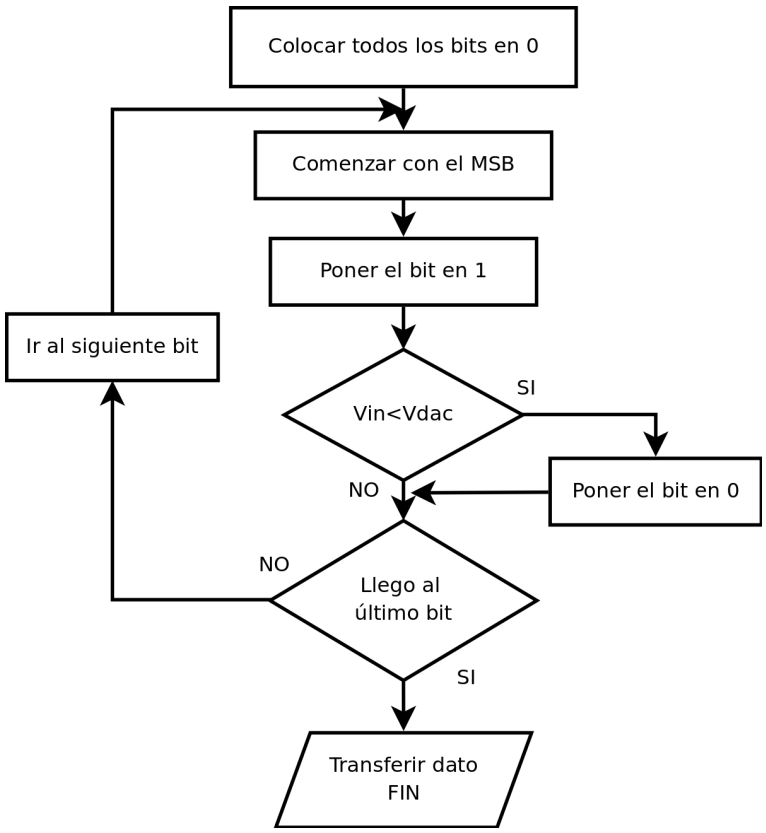
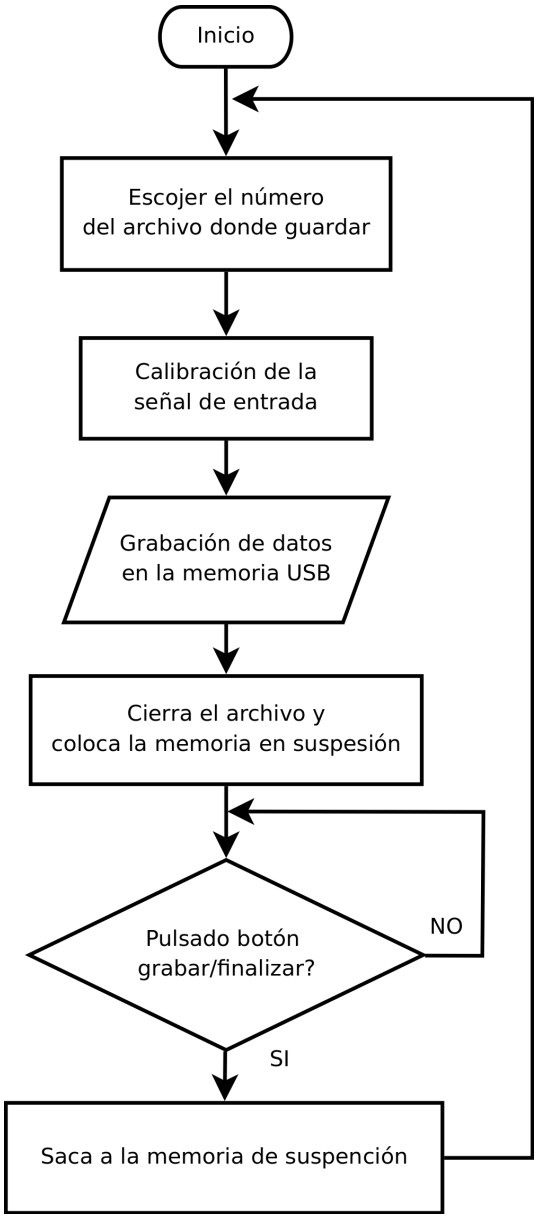


Figura 5.8: Diagrama de flujo programa ECG



6 CONCLUSIONES

Es muy importante realizar una investigación profunda de las nuevas tecnologías existentes que permiten llegar a una solución, ya que con ellas se pueden implementar sistemas mucho mas eficientes y así aprovechar al máximo los recurosos con los que se cuenta. En este caso el uso del modulo VDRIVE1, de la empresa inglesa Vinculum, logró que este proyecto fuera mas haya de lo que se había planteado desarrollando dos muy buenas soluciones que hacen un excelente uso de la capacidad con la que cuenta una memoria USB para almacenar datos.

El uso de la emulación de un puerto serial RS-232 mediante el puerto USB con la ayuda del PIC18F2550 facilitó en gran medida el desarrollo de la terminal USB a SPI, puesto que el programa dentro del microcontrolador no tuvo que ser muy complejo ya que la librería de comunicación la provee Microchip. No hubo también la necesidad de desarrollar un software para que el computador envíe comandos ya que existen muchos programas para el manejo de una terminal por serial como HiperTerminal de Microsoft o el software libre Putty de Simon Tatham el cual fue usado en este proyecto.

La terminal USB a SPI desarrollada en este proyecto fue de gran ayuda para probar los comandos y su respuesta en el modulo VDRIVE1, además sirvió como debugger pues en el desarrollo de las soluciones mostradas dentro de este proyecto cuando los comandos enviados no respondían de manera adecuada, se enviaban paso a paso mediante la terminal para conocer exactamente lo que sucedía y así poder hallar e implementar de forma rápida una solución

Una matriz de LEDS de 8 filas por 26 columnas es un muy buen elemento para mostrar mensajes y una memoria USB un gran medio para almacenarlos lo que conforman un sistema idoneo para mostrar de una manera sencilla y practica un texto o diferentes textos donde se quieran expresar nuevas ideas.

El uso de un sistema de archivos por parte de una memoria es una gran ventaja para el desarrollo del electrocardiógrafo pues no existe una mejor manera de organizar distintas señales que en archivos lo que da un orden a la información y una gran facilidad al usuario para encontrarla.

El modulo VDRIVE1 cuenta con un firmware muy bien soportado capaz de reconocer una gran cantidad de memorias USB antiguas o modernas. En la etapa de investigación de este modulo se realizaron pruebas con una memoria Kingston DatraTravel de 2GB la cual fue reconocida sin problemas, luego se probo con una memoria mas nueva, de la misma marca y modelo con capacidad de un 1GB. Esta memoria nueva no fue reconocida por el modulo, por lo cual se descargó una actualización del firmware por internet, se la coloco en la memoria de 2GB y luego se la inserto en el VDRIVE1, este actualizo su software interno y despues reconoció sin problemas la memoria nueva de 1GB.

7 RECOMENDACIONES

Para lograr una mayor frecuencia de muestreo se recomienda programar un buffer en el microcontrolador para mandar los datos al VDRIVE1, puesto que este tiene unos retardos en la escritura ya que busca cluster vacíos en la memoria donde pueda guardar los datos.

Implementar las normas medicas pertinentes en el electrocardiógrafo para un uso profesional del mismo.

BIBLIOGRAFIA

CARATINI, Roger. Argos Enciclopedia temática, Barcelona, Argos, 1970, 189p.

COMPAQ, HEWLETT-PACKAR, INTEL, LECENT, MICROSOFT, NEC, PHILIPS, Universal Serial Bus Specification, 2000, 622 p.

Emulating RS-232 over USB with PIC18F4550, Microchip Technology Inc., WebSeminar, 2004 , Disponible en Internet: URL:
http://techtrain.microchip.com/webseminars/documents/
EmulatingRS-232overUSB_121004.pdf, 28p.

GUTIERREZ, Rafael M. y CERQUERA, Edwin A., Medidas dinamicas: detección de nueva información contenida en el ECG, Revista Colombiana de Física, Vol. 36, No. 2, Bogotá, Universidad Antonio Nariño, 2004,

HEYBRUCK, William F. An Introduction to FAT 16/FAT 32 File Systems, [monografía en linea], 1 ed [Charlotte, NC]: Hitachi Global Storage Technologies, 2005, Disponible en internet: URL:
http://www.hitachigst.com/tech/techlib.nsf/techdocs/BB4945CEAAE-
4DAD986256D890016E8F4/$ file/FAT_ White_ Paper_ FINAL.pdf, 4p.

MARTINEZ DURA, Rafael J., BOLUDA, Jose A., PEREZ, Juan J., Estructura de Computadores y Periféricos, Valencia, RA-MA, 2001, 404p.

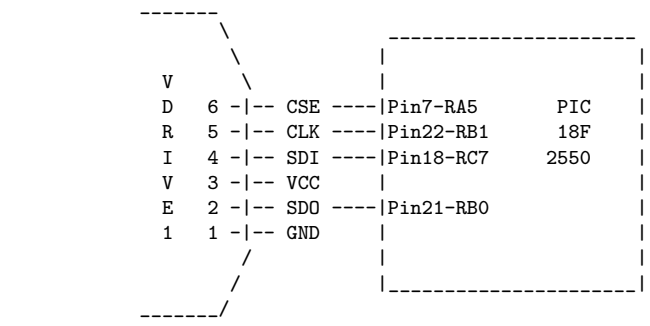
Migrating Applications to USB from RS-232 UART with Minimal Impact on PC Software, Microchip Technology Inc, 2004, Disponible en Internet: URL:
http://ww1.microchip.com/downloads/en/AppNotes/00956b.pdf, 16p.

MPLAB C18 C COMPILER GETTING STARTED, Microchip Technology Inc, 2005, Disponible en Internet: URL:
http://ww1.microchip.com/downloads/en/DeviceDoc/
MPLAB_C18_Getting_Started_51295f.pdf, 128p.

MPLAB C18 C COMPILER USER

ANEXO A. Código principal terminal USB

```
/********************************** TERMINAL USB A SPI ************************************
Este programa permite enviar comandos SPI al VDRIVE1 mediante el puerto USB en una emulación
del puerto serial. Se usa el PIC 18F2550 en modo CDC (Communication Device Class), el driver
se puede descargar desde: http://www.microchip.com buscando "USB Framework for PIC18"

Esta configurado para trabajar a 152000 bps, se conecta al VDRIVE1 como se muestra a
continuación:
```




```
Posee dos modos de funcionamiento, el modo comandos cortos en el cual solo se pueden
introducir valores hexadecimales en ASCII en mayúsculas, los cuales serán representados
en la pantalla con su carácter ASCII si es imprimible, los datos entregados por el VDRIVE1
se mostraran de la misma forma y el modo de comandos extendidos que trabaja con caracteres
ASCII imprimibles, con instrucciones como OPW u OPR, el VDRIVE1 responderá de la misma
forma los datos.


    ================================================================================
    ||                              PROYECTO DE GRADO                             ||
    ||                                                                            ||
    ||     ESTUDIO Y DISEÑO DE ALGORITMOS E INTERFAZ ELECTRÓNICA DE COMUNICACIÓN  ||
    ||    ENTRE MICROCONTROLADORES SERIE PIC Y DISPOSITIVOS DE ALMACENAMIENTO USB ||
    ||                                                                            ||
    ||                          Juan Pablo Ruiz Rosero                            ||
    ||                          Ingeniería Electrónica                            ||
    ||                           Universidad de Nariño                            ||
    ||                               Octubre 2008                                 ||
    ================================================================================


*********************************************************************************************/


/** LIBRERIAS*********************************************************************************/

#include <p18f2550.h>         // Librería de direcciones del microcontrolador
#include "system\typedefs.h"  // Definiciones
#include "system\usb\usb.h"      // Manejo del usb
#include "io_cfg.h"     // Configuración entrada y salida
#include "spi.h"     // Comunicación SPI

/** CONFIGURACIÓN DEL MICROCONTROLADOR ********************************************************/

        #pragma config PLLDIV   = 3          // Se usa un cristal de 12 MHz y se divide entre 3
        #pragma config CPUDIV   = OSC1_PLL2     // para el PLL
        #pragma config USBDIV   = 2          // Reloj del USB de 96MHz PLL/2
        #pragma config FOSC     = HSPLL_HS
        #pragma config FCMEN    = OFF
```

```
        #pragma config IESO     = OFF
        #pragma config PWRT     = OFF
        #pragma config BOR      = ON
        #pragma config BORV     = 3
        #pragma config VREGEN   = ON         // USB Voltage Regulator encendido
        #pragma config WDT      = OFF
        #pragma config WDTPS    = 32768
        #pragma config MCLRE    = ON
        #pragma config LPT1OSC  = OFF
        #pragma config PBADEN   = OFF
        #pragma config STVREN   = ON
        #pragma config LVP      = OFF
        #pragma config XINST    = OFF        // Extended Instruction Set deshabilitadas
        #pragma config CP0      = OFF
        #pragma config CP1      = OFF
        #pragma config CPB      = OFF
        #pragma config WRT0     = OFF
        #pragma config WRT1     = OFF
        #pragma config WRTB     = ON
        #pragma config WRTC     = OFF
        #pragma config EBTR0    = OFF
        #pragma config EBTR1    = OFF
        #pragma config EBTRB    = OFF


/** DELCARACION DE VARIABLES GLOBALES ******************************************************/
char input_buffer[32];
char output_buffer[8];

/** DELCARACION DE SUBRUTINAS **************************************************************/
static void InitializeSystem(void);
void delay (void);
void UserInit(void);
void InitializeUSART(void);
void USBTasks(void);
void putsUSART(char *data);
void str2ram(static char *dest, static char rom *src);
void putsUSART_rom( static const rom char *data );
void UART_char(char dato);
void UART_cyb(char dato);
char ha2bin(char carbyte[2]);
void ejeccom();

/** CODIGO PRINCIPAL ***********************************************************************/
#pragma code
void main (void)
{

/** VARIABLES ******************************************************/

char entusb;
char leido;
char datoru;
char carbyte[2];
char ncarbyte;
char datoinuart;
char i;
unsigned int im;

UART_char(130);
UART_char(55);
UART_char(241);
SPI_init();       // Función para inicializar la comunicación SPI
InitializeSystem();     // Función para inicializar la comunicación USB
```

```
TRISAbits.TRISA3=0;     // Coloca como salida al pin que maneja el LED
                  // que muestra la actividad de la comunicación SPI.
PORTAbits.RA3=0;     // Inicia el LED de comunicación SPI apagado

for(i=0; i<100; i++)    // Ciclo para establecer la comunicación USB-CDC
{
   while(1)
   {
      USBTasks();    // Servicio de comunicación USB
               // debe ser llamado constantemente para el funcionamiento correcto
               // de la comunicación USB
      if(mUSBUSARTIsTxTrfReady())
      {
         break; // Si la comunicación está lista sale del while
      }
   } // end while, se confirma 100 veces si la comunicación está lista
} // end for


while(1)
{
// En este ciclo se espera que se presione cualquier tecla en la terminal
// de comunicación, luego se presenta un mensaje de bienvenida

   USBTasks();    // Servicio de comunicación USB
   if(mUSBUSARTIsTxTrfReady())      // Se pregunta si el USBUSART esta listo
    {
      entusb=getsUSBUSART(input_buffer,32);     // Se reciben datos en inputbuffer
   }

   if(entusb>0)   // Si entusb es mayor de 0 hay un dato nuevo

   {
      while(1)    // Ciclo para esperar que el USB este listo para la comunicación
      {
         USBTasks();
         if(mUSBUSARTIsTxTrfReady())
         {
            putrsUSBUSART("TERMINAL USB A SPI - UNIVERSIDAD DE NARÑO\r\n\r\n
            1.Modo SCS\r\n2.Modo ESC\r\nEscoja una opción\r\n");
            entusb=getsUSBUSART(input_buffer,32);
            USBTasks();
            break;
            // Se envía el mensaje de bienvenida seguido de recepción de datos y servicios USB para
            // no tener retornos no deseados, luego con el break se termina el while de comunicación.
         }
      }
   break; // Se termina el while que espera la pulsación de cualquier tecla
   }


}

while(1)
// Ciclo que espera la pulsación de una de las opciones del modo
// de la terminal.
{
   USBTasks();
   if(mUSBUSARTIsTxTrfReady())
   {
   // Cuando esta listo recibe el dato
      entusb=getsUSBUSART(input_buffer,32);
   }
   if(entusb>0)
   // Si hay un dato nuevo en la entrada entusb es mayor de 0
```

```
   {
   // Se realiza un salto a la opción escogida
   // Los if y el while solo permiten que sean válidas únicamente las teclas 1 y 2
      if(input_buffer[0]=='1')
         goto modoscs;
      if(input_buffer[0]=='2')
         goto modoecs;

   } // end iff

} // end while



//*******************************MODO SCS*************************************************************
modoscs:

// Se manda la instrucción SCS al VDRIVE1 para informarle
// que se comunique en modo comandos cortos

SPI_write(0x0D); // Se coloca un 0x0D para que entradas anteriores sean descartadas
SPI_write('S');
SPI_write('C');
SPI_write('S');
SPI_write(0x0D);

while(1)
// Ciclo que muestra el mensaje de bienvenida al modo SCS
{
   USBTasks();
   if(mUSBUSARTIsTxTrfReady())
   {
      putrsUSBUSART("\r\n1\r\nModo Comandos Cortos (SCS)\r\nSolo se pueden introducir
      valores hexadecimales con la ayuda del teclado, por favor usar mayúsculas\r\n\r\n");
      break;
   }
}

ncarbyte=0; // Se inicializa en cero para informar que se leerá el primer nibble del byte de entrada

while(1)
// Ciclo principal del modo SCS donde se reciben datos de la terminal del PC y del VDRIVE1 por SPI
{
   PORTAbits.RA3=0;     // Se apaga el LED de comunicación SPI
   USBTasks();

   if(mUSBUSARTIsTxTrfReady())
   // Si esta listo para la comunicación USB recibe datos
    {
      entusb=getsUSBUSART(input_buffer,32);
   }

   if(entusb>0)
   // Si hay alguna nueva entrada por parte de la terminal del PC
   // entusb es mayor a 0
   {
      datoru=input_buffer[0];    // Se entrega el dato recibido a datoru
      while(1)
      // Ciclo que retorna la entrada a la terminal del PC para que sea vista por
      // el usuario, la compone y la manda al VDRIVE1
      {
         if(datoru<'0'|datoru>'9'&datoru<'A'|datoru>'F')
            break;
         // Si la entrada no tiene caracteres hexadecimales no la acepta
         // y abandona el ciclo
```

```
USBTasks();
if(mUSBUSARTIsTxTrfReady())
// Si está la comunicación USB lista prosigue.
{

   if(ncarbyte==0)
   // Si está en el primer nibble
   {
      ncarbyte=1;    // Se coloca ncarbyte en 1 para informar que en el próximo ciclo
                  // continúe con el segundo nibble
      output_buffer[0]=datoru;            // Se retorna el dato que entro del PC
      mUSBUSARTTxRam((byte*)output_buffer,1);   // a la terminal del PC
      entusb=getsUSBUSART(input_buffer,32);  // se lee la entrada para evitar falsos retornos
      USBTasks();
      carbyte[0]=datoru;               // Se almacena el primer nibble
   }
   else
   {
   // Si esta en el segundo nibble
      ncarbyte=0;    // Se coloca ncarbyte en 0 para informar que en el próximo ciclo
                  // inicie en el primer nibble
      carbyte[1]=datoru;      // Se almacena el segundo nibble
      datoinuart=ha2bin(carbyte);      // Con esta función se convierte el vector que contiene
                     // al primer y al segundo nibble en un byte
      // El dato de entrada del PC es datoinuart
      if(datoinuart==0x0D)
      // Si el dato es un retorno de carril (0x0D)
      {
         putrsUSBUSART("D\n\r");
         entusb=getsUSBUSART(input_buffer,32);
         USBTasks();
         // Se publica una D por que el 0 ya se publicó
         // y se manda un /n/r para salto y retorno de carril
      }
      else
      // Si el dato no es un retorno de carril
      {
         if((datoinuart>=0x20&datoinuart<=0x7E)|(datoinuart>=0xA0&datoinuart<=0xFF))
         // Si el dato es un carácter imprimible
         {
            output_buffer[0]=datoru;
            output_buffer[1]=datoinuart;
            output_buffer[2]=';';
            mUSBUSARTTxRam((byte*)output_buffer,3);
            entusb=getsUSBUSART(input_buffer,32);
            USBTasks();
            // Imprime el segundo nibble, datoru
            // Imprime el ASCII del byte, datoinuart
            // Imprime un ";" para separar los bytes
         }
         else
         // Si el dato no es un carácter imprimible
         {
            output_buffer[0]=datoru;
            output_buffer[1]=' ';
            output_buffer[2]=';';
            mUSBUSARTTxRam((byte*)output_buffer,3);
            entusb=getsUSBUSART(input_buffer,32);
            USBTasks();
            // Imprime el segundo nibble, datoru
            // Imprime un espacio que representa el dato no imprimible
            // Imprime un ";" para separa los bytes
         }
      } // end else
```

```
            SPI_write(datoinuart);
            // Manda el dato por SPI al VDRIVE1

         } // end else del segundo nibble
         USBTasks();
         break;
         // Como el dato fue envidado sale del while

      } // end if(mUSBUSARTIsTxTrfReady())

   }// end while
   // Fin del ciclo que retorna la entrada a la terminal del PC para que sea vista por
   // el usuario, la compone y la manda al VDRIVE1

} //end if(entusb>0)

if(SPI_read(&leido)==0)
// Si el VDRIVE1 manda un dato nuevo la función SPI_read retorna un 0
{
   while(1)
   // Ciclo que imprime en la terminal del PC el primer dato
   // obtenido por el VDRIVE1, el dato se guarda en "leido"
   {
      USBTasks();
      if(mUSBUSARTIsTxTrfReady())
      // Si está listo para transmitir por el USB al PC
      {
         if(leido==0x0D)
         // Si es un retorno de carril (0x0D)
         {
            putrsUSBUSART("0D\n\r");
            entusb=getsUSBUSART(input_buffer,32);
            USBTasks();
            break;
            // Si el dato es un retorno de carril
            // imprime un 0D mas \n\r para salto
            // de linea y retorno de carril
            // Sale del while
         }
         else
         // Si no es un retorno de carril (0x0D)
         {
            UART_cyb(leido);  // Convierte el dato de entrada hexadecimal ASCII
                              // y lo imprime en la terminal del PC
            break;
            // Sale del while

         }
      } // end if(mUSBUSARTIsTxTrfReady())
   } // end while
   // Fin del while que lee e imprime el primer dato leído



   while(1)
   // Ciclo que imprime el resto de datos
   {
      PORTAbits.RA3=~PORTAbits.RA3;    // Se invierte el estado del LED de comunicación SPI
      USBTasks();
      if(mUSBUSARTIsTxTrfReady())
      // Si esta listo para transmitir por el USB al PC
      {
         if(SPI_read(&leido)==1)
         // Lee el siguiente dato y comprueba si es nuevo
```

```
            // Si no hay datos nuevos SPI_read retorna un 1
            {
               ncarbyte=0;
               break;
               // Se coloca a ncarbyte en 0 para indicar que se leerá el
               // primer nibble y se termina el while de lectura del resto de datos
            }

            if(leido==0x0D)
            // Si es un retorno de carril (0x0D)
            {
               putrsUSBUSART("0D\n\r");
               entusb=getsUSBUSART(input_buffer,32);
               USBTasks();
               // Si el dato es un retorno de carril
               // imprime un 0D mas \n\r para salto
               // de linea y retorno de carril
            }
            else
            // Si no es un retorno de carril (0x0D)
            {
               UART_cyb(leido);  // Convierte el dato de entrada hexadecimal ASCII
                              // y lo imprime en la terminal del PC
            }
         }
      } //end while
      // Fin del ciclo que lee el resto de datos
   }// end if(SPI_read(&leido)==0)
   // Fin del IF que lee e imprime los datos del VDRIVE1 en la terminal del PC

} // end main while
// Fin del ciclo principal del modo de comandos cortos.



//*******************************MODO ECS*************************************************************
//****************************************************************************************************
//****************************************************************************************************

modoecs:

// Se manda la instrucción SCS al VDRIVE1 para informarle
// que se comunique en modo comandos cortos

SPI_write(0x0D); // Se coloca un 0x0D para que las entradas anteriores sean descartadas
SPI_write('E');
SPI_write('C');
SPI_write('S');
SPI_write(0x0D);

while(1)
// Ciclo que muestra el mensaje de bienvenida al modo ECS
{
   USBTasks();
   if(mUSBUSARTIsTxTrfReady())
   {
      putrsUSBUSART("\r\n2\r\nModo Comandos Extendidos (ECS)\r\n\r\n");
      break;
   }
}

while(1)
// Ciclo principal del modo ECS donde se reciben datos de la terminal del PC y del VDRIVE1 por SPI
{
   PORTAbits.RA3=0;  // Se apaga el LED de comunicación SPI
```

```
USBTasks();

if(mUSBUSARTIsTxTrfReady())
// Si esta listo para la comunicación USB, recibe datos
 {
   entusb=getsUSBUSART(input_buffer,32);
}
if(entusb>0)
// Si hay alguna nueva entrada por parte de la terminal del PC
// entusb es mayor a 0
{
   datoinuart=input_buffer[0];      // Se entrega el dato recibido a datoinuart
   while(1)
   // Ciclo que retorna la entrada a la terminal del PC para que sea vista por
   // el usuario y la manda por SPI al VDRIVE1
   {
      USBTasks();
      if(mUSBUSARTIsTxTrfReady())
      // Si está la comunicación USB lista prosigue.
      {

         if(datoinuart==0x0D)
         // Si el dato es un retorno de carril (0x0D)
         {
            putrsUSBUSART("\r\n");
            entusb=getsUSBUSART(input_buffer,32);
            USBTasks();
            // Se manda un /n/r para salto y retorno de carril
         }
         else
         // Si el dato no es un retorno de carril (0x0D)
         {
            UART_char(datoinuart);
            entusb=getsUSBUSART(input_buffer,32);
            USBTasks();
            // Se imprime el dato recibido en el terminal del PC
         }// end else que detecta si el dato es un retorno de carril

         SPI_write(datoinuart);
         // Manda el dato por SPI al VDRIVE1
         break;
         // Como el dato fue envidado y sale del while
      }//end if(mUSBUSARTIsTxTrfReady())

   }// end while
   // Fin del ciclo principal del modo ECS donde se reciben
   // datos de la terminal del PC y del VDRIVE1 por SPI

}// end if(entusb>0)

if(SPI_read(&leido)==0)
// Si el VDRIVE1 manda un dato nuevo la función SPI_read retorna un 0
{
   while(1)
   // Ciclo que imprime en la terminal del PC el primer dato
   // obtenido por el VDRIVE1, el dato se guarda en "leido"
   {
      USBTasks();
      if(mUSBUSARTIsTxTrfReady())
      // Si esta listo para transmitir por el USB al PC
      {
         if(leido==0x0D)
         // Si es un retorno de carril (0x0D)
         {
            putrsUSBUSART("\r\n");
```

```
            entusb=getsUSBUSART(input_buffer,32);
            USBTasks();
            break;
            // Si el dato es un retorno de carril
            // manda \n\r para salto
            // de linea y retorno de carril
            // Sale del while
         }
         else
         // Si no es un retorno de carril (0x0D)
         {
            UART_char(leido);
            entusb=getsUSBUSART(input_buffer,32);
            USBTasks();
            break;
            // Se imprime el dato leido en la terminal del PC
            // sale del while
         }
      } // end if(mUSBUSARTIsTxTrfReady())
   } // end while


   while(1)
   // Ciclo que imprime el resto de datos
   {
      PORTAbits.RA3=~PORTAbits.RA3;    // Se invierte el estado del LED de comunicación SPI
      USBTasks();
      if(mUSBUSARTIsTxTrfReady())
      // Si esta listo para transmitir por el USB al PC
      {
         if(SPI_read(&leido)==1)
         // Lee el siguiente dato y comprueba si es nuevo
         // Si no hay datos nuevos SPI_read retorna un 1
         {
            break;
            // se termina el while de lectura del resto de datos
         }

         if(leido==0x0D)
         // Si es un retorno de carril (0x0D)
         {
            putrsUSBUSART("\r\n");
            entusb=getsUSBUSART(input_buffer,32);
            USBTasks();
            // Si el dato es un retorno de carril
            // manda \n\r para salto
            // de linea y retorno de carril
         }
         else
         {
            UART_char(leido);
            entusb=getsUSBUSART(input_buffer,32);
            USBTasks();
            // Se imprime el dato "leido" en la terminal del PC
         }
      }
   } // end while
   // Fin del ciclo que imprime el resto de datos

  }//end if(SPI_read(&leido)==0)
  // Fin del IF que lee e imprime los datos del VDRIVE1 en la terminal del PC

} // end main while
```

```
} // end void main


//** SUBRUTINAS ***************************************************************************/
//*****************************************************************************************/
//*****************************************************************************************/
//*****************************************************************************************/

/** Subrutina para iniciar la comunicación USB*********************************************/

static void InitializeSystem(void)
{

    ADCON1 |= 0x0F;                 // Default all pins to digital

    #if defined(USE_USB_BUS_SENSE_IO)
    tris_usb_bus_sense = INPUT_PIN; // See io_cfg.h
    #endif


    mInitializeUSBDriver();         // See usbdrv.h

}


/** Subrutina del serivcio de comunicación USB ********************************************/

void USBTasks(void)
{
    /*
     * Servicing Hardware
     */
    USBCheckBusStatus();                    // Must use polling method
    if(UCFGbits.UTEYE!=1)
        USBDriverService();                 // Interrupt or polling method

    #if defined(USB_USE_CDC)
    CDCTxService();
    #endif
}


/** Subrutina para imprimir caracteres en la terminal del PC **********************************/
void UART_char(char datouartc)
{

   if((datouartc>=0x20&datouartc<=0x7E)|(datouartc>=0xA0&datouartc<=0xFF))
   // Detecta si el dato es un carácter imprimible
   {
      output_buffer[0]=datouartc;
      mUSBUSARTTxRam((byte*)output_buffer,1);
      // Manda el dato a la terminal del PC
   }
   else
   {
      output_buffer[0]=' ';
      mUSBUSARTTxRam((byte*)output_buffer,1);
      // Manda un espacio " " a la terminal del PC
   }
}

/** Subrutina para imprimir el hexadecimal y el carácter ASCII del dato de entrada**************/
void UART_cyb(char dato)
{
   char dato2;
```

```
   char tdato;
   char tdato2;


   dato2=0xF0&dato;
   dato2=dato2>>4;
   dato2=0x0F&dato2;

   // Obtiene los nibbles ACSII en hexadecimal del dato
   tdato2=0x0F&dato2;
   if(tdato2<10)
   {
      tdato2=tdato2+'0';
   }
   else
   {
      tdato2=tdato2+'A'-0x0A;
   }

   tdato=0x0F&dato;
   if(tdato<10)
   {
      tdato=tdato+'0';
   }
   else
   {
      tdato=tdato+'A'-0x0A;
   }

   if((dato>=0x20&dato<=0x7E)|(dato>=0xA0&dato<=0xFF))
   // Si el dato es un carácter imprimible
   {
      output_buffer[0]=tdato2;
      output_buffer[1]=tdato;
      output_buffer[2]=dato;
      output_buffer[3]=';';
      mUSBUSARTTxRam((byte*)output_buffer,4);
      // Imprime los nibbles del hexadecimal en ASCII
      // Imprime el dato en ASCII
      // Imprime un ";" para separa los bytes
   }
   else
   // Si el dato no es un carácter imprimible
   {
      output_buffer[0]=tdato2;
      output_buffer[1]=tdato;
      output_buffer[2]=' ';
      output_buffer[3]=';';
      mUSBUSARTTxRam((byte*)output_buffer,4);
      // Imprime los nibbles del hexadecimal en ASCII
      // Imprime un espacio que representa el dato no imprimible
      // Imprime un ";" para separa los bytes

   }
}


/** Subrutina que convierte los nibbles hexadecimal ASCII en binario**********************************/
char ha2bin(char carbyte[2])
{
// Los nibbles de entrada deben estar en un vector

   char datob;
   char datob2;
```

```
   datob2=0;
   if(carbyte[0]>='0'&carbyte[0]<='9')
      datob2=carbyte[0]-'0';
   if(carbyte[0]>='A'&carbyte[0]<='F')
      datob2=carbyte[0]-'A'+0x0A;

   datob=0;
   if(carbyte[1]>='0'&carbyte[0]<='9')
      datob=carbyte[1]-'0';
   if(carbyte[1]>='A'&carbyte[0]<='F')
      datob=carbyte[1]-'A'+0x0A;

   datob2=datob2<<4;
   datob2=datob2&0xF0;
   datob=datob+datob2;
   return datob;

   // Retorna el dato en binario.
}
```

ANEXO B. Código matriz de LEDS USB

```
;************************************ MDL-MUSB.ASM ***********************************************
; Este programa lee el contenido de el archivo TEXTO.TXT de una memoria USB y lo manda a una matriz
; de LEDS. El texto debe estar bajo la codificación ISO-8859-1 del alfabeto latino o Europa
; Occidental para que sean reconocidos los caracteres especiales del idioma español como la ñ, ü y
; las vocales con tildes. Se utiliza un PIC16F877A con un oscilador de 4MHz, conectado al modulo
; VDRIVE1 para leer las memorias USB y a una matriz de leds que funciona con registros de
; desplazamiento para mostrar caracteres.
;
; El programa busca al inicio el archivo TEXTO.TXT, lo abre y manda su contenido de 3 en 3
; caracteres a la matriz de leds, tiene un botón para interrumpir la lectura, hay que tenerlo
; pulsado por mas de 3 caracteres para terminar, aparece un mensaje "Fin Lectura", en este momento
; se puede extraer la memoria para finalizar o para cambiarla por otra luego de que este reconocida
; se pulsa el botón nuevamente para leer el archivo. Este botón también se puede utilizar para
; interrumpir la lectura y comenzar la lectura del archivo desde el principio.
;
; Cuenta también con una entrada analógica para conectar un potenciómetro con el cual se puede
; controlar la velocidad de desplazamiento del texto en la matriz de leds. Para la conexión del
; modulo VDRIVE1 ver la librería VDRIVE.INC y para la matriz de leds la librería MATRIZL.INC
;
;
;     ================================================================================
;     ||                              PROYECTO DE GRADO                             ||
;     ||                                                                            ||
;     ||     ESTUDIO Y DISEÑO DE ALGORITMOS E INTERFAZ ELECTRÓNICA DE COMUNICACIÓN  ||
;     ||    ENTRE MICROCONTROLADORES SERIE PIC Y DISPOSITIVOS DE ALMACENAMIENTO USB ||
;     ||                                                                            ||
;     ||                          Juan Pablo Ruiz Rosero                            ||
;     ||                          Ingeniería Electrónica                            ||
;     ||                           Universidad de Nariño                            ||
;     ||                               Octubre 2008                                 ||
;     ================================================================================
;
;
;*******************************************************************************************




; ZONA DE DATOS *****************************************************************************


        list       P=16F877A                    ; Procesador utilizado.
        #include  <P16F877A.INC>                ; En este fichero se definen las etiquetas del PIC.

        __CONFIG _CP_OFF & _WDT_OFF & _BODEN_ON & _PWRTE_ON & _XT_OSC & _LVP_OFF & _CPD_ON
        ; Configuración del microcontrolador

        CBLOCK  20h             ; Definición de dirección para las variables
        CONTADOR
        CONTCAR
        ENDC

#DEFINE BOTON1  PORTC,5         ; Define el pin para el botón 1


; ZONA DE LIBRERIAS *************************************************************************

        ORG     0
        GOTO    INICIO
        INCLUDE "D:\Electronica\Tesis\Librerias\MATRIZL.INC"
        ; Librería para el manejo de la Matriz de Leds
```

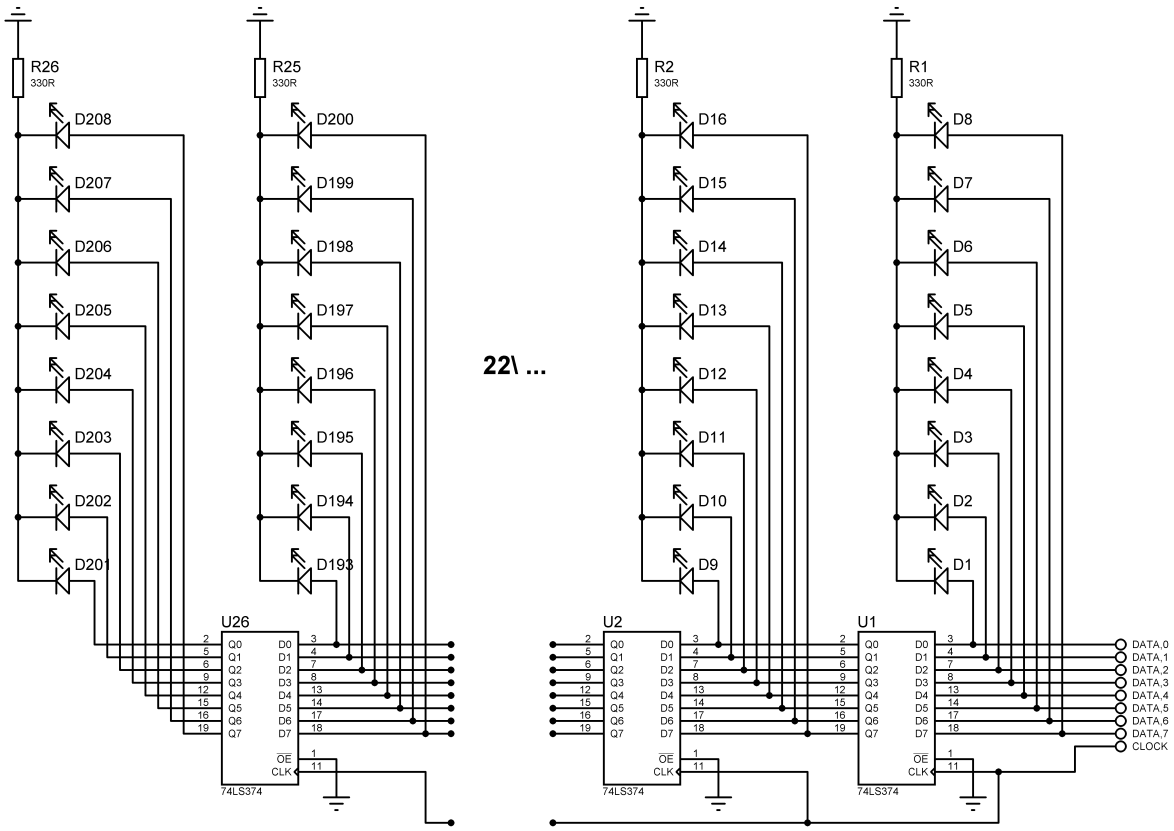
```
        ; Se coloca esta librería al principio del código ya que contiene una tabla que debe
        ; superar la dirección 0xFF del código del programa.
        INCLUDE "D:\Electronica\Tesis\Librerias\RETARDOS.INC"
        ; Libreria para el manejo de retardos.
        INCLUDE "D:\Electronica\Tesis\Librerias\VDRIVE.INC"
        INCLUDE "D:\Electronica\Tesis\Librerias\BINAD99.INC"
        ; Libreria para el manejo del modulo VDRIVE1


; ZONA DE CODIGO ***************************************************************************
INICIO
        BSF     STATUS,RP0              ; Pone en 1 el bit 5 del STATUS. Acceso al Banco 1.
        BSF     BOTON1                  ; Coloca el pin del botón como entrada
        BCF     STATUS,RP0              ; Acceso al banco 0
        CALL    MDLINIC                 ; Inicializa la comunicación con la matriz de leds
        CALL    VDINIC                  ; Inicializa la comunicación con el VDRIVE1
        CALL    VDIPHM                  ; Coloca al VDRIVE1 en modo datos binario
        CALL    VDRRC                   ; Lee la respuesta del comando


PRINCIPAL
        CALL    VDUSBP                  ; Testea si esta conectada la memoria USB
        BTFSC   STATUS,C
        GOTO    NOMEMUSB                ; Si no esta conectada da salta a NOMEMUSB
        CALL    ABRIRARCHIVO            ; Si esta conectada llama a ABRIRARCHIVO para abrir el
                                        ; archivo a leer

LECTURA
        BTFSS   BOTON1                  ; Testea si el boton 1 es presionado
        GOTO    FINLECTURA              ; Si es presionado salta a FINLECTURA
        CALL    LEER3C                  ; Si no es presionado continua y manda un comando para
        MOVLW   0x03                    ; leer 3 caracteres
        MOVWF   CONTCAR                 ; Inicializa al contador de caracteres CONTCAR en 3

LECTURA1
        CALL    VDSPIRN                 ; Lee un dato nuevo del VDRIVE1
        CALL    MLETRA                  ; Manda el dato a la Matriz de leds
        DECFSZ  CONTCAR,F               ; Decrementa a CONTCAR para saber cuantos caracteres
        GOTO    LECTURA1                ; se han leido, salta a LECTURA1 para continuar.
        CALL    VDRCF                   ; Si acabo de leer los 3 caracteres detecta si el
        BTFSC   STATUS,C                ; archivo llego a su fin.
        GOTO    LECTURA                 ; Si no ha llegado a su fin salta a LECTURA para leer
                                        ; otros 3 caracteres
        CALL    INICIOARCHIVO           ; Si el archivo llegó a su fin manda un comando para
                                        ; que el puntero se coloque en la posición inicial
        GOTO    LECTURA                 ; Salta a lectura para continuar con 3 caracteres.


FINLECTURA
        CALL    Retardo_100ms           ; Llama a un retardo de 100ms para dar un tiempo al botón 1
        CALL    CERRARACHIVO            ; Cierra el archivo
        CALL    VDSUD                   ; Coloca la memoria en suspensión
        CALL    VDRRC                   ; Lee la respuesta del comando
        CALL    MENSAJEFIN              ; Manda un mensaje a la matriz de fin de lectura

FINLECTURA2
        CALL    VDSPIR                  ; Lee datos del VDRIVE1 para que actualice sus datos
        BTFSC   BOTON1                  ; Testea si el botón esta pulsado
        GOTO    FINLECTURA2             ; Si no esta pulsado salta a FINLECTURA2
        CALL    Retardo_50ms            ; Si esta pulsado espera 50ms para evitar ruidos
FINLECTURA3
        BTFSS   BOTON1                  ; Testea si el botón se ha dejado de pulsar
        GOTO    FINLECTURA3             ; Si no se ha dejado de pulsar salta a FINLECTURA3

        CALL    VDWKD                   ; Saca la memoria del estado de suspensión
```

```
        CALL    VDRRC                   ; Lee la respuesta del comando
        GOTO    PRINCIPAL               ; Salta a principal para empezar de nuevo




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

INICIOARCHIVO
; Manda una orden al VDRIVE1 para que el puntero se coloque al inicio del archivo
        CALL    VDSEK
        MOVLW   0x00
        CALL    VDSPIW
        MOVLW   0x00
        CALL    VDSPIW
        MOVLW   0x00
        CALL    VDSPIW
        MOVLW   0x00
        CALL    VDSPIW
        MOVLW   0x0D
        CALL    VDSPIW
        CALL    VDRRC
        RETURN



LEER3C
; Manda una orden al VDRIVE1 para que lea 3 caracteres del archivo abierto
        CALL    VDRDF
        MOVLW   0x00
        CALL    VDSPIW
        MOVLW   0x00
        CALL    VDSPIW
        MOVLW   0x00
        CALL    VDSPIW
        MOVLW   0x03
        CALL    VDSPIW
        MOVLW   0x0D
        CALL    VDSPIW
        RETURN

CERRARACHIVO
; Manda la orden al VDRIVE1 de cerrar el archivo TEXTO.TXT
        CALL    VDCLF
        MOVLW   'T'
        CALL    VDSPIW
        MOVLW   'E'
        CALL    VDSPIW
        MOVLW   'X'
        CALL    VDSPIW
        MOVLW   'T'
        CALL    VDSPIW
        MOVLW   'O'
        CALL    VDSPIW
        MOVLW   '.'
        CALL    VDSPIW
        MOVLW   'T'
        CALL    VDSPIW
        MOVLW   'X'
        CALL    VDSPIW
        MOVLW   'T'
        CALL    VDSPIW
        MOVLW   0x0D
```

```
        CALL    VDSPIW
        RETURN


ABRIRARCHIVO
; Manda la orden al VDRIVE1 para que abra el archivo TEXTO.TXT
        CALL    VDOPR
        MOVLW   'T'
        CALL    VDSPIW
        MOVLW   'E'
        CALL    VDSPIW
        MOVLW   'X'
        CALL    VDSPIW
        MOVLW   'T'
        CALL    VDSPIW
        MOVLW   'O'
        CALL    VDSPIW
        MOVLW   '.'
        CALL    VDSPIW
        MOVLW   'T'
        CALL    VDSPIW
        MOVLW   'X'
        CALL    VDSPIW
        MOVLW   'T'
        CALL    VDSPIW
        MOVLW   0x0D
        CALL    VDSPIW
        CALL    VDRRC
        RETURN



MENSAJEFIN
; Manda el texto "Fin Lectura a la matriz de leds"
        MOVLW   ' '
        CALL    MLETRA
        MOVLW   'F'
        CALL    MLETRA
        MOVLW   'i'
        CALL    MLETRA
        MOVLW   'n'
        CALL    MLETRA
        MOVLW   ' '
        CALL    MLETRA
        MOVLW   'L'
        CALL    MLETRA
        MOVLW   'e'
        CALL    MLETRA
        MOVLW   'c'
        CALL    MLETRA
        MOVLW   't'
        CALL    MLETRA
        MOVLW   'u'
        CALL    MLETRA
        MOVLW   'r'
        CALL    MLETRA
        MOVLW   'a'
        CALL    MLETRA
        MOVLW   ' '
        CALL    MLETRA
        RETURN


NOMEMUSB
; Manda el mensaje "Conectar Mem USB" a la matriz de leds
```

```
        MOVLW   'C'
        CALL    MLETRA
        MOVLW   'o'
        CALL    MLETRA
        MOVLW   'n'
        CALL    MLETRA
        MOVLW   'e'
        CALL    MLETRA
        MOVLW   'c'
        CALL    MLETRA
        MOVLW   't'
        CALL    MLETRA
        MOVLW   'a'
        CALL    MLETRA
        MOVLW   'r'
        CALL    MLETRA
        MOVLW   ' '
        CALL    MLETRA
        MOVLW   'M'
        CALL    MLETRA
        MOVLW   'e'
        CALL    MLETRA
        MOVLW   'm'
        CALL    MLETRA
        MOVLW   ' '
        CALL    MLETRA
        MOVLW   'U'
        CALL    MLETRA
        MOVLW   'S'
        CALL    MLETRA
        MOVLW   'B'
        CALL    MLETRA
        MOVLW   ' '
        CALL    MLETRA
        GOTO    PRINCIPAL


; FIN DE CODIGO ********************************************************************************
        END
```

ANEXO C. Esquema circuital matriz de LEDS



ANEXO D. Código librería de la matriz de LEDS

```
;*************************** Librería "MATRIZL.INC" ****************************************
; Esta librería permite controlar una MATRIZ de LEDS que funcione con registros de
; desplazamiento mediante un microcontrolador serie PIC16. Se incluyen dos funciones, una de
; inicialización y otro para imprimir caracteres, la velocidad de desplazamiento del texto
; depende de una entrada análoga definida, lo cual permite modificarla con la ayuda de un
; potenciómetro.
;
; Puerto y pines usados en el microcontrolador PIC16F877
; Puerto B, pines del 33 al 40: Se usa para mandar los datos a la primera columna de la matriz
; de LEDS.
; Bit RC4 o PORTC,4 pin 23: Señal de reloj de la matriz de LEDS, desde aquí se manda el impulso
; para que en la matriz se genere un desplazamiento de datos.
;
;
;     ================================================================================
;     ||                              PROYECTO DE GRADO                             ||
;     ||                                                                            ||
;     ||     ESTUDIO Y DISEÑO DE ALGORITMOS E INTERFAZ ELECTRÓNICA DE COMUNICACIÓN  ||
;     ||    ENTRE MICROCONTROLADORES SERIE PIC Y DISPOSITIVOS DE ALMACENAMIENTO USB ||
;     ||                                                                            ||
;     ||                          Juan Pablo Ruiz Rosero                            ||
;     ||                          Ingeniería Electrónica                            ||
;     ||                           Universidad de Nariño                            ||
;     ||                               Octubre 2008                                 ||
;     ================================================================================
;

;
;*********************************************************************************************

; ZONA DE DATOS ***************************************************************************

        CBLOCK
        LETRAM          ; Define direcciones en memoria para las variables
        MDLRETC
        REGM1
        REGM2
        REGM3
        REGM4
        REGM5
        ENDC


#DEFINE MDLPCLK PORTD,7         ; Define el bit para la señal de reloj
#DEFINE MDLDATA PORTB           ; Define el puerto de datos para la matriz

; ZONA DE FUNCIONES ***********************************************************************

; FUNCION DE INICIALIZACIÓN ***************************************************************
; Esta función configura los pines utilizados para el manejo de la matriz de leds, también
; configura el conversor análogo digital para entrar los datos del potenciómetro que controla
; la velocidad de desplazamiento del texto.
MDLINIC
        BSF     STATUS,RP0              ; Pone en 1 el bit 5 del STATUS. Acceso al Banco 1.
        CLRF    MDLDATA                 ; Coloca al puerto de datos como salida
        BCF     MDLPCLK                 ; Coloca al pin de reloj como salida
        MOVLW   b'00001110'             ; Justificación izquierda, solo RA0 como entrada análoga
        MOVWF   ADCON1                  ; para la configuración del conversor A/D
        BCF     STATUS,RP0              ; Acceso al banco 0
        BCF     MDLPCLK                 ; Inicializa el reloj de la matriz en cero
        MOVLW   b'01000001'             ; Configuración conversor A/D
        MOVWF   ADCON0                  ; Fosc/8, entrada por chan0, god 0, adon 1 encendido
```

```
        MOVLW   ' '                     ; Envía espacios en blanco a la matriz para limpiarla
        CALL    MLETRA
        MOVLW   ' '
        CALL    MLETRA
        MOVLW   ' '
        CALL    MLETRA
        MOVLW   ' '
        CALL    MLETRA

        RETURN


; FUNCION DE ESCRITURA DE LETRAS EN MATRIZ MLETRA************************************************
; Esta función toma el dato que esta en el registro de trabajo W y lo manda como carácter a la
; matriz de leds, si no es un carácter imprimible manda un espacio.
; Ejemplo: Manda las palabra UDENAR a la matriz de leds.
;       MOVLW   'U'
;       CALL    MLETRA
;       MOVLW   'D'
;       CALL    MLETRA
;       MOVLW   'E'
;       CALL    MLETRA
;       MOVLW   'N'
;       CALL    MLETRA
;       MOVLW   'A'
;       CALL    MLETRA
;       MOVLW   'R'
;       CALL    MLETRA
;*****************************************************************************************************
MLETRA
        MOVWF   LETRAM          ; Guarda el dato del registro de trabajo W en LETRAM
        SUBLW   0x80            ; Comprueba si el dato de la letra es mayor o igual de 0x80 para
        BTFSC   STATUS,C        ; conocer si es un carácter especial
        GOTO    MLETRA1         ; Si es menor continua a MLETRA1
        GOTO    MLETESPC        ; Si es mayor o igual continua a MLETESPC

MLETRA1
        MOVLW   0x20            ; Comprueba si el dato es mayor o igual a 0x20 para conocer si es
        SUBWF   LETRAM,W        ; un carácter imprimible
        BTFSC   STATUS,C
        GOTO    MLETRA2         ; Si es mayor o igual el dato es un carácter imprimible
        GOTO    MletEsp         ; Si es menor el dato es una instrucción y no es imprimible
                                ; por lo cual va a MLETESP

MLETRA2
        MOVLW   0x20            ; Resta el valor 0x20 a LETRAM
        SUBWF   LETRAM,F
        MOVF    LETRAM,W        ; Suma el valor de LETRAM al contador de programa PCL para realizar
        ADDWF   PCL,F           ; un salto al carácter correspondiente.

        GOTO    MletEsp         ; Imprime " "
        GOTO    MletAdm         ; Imprime "!"
        GOTO    MletComi        ; Imprime """
        GOTO    MletNum         ; Imprime "#"
        GOTO    MletPes         ; Imprime "$"
        GOTO    MletPcen        ; Imprime "%"
        GOTO    MletAnd         ; Imprime "&"
        GOTO    MletScom        ; Imprime "'"
        GOTO    MletIpar        ; Imprime "("
        GOTO    MletFcom        ; Imprime ")"
        GOTO    MletAste        ; Imprime "*"
        GOTO    MletMas         ; Imprime "+"
        GOTO    Mletcom         ; Imprime ","
```

```
GOTO    MletMenos       ; Imprime "-"
GOTO    MletPunto       ; Imprime "."
GOTO    MletSlash       ; Imprime "/"
GOTO    Mlet0           ; Imprime "0"
GOTO    Mlet1           ; Imprime "1"
GOTO    Mlet2           ; Imprime "2"
GOTO    Mlet3           ; Imprime "3"
GOTO    Mlet4           ; Imprime "4"
GOTO    Mlet5           ; Imprime "5"
GOTO    Mlet6           ; Imprime "6"
GOTO    Mlet7           ; Imprime "7"
GOTO    Mlet8           ; Imprime "8"
GOTO    Mlet9           ; Imprime "9"
GOTO    MletDosp        ; Imprime ":"
GOTO    MletPcoma       ; Imprime ";"
GOTO    MletMenor       ; Imprime "<"
GOTO    MletIgual       ; Imprime "="
GOTO    MletMayor       ; Imprime ">"
GOTO    MletPreg        ; Imprime "?"
GOTO    MletArrob       ; Imprime "@"
GOTO    MletA           ; Imprime "A"
GOTO    MletB           ; Imprime "B"
GOTO    MletC           ; Imprime "C"
GOTO    MletD           ; Imprime "D"
GOTO    MletE           ; Imprime "E"
GOTO    MletF           ; Imprime "F"
GOTO    MletG           ; Imprime "G"
GOTO    MletH           ; Imprime "H"
GOTO    MletI           ; Imprime "I"
GOTO    MletJ           ; Imprime "J"
GOTO    MletK           ; Imprime "K"
GOTO    MletL           ; Imprime "L"
GOTO    MletM           ; Imprime "M"
GOTO    MletN           ; Imprime "N"
GOTO    MletO           ; Imprime "O"
GOTO    MletP           ; Imprime "P"
GOTO    MletQ           ; Imprime "Q"
GOTO    MletR           ; Imprime "R"
GOTO    MletS           ; Imprime "S"
GOTO    MletT           ; Imprime "T"
GOTO    MletU           ; Imprime "U"
GOTO    MletV           ; Imprime "V"
GOTO    MletW           ; Imprime "W"
GOTO    MletX           ; Imprime "X"
GOTO    MletY           ; Imprime "Y"
GOTO    MletZ           ; Imprime "Z"
GOTO    Mletillav       ; Imprime "["
GOTO    Mletbizq        ; Imprime "\"
GOTO    Mletfllav       ; Imprime "]"
GOTO    Mletgorro       ; Imprime "^"
GOTO    Mletsubrall     ; Imprime "_"
GOTO    Mlettildi       ; Imprime "`"
GOTO    MletAm          ; Imprime "a"
GOTO    MletBm          ; Imprime "b"
GOTO    MletCm          ; Imprime "c"
GOTO    MletDm          ; Imprime "d"
GOTO    MletEm          ; Imprime "e"
GOTO    MletFm          ; Imprime "f"
GOTO    MletGm          ; Imprime "g"
GOTO    MletHm          ; Imprime "h"
GOTO    MletIm          ; Imprime "i"
GOTO    MletJm          ; Imprime "j"
GOTO    MletKm          ; Imprime "k"
GOTO    MletLm          ; Imprime "l"
GOTO    MletMm          ; Imprime "m"
```

```
        GOTO    MletNm          ; Imprime "n"
        GOTO    MletOm          ; Imprime "o"
        GOTO    MletPm          ; Imprime "p"
        GOTO    MletQm          ; Imprime "q"
        GOTO    MletRm          ; Imprime "r"
        GOTO    MletSm          ; Imprime "s"
        GOTO    MletTm          ; Imprime "t"
        GOTO    MletUm          ; Imprime "u"
        GOTO    MletVm          ; Imprime "v"
        GOTO    MletWm          ; Imprime "w"
        GOTO    MletXm          ; Imprime "x"
        GOTO    MletYm          ; Imprime "y"
        GOTO    MletZm          ; Imprime "z"
        GOTO    Mleticorch      ; Imprime "{"
        GOTO    Mlettuberia     ; Imprime "|"
        GOTO    Mletfcorch      ; Imprime "}"
        GOTO    Mletvirgu       ; Imprime "~"
FINTABLA

        IF (FINTABLA > 0xFF)
                ERROR   "¡CUIDADO!: La tabla ha superado el tamaño de la página de los"
                MESSG   "primeros 256 bytes de memoria ROM. NO funcionará correctamente."
        ENDIF
        ; Se comprueba que esta tabla de datos no quede después de la dirección de programa
        ; 0xFF para que el salto no tenga errores, si queda después se produce un mensaje
        ; de error en la compilación.

; CARACTERES ESPECIALES ********************************************************************
; ASCII MAYOR A 0x80 ***********************************************************************
MLETESPC
        MOVLW   'á'             ; Se compara LETRAM con el valor de cada uno de los caracteres
        SUBWF   LETRAM,W        ; especiales usados en la lengua española, si es igual realiza
        BTFSC   STATUS,Z        ; un salto al lugar correspondiente.
        GOTO    MletAmt
        MOVLW   'é'
        SUBWF   LETRAM,W
        BTFSC   STATUS,Z
        GOTO    MletEmt
        MOVLW   'í'
        SUBWF   LETRAM,W
        BTFSC   STATUS,Z
        GOTO    MletImt
        MOVLW   'ó'
        SUBWF   LETRAM,W
        BTFSC   STATUS,Z
        GOTO    MletOmt
        MOVLW   'ú'
        SUBWF   LETRAM,W
        BTFSC   STATUS,Z
        GOTO    MletUmt
        MOVLW   'Á'
        SUBWF   LETRAM,W
        BTFSC   STATUS,Z
        GOTO    MletAmt
        MOVLW   'É'
        SUBWF   LETRAM,W
        BTFSC   STATUS,Z
        GOTO    MletEmt
        MOVLW   'Í'
        SUBWF   LETRAM,W
        BTFSC   STATUS,Z
        GOTO    MletImt
        MOVLW   'Ó'
        SUBWF   LETRAM,W
        BTFSC   STATUS,Z
```

```
        GOTO    MletOmt
        MOVLW   'Ú'
        SUBWF   LETRAM,W
        BTFSC   STATUS,Z
        GOTO    MletUmt
        MOVLW   'ñ'
        SUBWF   LETRAM,W
        BTFSC   STATUS,Z
        GOTO    Mletñmt
        MOVLW   'Ñ'
        SUBWF   LETRAM,W
        BTFSC   STATUS,Z
        GOTO    Mletñmt
        MOVLW   'ü'
        SUBWF   LETRAM,W
        BTFSC   STATUS,Z
        GOTO    MletUdir
        MOVLW   'Ü'
        SUBWF   LETRAM,W
        BTFSC   STATUS,Z
        GOTO    MletUdir



; TABLAS ************************************************************************************
; ************************************************************************************
; ************************************************************************************

; TABLA SIMBOLOS Y NUMEROS*****************************************************************
; En esta parte del código se encuentra los datos que enviara la matriz de led por cada carácter
; correspondiente soportado, estos datos están en 5 bytes, cada byte representa una columna que
; juntos forman el carácter, estos datos se guardan en los registros, REGM1, REGM2, REGM3, REGM4
; y REGM5, después de esto se salta a MletEscrib, lugar del programa donde se envían los datos.


MletEsp
        MOVLW   b'00000000'
        MOVWF   REGM1
        MOVLW   b'00000000'
        MOVWF   REGM2
        MOVLW   b'00000000'
        MOVWF   REGM3
        MOVLW   b'00000000'
        MOVWF   REGM4
        MOVLW   b'00000000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletAdm
        MOVLW   b'00000000'
        MOVWF   REGM1
        MOVLW   b'00000000'
        MOVWF   REGM2
        MOVLW   b'10011110'
        MOVWF   REGM3
        MOVLW   b'00000000'
        MOVWF   REGM4
        MOVLW   b'00000000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletComi
        MOVLW   b'00000000'
        MOVWF   REGM1
```

```
        MOVLW   b'00001110'
        MOVWF   REGM2
        MOVLW   b'00000000'
        MOVWF   REGM3
        MOVLW   b'00001110'
        MOVWF   REGM4
        MOVLW   b'00000000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletNum
        MOVLW   b'00101000'
        MOVWF   REGM1
        MOVLW   b'11111110'
        MOVWF   REGM2
        MOVLW   b'00101000'
        MOVWF   REGM3
        MOVLW   b'11111110'
        MOVWF   REGM4
        MOVLW   b'00101000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletPes
        MOVLW   b'01001000'
        MOVWF   REGM1
        MOVLW   b'01010100'
        MOVWF   REGM2
        MOVLW   b'11111110'
        MOVWF   REGM3
        MOVLW   b'01010100'
        MOVWF   REGM4
        MOVLW   b'00100100'
        MOVWF   REGM5
        GOTO    MletEscrb

MletPcen
        MOVLW   b'01000110'
        MOVWF   REGM1
        MOVLW   b'00100110'
        MOVWF   REGM2
        MOVLW   b'00010000'
        MOVWF   REGM3
        MOVLW   b'11001000'
        MOVWF   REGM4
        MOVLW   b'11000100'
        MOVWF   REGM5
        GOTO    MletEscrb

MletAnd
        MOVLW   b'01101100'
        MOVWF   REGM1
        MOVLW   b'10010010'
        MOVWF   REGM2
        MOVLW   b'10101010'
        MOVWF   REGM3
        MOVLW   b'01000100'
        MOVWF   REGM4
        MOVLW   b'10100000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletScom
        MOVLW   b'00000000'
        MOVWF   REGM1
```

```
        MOVLW   b'00001010'
        MOVWF   REGM2
        MOVLW   b'00000110'
        MOVWF   REGM3
        MOVLW   b'00000000'
        MOVWF   REGM4
        MOVLW   b'00000000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletIpar
        MOVLW   b'00000000'
        MOVWF   REGM1
        MOVLW   b'00111000'
        MOVWF   REGM2
        MOVLW   b'01000100'
        MOVWF   REGM3
        MOVLW   b'10000010'
        MOVWF   REGM4
        MOVLW   b'00000000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletFcom
        MOVLW   b'00000000'
        MOVWF   REGM1
        MOVLW   b'10000010'
        MOVWF   REGM2
        MOVLW   b'01000100'
        MOVWF   REGM3
        MOVLW   b'00111000'
        MOVWF   REGM4
        MOVLW   b'00000000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletAste
        MOVLW   b'00101000'
        MOVWF   REGM1
        MOVLW   b'00010000'
        MOVWF   REGM2
        MOVLW   b'01111100'
        MOVWF   REGM3
        MOVLW   b'00010000'
        MOVWF   REGM4
        MOVLW   b'00101000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletMas
        MOVLW   b'00010000'
        MOVWF   REGM1
        MOVLW   b'00010000'
        MOVWF   REGM2
        MOVLW   b'01111100'
        MOVWF   REGM3
        MOVLW   b'00010000'
        MOVWF   REGM4
        MOVLW   b'00010000'
        MOVWF   REGM5
        GOTO    MletEscrb

Mletcom
        MOVLW   b'00000000'
        MOVWF   REGM1
```

```
        MOVLW   b'10100000'
        MOVWF   REGM2
        MOVLW   b'01100000'
        MOVWF   REGM3
        MOVLW   b'00000000'
        MOVWF   REGM4
        MOVLW   b'00000000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletMenos
        MOVLW   b'00010000'
        MOVWF   REGM1
        MOVLW   b'00010000'
        MOVWF   REGM2
        MOVLW   b'00010000'
        MOVWF   REGM3
        MOVLW   b'00010000'
        MOVWF   REGM4
        MOVLW   b'00010000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletPunto
        MOVLW   b'00000000'
        MOVWF   REGM1
        MOVLW   b'11000000'
        MOVWF   REGM2
        MOVLW   b'11000000'
        MOVWF   REGM3
        MOVLW   b'00000000'
        MOVWF   REGM4
        MOVLW   b'00000000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletSlash
        MOVLW   b'01000000'
        MOVWF   REGM1
        MOVLW   b'00100000'
        MOVWF   REGM2
        MOVLW   b'00010000'
        MOVWF   REGM3
        MOVLW   b'00001000'
        MOVWF   REGM4
        MOVLW   b'00000100'
        MOVWF   REGM5
        GOTO    MletEscrb

Mlet0
        MOVLW   b'01111100'
        MOVWF   REGM1
        MOVLW   b'10100010'
        MOVWF   REGM2
        MOVLW   b'10010010'
        MOVWF   REGM3
        MOVLW   b'10001010'
        MOVWF   REGM4
        MOVLW   b'01111100'
        MOVWF   REGM5
        GOTO    MletEscrb

Mlet1
        MOVLW   b'00000000'
        MOVWF   REGM1
```

```
        MOVLW   b'10000100'
        MOVWF   REGM2
        MOVLW   b'11111110'
        MOVWF   REGM3
        MOVLW   b'10000000'
        MOVWF   REGM4
        MOVLW   b'00000000'
        MOVWF   REGM5
        GOTO    MletEscrb

Mlet2
        MOVLW   b'10000100'
        MOVWF   REGM1
        MOVLW   b'11000010'
        MOVWF   REGM2
        MOVLW   b'10100010'
        MOVWF   REGM3
        MOVLW   b'10010010'
        MOVWF   REGM4
        MOVLW   b'10001100'
        MOVWF   REGM5
        GOTO    MletEscrb

Mlet3
        MOVLW   b'01000010'
        MOVWF   REGM1
        MOVLW   b'10000010'
        MOVWF   REGM2
        MOVLW   b'10001010'
        MOVWF   REGM3
        MOVLW   b'10010110'
        MOVWF   REGM4
        MOVLW   b'01100010'
        MOVWF   REGM5
        GOTO    MletEscrb

Mlet4
        MOVLW   b'00110000'
        MOVWF   REGM1
        MOVLW   b'00101000'
        MOVWF   REGM2
        MOVLW   b'00100100'
        MOVWF   REGM3
        MOVLW   b'11111110'
        MOVWF   REGM4
        MOVLW   b'00100000'
        MOVWF   REGM5
        GOTO    MletEscrb

Mlet5
        MOVLW   b'01001110'
        MOVWF   REGM1
        MOVLW   b'10001010'
        MOVWF   REGM2
        MOVLW   b'10001010'
        MOVWF   REGM3
        MOVLW   b'10001010'
        MOVWF   REGM4
        MOVLW   b'01110010'
        MOVWF   REGM5
        GOTO    MletEscrb

Mlet6
        MOVLW   b'01111000'
        MOVWF   REGM1
```

```
        MOVLW   b'10010100'
        MOVWF   REGM2
        MOVLW   b'10010010'
        MOVWF   REGM3
        MOVLW   b'10010010'
        MOVWF   REGM4
        MOVLW   b'01100000'
        MOVWF   REGM5
        GOTO    MletEscrb

Mlet7
        MOVLW   b'00000010'
        MOVWF   REGM1
        MOVLW   b'11100010'
        MOVWF   REGM2
        MOVLW   b'00010010'
        MOVWF   REGM3
        MOVLW   b'00001010'
        MOVWF   REGM4
        MOVLW   b'00000110'
        MOVWF   REGM5
        GOTO    MletEscrb

Mlet8
        MOVLW   b'01101100'
        MOVWF   REGM1
        MOVLW   b'10010010'
        MOVWF   REGM2
        MOVLW   b'10010010'
        MOVWF   REGM3
        MOVLW   b'10010010'
        MOVWF   REGM4
        MOVLW   b'01101100'
        MOVWF   REGM5
        GOTO    MletEscrb

Mlet9
        MOVLW   b'00001100'
        MOVWF   REGM1
        MOVLW   b'10010010'
        MOVWF   REGM2
        MOVLW   b'10010010'
        MOVWF   REGM3
        MOVLW   b'01010010'
        MOVWF   REGM4
        MOVLW   b'00111100'
        MOVWF   REGM5
        GOTO    MletEscrb

MletDosp
        MOVLW   b'00000000'
        MOVWF   REGM1
        MOVLW   b'01101100'
        MOVWF   REGM2
        MOVLW   b'01101100'
        MOVWF   REGM3
        MOVLW   b'00000000'
        MOVWF   REGM4
        MOVLW   b'00000000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletPcoma
        MOVLW   b'00000000'
        MOVWF   REGM1
```

```
        MOVLW   b'10101100'
        MOVWF   REGM2
        MOVLW   b'01101100'
        MOVWF   REGM3
        MOVLW   b'00000000'
        MOVWF   REGM4
        MOVLW   b'00000000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletMenor
        MOVLW   b'00010000'
        MOVWF   REGM1
        MOVLW   b'00101000'
        MOVWF   REGM2
        MOVLW   b'01000100'
        MOVWF   REGM3
        MOVLW   b'10000010'
        MOVWF   REGM4
        MOVLW   b'00000000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletIgual
        MOVLW   b'00101000'
        MOVWF   REGM1
        MOVLW   b'00101000'
        MOVWF   REGM2
        MOVLW   b'00101000'
        MOVWF   REGM3
        MOVLW   b'00101000'
        MOVWF   REGM4
        MOVLW   b'00101000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletMayor
        MOVLW   b'00000000'
        MOVWF   REGM1
        MOVLW   b'10000010'
        MOVWF   REGM2
        MOVLW   b'01000100'
        MOVWF   REGM3
        MOVLW   b'00101000'
        MOVWF   REGM4
        MOVLW   b'00010000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletPreg
        MOVLW   b'00000100'
        MOVWF   REGM1
        MOVLW   b'00000010'
        MOVWF   REGM2
        MOVLW   b'10100010'
        MOVWF   REGM3
        MOVLW   b'00010010'
        MOVWF   REGM4
        MOVLW   b'00001100'
        MOVWF   REGM5
        GOTO    MletEscrb

MletArrob
        MOVLW   b'01100100'
        MOVWF   REGM1
```

```
        MOVLW   b'10010010'
        MOVWF   REGM2
        MOVLW   b'11110010'
        MOVWF   REGM3
        MOVLW   b'10000010'
        MOVWF   REGM4
        MOVLW   b'01111100'
        MOVWF   REGM5
        GOTO    MletEscrb



; TABLA MAYUSCULAS ********************************************************************************

MletA
        MOVLW   b'11111100'
        MOVWF   REGM1
        MOVLW   b'00100010'
        MOVWF   REGM2
        MOVLW   b'00100010'
        MOVWF   REGM3
        MOVLW   b'00100010'
        MOVWF   REGM4
        MOVLW   b'11111100'
        MOVWF   REGM5
        GOTO    MletEscrb

MletB
        MOVLW   b'11111110'
        MOVWF   REGM1
        MOVLW   b'10010010'
        MOVWF   REGM2
        MOVLW   b'10010010'
        MOVWF   REGM3
        MOVLW   b'10010010'
        MOVWF   REGM4
        MOVLW   b'01101100'
        MOVWF   REGM5
        GOTO    MletEscrb

MletC
        MOVLW   b'01111100'
        MOVWF   REGM1
        MOVLW   b'10000010'
        MOVWF   REGM2
        MOVLW   b'10000010'
        MOVWF   REGM3
        MOVLW   b'10000010'
        MOVWF   REGM4
        MOVLW   b'01000100'
        MOVWF   REGM5
        GOTO    MletEscrb

MletD
        MOVLW   b'11111110'
        MOVWF   REGM1
        MOVLW   b'10000010'
        MOVWF   REGM2
        MOVLW   b'10000010'
        MOVWF   REGM3
        MOVLW   b'01000100'
        MOVWF   REGM4
        MOVLW   b'00111000'
        MOVWF   REGM5
```

```
        GOTO    MletEscrb

MletE
        MOVLW   b'11111110'
        MOVWF   REGM1
        MOVLW   b'10010010'
        MOVWF   REGM2
        MOVLW   b'10010010'
        MOVWF   REGM3
        MOVLW   b'10010010'
        MOVWF   REGM4
        MOVLW   b'10000010'
        MOVWF   REGM5
        GOTO    MletEscrb

MletF
        MOVLW   b'11111110'
        MOVWF   REGM1
        MOVLW   b'00010010'
        MOVWF   REGM2
        MOVLW   b'00010010'
        MOVWF   REGM3
        MOVLW   b'00010010'
        MOVWF   REGM4
        MOVLW   b'00000010'
        MOVWF   REGM5
        GOTO    MletEscrb

MletG
        MOVLW   b'01111100'
        MOVWF   REGM1
        MOVLW   b'10000010'
        MOVWF   REGM2
        MOVLW   b'10010010'
        MOVWF   REGM3
        MOVLW   b'10010010'
        MOVWF   REGM4
        MOVLW   b'11110100'
        MOVWF   REGM5
        GOTO    MletEscrb

MletH
        MOVLW   b'11111110'
        MOVWF   REGM1
        MOVLW   b'00010000'
        MOVWF   REGM2
        MOVLW   b'00010000'
        MOVWF   REGM3
        MOVLW   b'00010000'
        MOVWF   REGM4
        MOVLW   b'11111110'
        MOVWF   REGM5
        GOTO    MletEscrb

MletI
        MOVLW   b'00000000'
        MOVWF   REGM1
        MOVLW   b'10000010'
        MOVWF   REGM2
        MOVLW   b'11111110'
        MOVWF   REGM3
        MOVLW   b'10000010'
        MOVWF   REGM4
        MOVLW   b'00000000'
        MOVWF   REGM5
```

```
        GOTO    MletEscrb

MletJ
        MOVLW   b'01000000'
        MOVWF   REGM1
        MOVLW   b'10000000'
        MOVWF   REGM2
        MOVLW   b'10000010'
        MOVWF   REGM3
        MOVLW   b'01111110'
        MOVWF   REGM4
        MOVLW   b'00000010'
        MOVWF   REGM5
        GOTO    MletEscrb

MletK
        MOVLW   b'11111110'
        MOVWF   REGM1
        MOVLW   b'00010000'
        MOVWF   REGM2
        MOVLW   b'00101000'
        MOVWF   REGM3
        MOVLW   b'01000100'
        MOVWF   REGM4
        MOVLW   b'10000010'
        MOVWF   REGM5
        GOTO    MletEscrb

MletL
        MOVLW   b'11111110'
        MOVWF   REGM1
        MOVLW   b'10000000'
        MOVWF   REGM2
        MOVLW   b'10000000'
        MOVWF   REGM3
        MOVLW   b'10000000'
        MOVWF   REGM4
        MOVLW   b'10000000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletM
        MOVLW   b'11111110'
        MOVWF   REGM1
        MOVLW   b'00000100'
        MOVWF   REGM2
        MOVLW   b'00011000'
        MOVWF   REGM3
        MOVLW   b'00000100'
        MOVWF   REGM4
        MOVLW   b'11111110'
        MOVWF   REGM5
        GOTO    MletEscrb

MletN
        MOVLW   b'11111110'
        MOVWF   REGM1
        MOVLW   b'00001000'
        MOVWF   REGM2
        MOVLW   b'00010000'
        MOVWF   REGM3
        MOVLW   b'00100000'
        MOVWF   REGM4
        MOVLW   b'11111110'
        MOVWF   REGM5
```

```
        GOTO    MletEscrb

MletO
        MOVLW   b'01111100'
        MOVWF   REGM1
        MOVLW   b'10000010'
        MOVWF   REGM2
        MOVLW   b'10000010'
        MOVWF   REGM3
        MOVLW   b'10000010'
        MOVWF   REGM4
        MOVLW   b'01111100'
        MOVWF   REGM5
        GOTO    MletEscrb

MletP
        MOVLW   b'11111110'
        MOVWF   REGM1
        MOVLW   b'00010010'
        MOVWF   REGM2
        MOVLW   b'00010010'
        MOVWF   REGM3
        MOVLW   b'00010010'
        MOVWF   REGM4
        MOVLW   b'00001100'
        MOVWF   REGM5
        GOTO    MletEscrb

MletQ
        MOVLW   b'01111100'
        MOVWF   REGM1
        MOVLW   b'10000010'
        MOVWF   REGM2
        MOVLW   b'10100010'
        MOVWF   REGM3
        MOVLW   b'01000010'
        MOVWF   REGM4
        MOVLW   b'10111100'
        MOVWF   REGM5
        GOTO    MletEscrb

MletR
        MOVLW   b'11111110'
        MOVWF   REGM1
        MOVLW   b'00010010'
        MOVWF   REGM2
        MOVLW   b'00110010'
        MOVWF   REGM3
        MOVLW   b'01010010'
        MOVWF   REGM4
        MOVLW   b'10001100'
        MOVWF   REGM5
        GOTO    MletEscrb

MletS
        MOVLW   b'10001100'
        MOVWF   REGM1
        MOVLW   b'10010010'
        MOVWF   REGM2
        MOVLW   b'10010010'
        MOVWF   REGM3
        MOVLW   b'10010010'
        MOVWF   REGM4
        MOVLW   b'01100010'
        MOVWF   REGM5
```

```
        GOTO    MletEscrb

MletT
        MOVLW   b'00000010'
        MOVWF   REGM1
        MOVLW   b'00000010'
        MOVWF   REGM2
        MOVLW   b'11111110'
        MOVWF   REGM3
        MOVLW   b'00000010'
        MOVWF   REGM4
        MOVLW   b'00000010'
        MOVWF   REGM5
        GOTO    MletEscrb

MletU
        MOVLW   b'01111110'
        MOVWF   REGM1
        MOVLW   b'10000000'
        MOVWF   REGM2
        MOVLW   b'10000000'
        MOVWF   REGM3
        MOVLW   b'10000000'
        MOVWF   REGM4
        MOVLW   b'01111110'
        MOVWF   REGM5
        GOTO    MletEscrb

MletV
        MOVLW   b'00111110'
        MOVWF   REGM1
        MOVLW   b'01000000'
        MOVWF   REGM2
        MOVLW   b'10000000'
        MOVWF   REGM3
        MOVLW   b'01000000'
        MOVWF   REGM4
        MOVLW   b'00111110'
        MOVWF   REGM5
        GOTO    MletEscrb

MletW
        MOVLW   b'01111110'
        MOVWF   REGM1
        MOVLW   b'10000000'
        MOVWF   REGM2
        MOVLW   b'01110000'
        MOVWF   REGM3
        MOVLW   b'10000000'
        MOVWF   REGM4
        MOVLW   b'01111110'
        MOVWF   REGM5
        GOTO    MletEscrb

MletX
        MOVLW   b'11000110'
        MOVWF   REGM1
        MOVLW   b'00101000'
        MOVWF   REGM2
        MOVLW   b'00010000'
        MOVWF   REGM3
        MOVLW   b'00101000'
        MOVWF   REGM4
        MOVLW   b'11000110'
        MOVWF   REGM5
```

```
        GOTO    MletEscrb

MletY
        MOVLW   b'00001110'
        MOVWF   REGM1
        MOVLW   b'00010000'
        MOVWF   REGM2
        MOVLW   b'11110000'
        MOVWF   REGM3
        MOVLW   b'00010000'
        MOVWF   REGM4
        MOVLW   b'00001110'
        MOVWF   REGM5
        GOTO    MletEscrb

MletZ
        MOVLW   b'11000010'
        MOVWF   REGM1
        MOVLW   b'10100010'
        MOVWF   REGM2
        MOVLW   b'10010010'
        MOVWF   REGM3
        MOVLW   b'10001010'
        MOVWF   REGM4
        MOVLW   b'10000110'
        MOVWF   REGM5
        GOTO    MletEscrb

Mletillav
        MOVLW   b'11111110'
        MOVWF   REGM1
        MOVLW   b'10000010'
        MOVWF   REGM2
        MOVLW   b'10000010'
        MOVWF   REGM3
        MOVLW   b'00000000'
        MOVWF   REGM4
        MOVLW   b'00000000'
        MOVWF   REGM5
        GOTO    MletEscrb

Mletbizq
        MOVLW   b'00000100'
        MOVWF   REGM1
        MOVLW   b'00001000'
        MOVWF   REGM2
        MOVLW   b'00010000'
        MOVWF   REGM3
        MOVLW   b'00100000'
        MOVWF   REGM4
        MOVLW   b'01000000'
        MOVWF   REGM5
        GOTO    MletEscrb

Mletfllav
        MOVLW   b'00000000'
        MOVWF   REGM1
        MOVLW   b'00000000'
        MOVWF   REGM2
        MOVLW   b'10000010'
        MOVWF   REGM3
        MOVLW   b'10000010'
        MOVWF   REGM4
        MOVLW   b'11111110'
        MOVWF   REGM5
```

```
        GOTO    MletEscrb

Mletgorro
        MOVLW   b'00001000'
        MOVWF   REGM1
        MOVLW   b'00000100'
        MOVWF   REGM2
        MOVLW   b'00000010'
        MOVWF   REGM3
        MOVLW   b'00000100'
        MOVWF   REGM4
        MOVLW   b'00001000'
        MOVWF   REGM5
        GOTO    MletEscrb

Mletsubrall
        MOVLW   b'10000000'
        MOVWF   REGM1
        MOVLW   b'10000000'
        MOVWF   REGM2
        MOVLW   b'10000000'
        MOVWF   REGM3
        MOVLW   b'10000000'
        MOVWF   REGM4
        MOVLW   b'10000000'
        MOVWF   REGM5
        GOTO    MletEscrb

Mlettildi
        MOVLW   b'00000000'
        MOVWF   REGM1
        MOVLW   b'00000010'
        MOVWF   REGM2
        MOVLW   b'00000100'
        MOVWF   REGM3
        MOVLW   b'00001000'
        MOVWF   REGM4
        MOVLW   b'00000000'
        MOVWF   REGM5
        GOTO    MletEscrb



; TABLA MINUSCULAS ********************************************************************************
MletAm
        MOVLW   b'01000000'
        MOVWF   REGM1
        MOVLW   b'10101000'
        MOVWF   REGM2
        MOVLW   b'10101000'
        MOVWF   REGM3
        MOVLW   b'10101000'
        MOVWF   REGM4
        MOVLW   b'11110000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletBm
        MOVLW   b'11111110'
        MOVWF   REGM1
        MOVLW   b'10010000'
        MOVWF   REGM2
        MOVLW   b'10001000'
        MOVWF   REGM3
        MOVLW   b'10001000'
```

```
        MOVWF   REGM4
        MOVLW   b'01110000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletCm
        MOVLW   b'01110000'
        MOVWF   REGM1
        MOVLW   b'10001000'
        MOVWF   REGM2
        MOVLW   b'10001000'
        MOVWF   REGM3
        MOVLW   b'10001000'
        MOVWF   REGM4
        MOVLW   b'01000000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletDm
        MOVLW   b'01110000'
        MOVWF   REGM1
        MOVLW   b'10001000'
        MOVWF   REGM2
        MOVLW   b'10001000'
        MOVWF   REGM3
        MOVLW   b'10010000'
        MOVWF   REGM4
        MOVLW   b'11111110'
        MOVWF   REGM5
        GOTO    MletEscrb

MletEm
        MOVLW   b'01110000'
        MOVWF   REGM1
        MOVLW   b'10101000'
        MOVWF   REGM2
        MOVLW   b'10101000'
        MOVWF   REGM3
        MOVLW   b'10101000'
        MOVWF   REGM4
        MOVLW   b'00110000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletFm
        MOVLW   b'00010000'
        MOVWF   REGM1
        MOVLW   b'11111100'
        MOVWF   REGM2
        MOVLW   b'00010010'
        MOVWF   REGM3
        MOVLW   b'00000010'
        MOVWF   REGM4
        MOVLW   b'00000100'
        MOVWF   REGM5
        GOTO    MletEscrb

MletGm
        MOVLW   b'00011000'
        MOVWF   REGM1
        MOVLW   b'10100100'
        MOVWF   REGM2
        MOVLW   b'10100100'
        MOVWF   REGM3
        MOVLW   b'10100100'
```

```
        MOVWF   REGM4
        MOVLW   b'01111100'
        MOVWF   REGM5
        GOTO    MletEscrb

MletHm
        MOVLW   b'11111110'
        MOVWF   REGM1
        MOVLW   b'00010000'
        MOVWF   REGM2
        MOVLW   b'00001000'
        MOVWF   REGM3
        MOVLW   b'00001000'
        MOVWF   REGM4
        MOVLW   b'11110000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletIm
        MOVLW   b'00000000'
        MOVWF   REGM1
        MOVLW   b'10001000'
        MOVWF   REGM2
        MOVLW   b'11111010'
        MOVWF   REGM3
        MOVLW   b'10000000'
        MOVWF   REGM4
        MOVLW   b'00000000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletJm
        MOVLW   b'01000000'
        MOVWF   REGM1
        MOVLW   b'10000000'
        MOVWF   REGM2
        MOVLW   b'10001000'
        MOVWF   REGM3
        MOVLW   b'01111010'
        MOVWF   REGM4
        MOVLW   b'00000000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletKm
        MOVLW   b'11111110'
        MOVWF   REGM1
        MOVLW   b'00100000'
        MOVWF   REGM2
        MOVLW   b'01010000'
        MOVWF   REGM3
        MOVLW   b'10001000'
        MOVWF   REGM4
        MOVLW   b'00000000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletLm
        MOVLW   b'00000000'
        MOVWF   REGM1
        MOVLW   b'10000010'
        MOVWF   REGM2
        MOVLW   b'11111110'
        MOVWF   REGM3
        MOVLW   b'10000000'
```

```
        MOVWF   REGM4
        MOVLW   b'00000000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletMm
        MOVLW   b'11111000'
        MOVWF   REGM1
        MOVLW   b'00001000'
        MOVWF   REGM2
        MOVLW   b'00110000'
        MOVWF   REGM3
        MOVLW   b'00001000'
        MOVWF   REGM4
        MOVLW   b'11110000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletNm
        MOVLW   b'11111000'
        MOVWF   REGM1
        MOVLW   b'00010000'
        MOVWF   REGM2
        MOVLW   b'00001000'
        MOVWF   REGM3
        MOVLW   b'00001000'
        MOVWF   REGM4
        MOVLW   b'11110000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletOm
        MOVLW   b'01110000'
        MOVWF   REGM1
        MOVLW   b'10001000'
        MOVWF   REGM2
        MOVLW   b'10001000'
        MOVWF   REGM3
        MOVLW   b'10001000'
        MOVWF   REGM4
        MOVLW   b'01110000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletPm
        MOVLW   b'11111000'
        MOVWF   REGM1
        MOVLW   b'00101000'
        MOVWF   REGM2
        MOVLW   b'00101000'
        MOVWF   REGM3
        MOVLW   b'00101000'
        MOVWF   REGM4
        MOVLW   b'00010000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletQm
        MOVLW   b'00010000'
        MOVWF   REGM1
        MOVLW   b'00101000'
        MOVWF   REGM2
        MOVLW   b'00101000'
        MOVWF   REGM3
        MOVLW   b'00110000'
```

```
        MOVWF   REGM4
        MOVLW   b'11111000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletRm
        MOVLW   b'11111000'
        MOVWF   REGM1
        MOVLW   b'00010000'
        MOVWF   REGM2
        MOVLW   b'00001000'
        MOVWF   REGM3
        MOVLW   b'00001000'
        MOVWF   REGM4
        MOVLW   b'00010000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletSm
        MOVLW   b'10010000'
        MOVWF   REGM1
        MOVLW   b'10101000'
        MOVWF   REGM2
        MOVLW   b'10101000'
        MOVWF   REGM3
        MOVLW   b'10101000'
        MOVWF   REGM4
        MOVLW   b'01000000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletTm
        MOVLW   b'00001000'
        MOVWF   REGM1
        MOVLW   b'01111110'
        MOVWF   REGM2
        MOVLW   b'10001000'
        MOVWF   REGM3
        MOVLW   b'10000000'
        MOVWF   REGM4
        MOVLW   b'01000000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletUm
        MOVLW   b'01111000'
        MOVWF   REGM1
        MOVLW   b'10000000'
        MOVWF   REGM2
        MOVLW   b'10000000'
        MOVWF   REGM3
        MOVLW   b'01000000'
        MOVWF   REGM4
        MOVLW   b'11111000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletVm
        MOVLW   b'00111000'
        MOVWF   REGM1
        MOVLW   b'01000000'
        MOVWF   REGM2
        MOVLW   b'10000000'
        MOVWF   REGM3
        MOVLW   b'01000000'
```

```
        MOVWF   REGM4
        MOVLW   b'00111000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletWm
        MOVLW   b'01111000'
        MOVWF   REGM1
        MOVLW   b'10000000'
        MOVWF   REGM2
        MOVLW   b'01100000'
        MOVWF   REGM3
        MOVLW   b'10000000'
        MOVWF   REGM4
        MOVLW   b'01111000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletXm
        MOVLW   b'10001000'
        MOVWF   REGM1
        MOVLW   b'01010000'
        MOVWF   REGM2
        MOVLW   b'00100000'
        MOVWF   REGM3
        MOVLW   b'01010000'
        MOVWF   REGM4
        MOVLW   b'10001000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletYm
        MOVLW   b'00011000'
        MOVWF   REGM1
        MOVLW   b'10100000'
        MOVWF   REGM2
        MOVLW   b'10100000'
        MOVWF   REGM3
        MOVLW   b'10100000'
        MOVWF   REGM4
        MOVLW   b'01111000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletZm
        MOVLW   b'10001000'
        MOVWF   REGM1
        MOVLW   b'11001000'
        MOVWF   REGM2
        MOVLW   b'10101000'
        MOVWF   REGM3
        MOVLW   b'10011000'
        MOVWF   REGM4
        MOVLW   b'10001000'
        MOVWF   REGM5
        GOTO    MletEscrb

Mleticorch
        MOVLW   b'00000000'
        MOVWF   REGM1
        MOVLW   b'00010000'
        MOVWF   REGM2
        MOVLW   b'01101100'
        MOVWF   REGM3
        MOVLW   b'10000010'
```

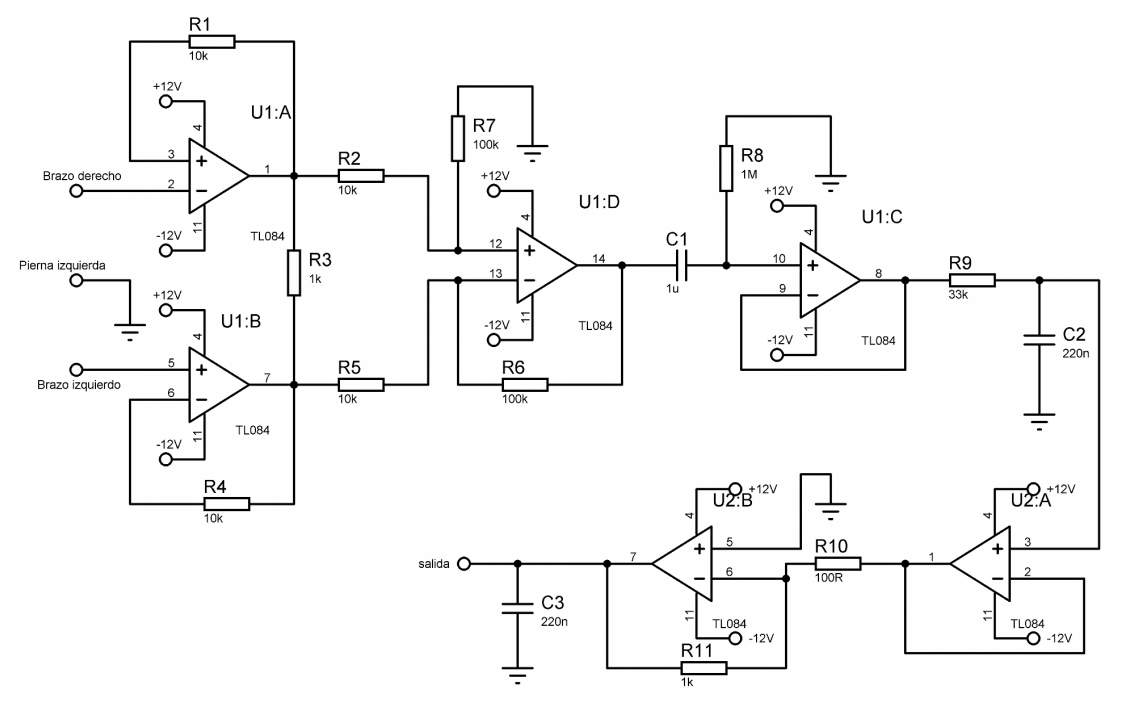
```
        MOVWF   REGM4
        MOVLW   b'00000000'
        MOVWF   REGM5
        GOTO    MletEscrb

Mlettuberia
        MOVLW   b'00000000'
        MOVWF   REGM1
        MOVLW   b'00000000'
        MOVWF   REGM2
        MOVLW   b'11111110'
        MOVWF   REGM3
        MOVLW   b'00000000'
        MOVWF   REGM4
        MOVLW   b'00000000'
        MOVWF   REGM5
        GOTO    MletEscrb

Mletfcorch
        MOVLW   b'00000000'
        MOVWF   REGM1
        MOVLW   b'10000010'
        MOVWF   REGM2
        MOVLW   b'01101100'
        MOVWF   REGM3
        MOVLW   b'00010000'
        MOVWF   REGM4
        MOVLW   b'00000000'
        MOVWF   REGM5
        GOTO    MletEscrb

Mletvirgu
        MOVLW   b'00010000'
        MOVWF   REGM1
        MOVLW   b'00001000'
        MOVWF   REGM2
        MOVLW   b'00001000'
        MOVWF   REGM3
        MOVLW   b'00010000'
        MOVWF   REGM4
        MOVLW   b'00001000'
        MOVWF   REGM5
        GOTO    MletEscrb


; TABLA CARACTERES ESPECIALES ********************************************************************
MletAmt
        MOVLW   b'01000000'
        MOVWF   REGM1
        MOVLW   b'10101000'
        MOVWF   REGM2
        MOVLW   b'10101010'
        MOVWF   REGM3
        MOVLW   b'10101001'
        MOVWF   REGM4
        MOVLW   b'11110000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletEmt
        MOVLW   b'01110000'
        MOVWF   REGM1
        MOVLW   b'10101000'
        MOVWF   REGM2
        MOVLW   b'10101010'
```

```
        MOVWF   REGM3
        MOVLW   b'10101001'
        MOVWF   REGM4
        MOVLW   b'00110000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletImt
        MOVLW   b'00000000'
        MOVWF   REGM1
        MOVLW   b'10001000'
        MOVWF   REGM2
        MOVLW   b'11111010'
        MOVWF   REGM3
        MOVLW   b'10000001'
        MOVWF   REGM4
        MOVLW   b'00000000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletOmt
        MOVLW   b'01110000'
        MOVWF   REGM1
        MOVLW   b'10001000'
        MOVWF   REGM2
        MOVLW   b'10001010'
        MOVWF   REGM3
        MOVLW   b'10001001'
        MOVWF   REGM4
        MOVLW   b'01110000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletUmt
        MOVLW   b'01111000'
        MOVWF   REGM1
        MOVLW   b'10000000'
        MOVWF   REGM2
        MOVLW   b'10000010'
        MOVWF   REGM3
        MOVLW   b'01000001'
        MOVWF   REGM4
        MOVLW   b'11111000'
        MOVWF   REGM5
        GOTO    MletEscrb

Mletñmt
        MOVLW   b'11111000'
        MOVWF   REGM1
        MOVLW   b'00010010'
        MOVWF   REGM2
        MOVLW   b'00001010'
        MOVWF   REGM3
        MOVLW   b'00001010'
        MOVWF   REGM4
        MOVLW   b'11110000'
        MOVWF   REGM5
        GOTO    MletEscrb

MletUdir
        MOVLW   b'01111010'
        MOVWF   REGM1
        MOVLW   b'10000000'
        MOVWF   REGM2
        MOVLW   b'10000000'
```

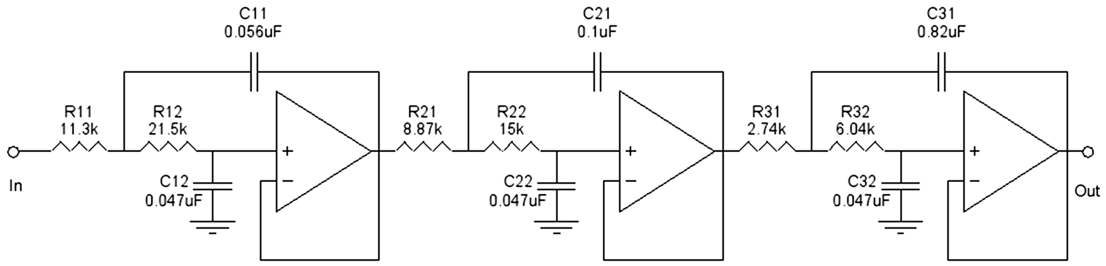
```
        MOVWF   REGM3
        MOVLW   b'01000000'
        MOVWF   REGM4
        MOVLW   b'11111010'
        MOVWF   REGM5
        GOTO    MletEscrb


; FUNCIÓN DE ESCRITURA ********************************************************************************
; Esta función escribe los datos de los registros REGM1, REGM2, REGM3, REGM4 y REGM5 en el puerto
; de datos para que sean visualizados en la matriz de leds.
MletEscrb
        CLRF    MDLDATA                 ; Manda una columna en blanco que representa el espacio
                                        ; entre los caracteres
        CALL    MDLRET                  ; Llama al retardo de las matriz
        BSF     MDLPCLK                 ; Manda un pulso al reloj para producir un desplazamiento
        CALL    Retardo_10micros        ; en la matriz
        BCF     MDLPCLK
        MOVF    REGM1,W                 ; Manda el REGM1 al puerto de datos
        MOVWF   MDLDATA
        CALL    MDLRET
        BSF     MDLPCLK
        CALL    Retardo_10micros
        BCF     MDLPCLK
        MOVF    REGM2,W                 ; Manda el REGM2 al puerto de datos
        MOVWF   MDLDATA
        CALL    MDLRET
        BSF     MDLPCLK
        CALL    Retardo_10micros
        BCF     MDLPCLK
        MOVF    REGM3,W                 ; Manda el REGM3 al puerto de datos
        MOVWF   MDLDATA
        CALL    MDLRET
        BSF     MDLPCLK
        CALL    Retardo_10micros
        BCF     MDLPCLK
        MOVF    REGM4,W                 ; Manda el REGM5 al puerto de datos
        MOVWF   MDLDATA
        CALL    MDLRET
        BSF     MDLPCLK
        CALL    Retardo_10micros
        BCF     MDLPCLK
        MOVF    REGM5,W                 ; Manda el REGM5 al puerto de datos
        MOVWF   MDLDATA
        CALL    MDLRET
        BSF     MDLPCLK
        CALL    Retardo_10micros
        BCF     MDLPCLK
        RETURN                          ; Termina la función de escritura

; FUNCIÓN DE RETARDO ***********************************************************************************
; Esta función genera un retardo que depende del valor leído en la entrada análoga del micro.
MDLRET
        CALL    LEERANALOG              ; Lee la entrada análoga
        MOVWF   MDLRETC                 ; Mueve el valor leído a MDLRETC el cual es un contador
        BCF     STATUS,C                ; Divide el valor leído entre dos por lo cual lo rota
        RRF     MDLRETC,F               ; hacia la derecha
        INCF    MDLRETC,F               ; Incrementa en 1 el valor MDLRETC para que 1 sea el
                                        ; mínimo valor
MDLRET1
        CALL    Retardo_1ms             ; Llama a un retardo de 1ms
        DECFSZ  MDLRETC,F               ; Decrementa el contador y sale del ciclo cuando este
        GOTO    MDLRET1                 ; llega a cero
        RETURN                          ; Finaliza la función de retardo
```

```
; FUNCIÓN DE LECTURA ENTRADA ANALOGA ************************************************************
; Esta función lee la entrada análoga y guara el valor leído en el registro de trabajo W
LEERANALOG
        BSF     ADCON0,2                ; Comienza a leer entrada
LEERANALOG1
        BTFSC   ADCON0,2                ; Espera a que la lectura halla terminado
        GOTO    LEERANALOG1
        MOVF    ADRESH,W                ; Si la lectura ha terminado mueve el dato leído del
        RETURN                          ; registro ADRESH al registro de trabajo W.

; FIN DE LA LIBRERIA ****************************************************************
```

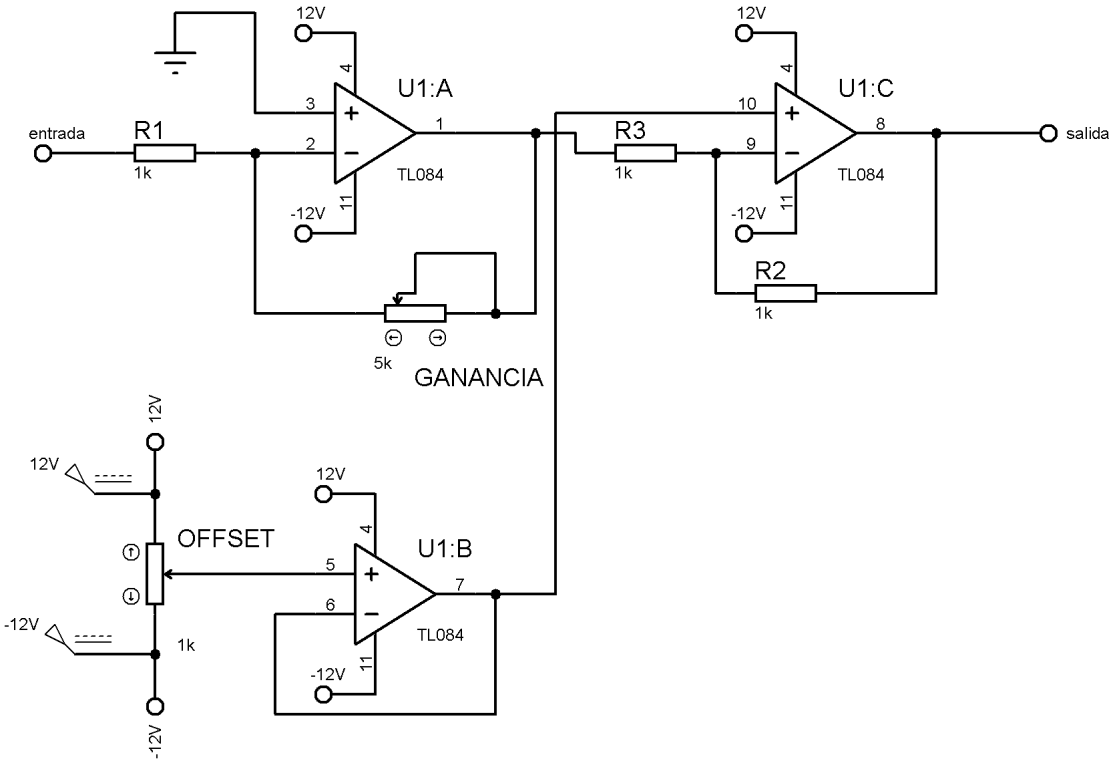
ANEXO E. Etapa de amplificación ECG



ANEXO F. Filtro antialias ECG



ANEXO G. Ganancia y offset ECG



ANEXO H. Código electrocardiógrafo

```
;************************************ MDL-MUSB.ASM ***************************************
; Este programa graba de forma digital una señal análoga de ECG entre 0 a 5v con un muestreo de
; 400Hz a una resolución de 10bits en un archivo ECG_XX.DAT donde XX es el número del archivo
; que se puede seleccionar mediante dos displays de 7 segmentos y dos bonotes de incrementar y
; decrementar.
;
; Al iniciar el programa, este le pide al usuario que escoja en que archivo desea guardar los datos,
; el numero del archivo es indicado por dos display de 7 segmentos, este numero se puede cambiar
; con los botones de incrementar y decrementar. Un LED denominado archivo libre indica que no existe
; un archivo con ese numero y que el lugar esta libre para escribir uno nuevo. Un LED denominado
; archivo ocupado indica que si existe un archivo con ese nombre y que no se puede escribir en el.
; Uno de estos LEDS se enciende cada vez que el usuario se posiciona en un archivo.
;
; Después de escoger el archivo se pulsa el botón grabar/finalizar para entrar en una etapa de
; calibración de la señal. Se tienen 4 LEDS, dos rojos exteriores de limites alto y bajo y dos
; verdes interiores de umbral alto y bajo. Para calibrar la entrada hay que ajustar los
; potenciómetros de offset y el de ganancia de tal forma que parpadeen solo los LEDS verdes de
; umbral, si parpadean los LEDS rojos de limite alto o bajo quiere decir que se esta acercando
; mucho a los limites de lectura del digitalizador.
;
; Una vez realizada la calibración se pulsa el botón grabar/finalizar para iniciar con la grabación,
; en este momento empiezan a guardar los datos en la memoria USB en el archivo seleccionado, se puede
; seguir observando los LEDS de calibración para ver como se comparta la entrada. Para finalizar la
; grabación se pulsa el botón grabar/finalizar, los display de 7 segmentos muestran un doble cero
; indicando que el proceso a terminado. En este momento se puede extraer la memoria de forma segura,
; se puede cambiarla por otra y/o se puede pulsar el botón grabar/finalizar para grabar en un
; diferente archivo.
;
;     ================================================================================
;     ||                              PROYECTO DE GRADO                             ||
;     ||                                                                            ||
;     ||     ESTUDIO Y DISEÑO DE ALGORITMOS E INTERFAZ ELECTRÓNICA DE COMUNICACIÓN  ||
;     ||    ENTRE MICROCONTROLADORES SERIE PIC Y DISPOSITIVOS DE ALMACENAMIENTO USB ||
;     ||                                                                            ||
;     ||                           Juan Pablo Ruiz Rosero                           ||
;     ||                           Ingeniería Electrónica                           ||
;     ||                            Universidad de Nariño                           ||
;     ||                                Octubre 2008                                ||
;     ================================================================================
;
;*******************************************************************************************

; ZONA DE DATOS ********************************************************************

       list       P=16F877A                   ; Procesador utilizado.
       #include  <P16F877A.INC>                ; En este fichero se definen las etiquetas del PIC.

       __CONFIG _CP_OFF & _WDT_OFF & _BODEN_ON & _PWRTE_ON & _XT_OSC & _LVP_OFF & _CPD_ON
       ; Configuración del microcontrolador


       CBLOCK  20h             ; Definición de dirección para las variables
       CONTADOR
       CONTRC
       TEMPW
       TEMPSTATUS
       ARCHDEC
       ARCHUNI
       NARCHIVO
       CONTBT
```

```
        ENDC

#DEFINE BTINC   PORTC,4         ; Define pin para el botón incrementar contador
#DEFINE BTDEC   PORTC,5         ; Define pin para el botón decrementar contador
#DEFINE BTGBFN  PORTC,6         ; Define pin para el botón grabar/finalizar
#DEFINE LEDLIB  PORTD,0         ; Define pin para el LED archivo libre
#DEFINE LEDOCP  PORTD,1         ; Define pin para el LED archivo ocupado
#DEFINE LEDULM  PORTD,4         ; Define pin para el LED limite alto
#DEFINE LEDUUM  PORTD,5         ; Define pin para el LED umbral alto
#DEFINE LEDDUM  PORTD,6         ; Define pin para el LED umbral bajo
#DEFINE LEDDLM  PORTD,7         ; Define pin para el LED limite bajo



; ZONA DE CODIGO ******************************************************************************

        ORG     0
        GOTO    INICIO                  ; Va al inicio del programa
        ORG     4                       ; La interrupción salta a la posición 4 del programa
        GOTO    INTLEER                 ; Va a la parte del código de interrupción

; INICIALIZACION ******************************************************************************
INICIO
        BSF     STATUS,RP0              ; Pone en 1 el bit 5 del STATUS. Acceso al Banco 1.
        BSF     BTINC                   ; Define los botones como entradas
        BSF     BTDEC
        BSF     BTGBFN
        BCF     LEDLIB                  ; Define los LEDS como salidas
        BCF     LEDOCP
        BCF     LEDULM
        BCF     LEDUUM
        BCF     LEDDUM
        BCF     LEDDLM
        MOVLW   b'11000101'             ; Define al TMR0 con fuente interna, preescaler 1:64
        MOVWF   OPTION_REG
        MOVLW   b'00001110'             ; Conversor A/D con justificación derecha, solo RA0
        MOVWF   ADCON1                  ; como entrada y Vref+ Vcc y  Vref- GND
        CLRF    PORTB                   ; Coloca en cero al PUERTOB donde están los displays 7seg
        BCF     STATUS,RP0              ; Acceso al banco 0
        MOVLW   b'10000001'             ; Conversor A/D con Fosc/32 para un reloj de 20Mhz
        MOVWF   ADCON0                  ; entrada por RA0 chan0 yadon 1 encendido
        BCF     LEDLIB                  ; Inicializa todos los LEDS en cero
        BCF     LEDOCP
        BCF     LEDULM
        BCF     LEDUUM
        BCF     LEDDUM
        BCF     LEDDLM

        CALL    VDINIC                  ; Inicializa la comunicación con el VDRIVE1
        CALL    VDIPHM                  ; Configura al VDRIVE1 en modo datos binario
        MOVLW   d'1'                    ; Coloca a NARCHIVO en 1
        MOVWF   NARCHIVO

; RUTINA PRINCIPAL ****************************************************************************

PRINCIPAL
        CALL    ESCGARCH                ; Llama la función que escoge el archivo a escribir
        CALL    CALBECG                 ; Llama la función que ayuda a calibrar la entrada

        CALL    ABRIRARCHIVOE           ; Abre el archivo seleccionado para su escritura
        CALL    ESCR512B                ; Da la orden para escribir 512 bytes
        MOVLW   0x00                    ; Inicializa el contador en 0x00 para contar 256 veces
        MOVWF   CONTADOR
```

```
        CLRF    TMR0
        MOVLW   b'10100000'             ; Activa las interrupciones solo con TMR0
        MOVWF   INTCON
        MOVLW   -d'195'                 ; Coloca al TMR0 en 195 para realizar interrupciones
        MOVWF   TMR0                    ; cada 2.5ms para un muestreo de 400Hz

BLUCE
        NOP
        GOTO    BLUCE



; INTERRUPCION    ********************************************************************************
; Esta rutina sera llamada automáticamente cada 2.5ms con la ayuda de una interrupción causada
; por el timer interno

INTLEER
        MOVWF   TEMPW                   ; Guarda los datos de W en TEMPW
        SWAPF   STATUS,W
        MOVWF   TEMPSTATUS              ; Guarda los datos de STATUS en TEMPSTATUS
        NOP                             ; Genera retardos para calibrar el retardo en exactamente
        MOVLW   d'1'                    ; 2.5ms
        CALL    RETARDOC


        MOVLW   -d'195'                 ; Coloca al TMR0 en 195 para realizar interrupciones
        MOVWF   TMR0                    ; cada 2.5ms para un muestreo de 400Hz

        CALL    LEERANALOG              ; Lee la entrada análoga
        CALL    COMPLEDS                ; Llama la función que activa los LEDS de comparación
        MOVF    ADRESH,W                ; Coloca los 8bits mas significativos en W
        CALL    VDSPIW                  ; Guarda este dato en la memoria USB
        BSF     STATUS,RP0              ; Pone a 1 el bit 5 del STATUS. Acceso al Banco 1.
        MOVF    ADRESL,W                ; Coloca los bits menos significativos en W
        BCF     STATUS,RP0              ; Pone a 0 el bit 5 del STATUS. Acceso al Banco 0.
        CALL    VDSPIW                  ; Guarda este dato en la memoria USB
        DECFSZ  CONTADOR,F              ; Decrementa el contador de bytes escritos
        GOTO    FININTLEER              ; Salta a FININTLEER donde finaliza la interrupción
        BTFSS   BTGBFN                  ; Si el contador llegó a cero testea el botón de grabar
                                        ; finalizar para saber si el usuario a finalizado la operación
        GOTO    FINLECTURA              ; Si el usuario ha finalizado la operación salta a FINLECUTRA

        CALL    ESCR512B                ; Si el usuario no ha finalizado manda la orden al VDRIVE1
        MOVLW   0x00                    ; para escribir nuevamente 512 bytes y coloca el contador
        MOVWF   CONTADOR                ; en 0x00


FININTLEER                              ; Fin de la interrupción de lectura
        SWAPF   TEMPSTATUS,W            ; Coloca el valor de TEMPSTATUS en el registro de STATUS
        MOVWF   STATUS
        SWAPF   TEMPW,F                 ; Coloca el valor de TEMPW en el registro W
        SWAPF   TEMPW,W
        BCF     INTCON,TOIF             ; Activa la interrupción para que pueda ser llamada de nuevo
        BCF     INTCON,INTF
        RETFIE                          ; Retorna de la interrupción


FINLECTURA                              ; Fin de la escritura del archivo
        CALL    CERRARARCHIVO           ; Llama a la función para cerrar el archivo
        CALL    CERRARARCHIVO
        CLRF    PORTB                   ; Coloca en ceros al puerto B para indicar que ha terminado
        CALL    VDSUD                   ; Coloca en suspensión a la memoria USB para poder cambiarla
                                        ; o sacarla
```

```
        CALL    WBTGBFN                 ; Espera a que se pulse el botón grabar/finalizar
        GOTO    PRINCIPAL               ; Salta a principal para grabar un nuevo archivo


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

; ESPERA A QUE SE PULSE EL BOTON ***********************************************************
WBTGBFN
        BTFSC   BTGBFN                  ; Espera que el botón sea pulsado
        GOTO    WBTGBFN
        CALL    Retardo_50ms            ; Si el botón fue pulsado espera 10ms para que no exista
                                        ; doble pulsación por ruido
WBTGBFN2
        BTFSS   BTGBFN                  ; Espera a que el botón deje de estar pulsado
        GOTO    WBTGBFN2
        CALL    Retardo_50ms            ; Si el botón dejo de estar pulsado espera 10ms para que
                                        ; no exista doble pulsación por ruido
        RETURN                          ; Finaliza la función


; LEER ENTRADA ANALOGA *********************************************************************
LEERANALOG
        BSF     ADCON0,2                ; Activa el inicio de la lectura de la entrada análoga
LEERANALOG1
        BTFSC   ADCON0,2                ; Espera a que la lectura haya finalizado
        GOTO    LEERANALOG1
        RETURN                          ; Termina la función

; RETARDO CICLOS ***************************************************************************
RETARDOC
        MOVWF   CONTRC                  ; Mueve el valor de W al contador CONTRC
RETARDOC1
        DECFSZ  CONTRC,F                ; Decrementa el contador y salta cuando llegue a cero
        GOTO    RETARDOC1
        RETURN                          ; Si el contador llega a cero termina la función


; FUNCIÓN CALIBRAR ENTRADA ECG *************************************************************
; Esta función lee la entrada análoga, compara con valores limites y de umbral y saca el resultado
; por 4 LEDS.
CALBECG
        CALL    LEERANALOG              ; Lee la entrada análoga
        CALL    COMPLEDS                ; Llama a la función comparar LEDS
        BTFSC   BTGBFN                  ; Testea si el botón grabar/finalizar es pulsado
        GOTO    CALBECG                 ; Si no es pulsado salta a CALBECG
        CALL    WBTGBFN                 ; Si es pulsado llama a WBTGBFN para esperar la pulsación
        RETURN                          ; Termina la función

; FUNCIÓN COMPARA LEDS *********************************************************************
; Esta función compara el dato de la entrada análoga con 4 valores.
COMPLEDS
        MOVLW   d'245'                  ; Compara si la entrada análoga es mayor a 245
        SUBWF   ADRESH,W
        BCF     LEDULM                  ; Supone que no es mayor y apaga el LED de limite alto
        BTFSC   STATUS,C
        BSF     LEDULM                  ; Si es mayor enciende el LED de limite alto

        MOVLW   d'215'                  ; Compara si la entrada análoga es mayor a 215
        SUBWF   ADRESH,W
        BCF     LEDUUM                  ; Supone que no es mayor y apaga el LED de umbral alto
        BTFSC   STATUS,C
        BSF     LEDUUM                  ; Si es mayor enciende el LED de umbral alto

        MOVLW   d'40'                   ; Compara si la entrada análoga es menor a 40
        SUBWF   ADRESH,W
```

```
        BCF     LEDDUM                  ; Supone que no es menor y apaga el LED de umbral bajo
        BTFSS   STATUS,C
        BSF     LEDDUM                  ; Si es menor enciende el LED de umbral bajo

        MOVLW   d'10'                   ; Compara si la entrada análoga es menor a 10
        SUBWF   ADRESH,W
        BCF     LEDDLM                  ; Supone que no es menor y apaga el LED de limite bajo
        BTFSS   STATUS,C
        BSF     LEDDLM                  ; Si es menor enciende el LED de limite bajo
        RETURN

; FUNCIÓN ESCOGER ARCHIVO ********************************************************************
; Esta función permite al usuario escoger el archivo en el cual guardara los datos de la señal ECG
ESCGARCH
        CALL    TESTBTINC               ; Testea el botón incrementar
        CALL    TESTBTDEC               ; Testea el botón decrementar
        MOVF    NARCHIVO,W              ; Coloca la variable del numero del archivo NARCHIVO en W
        CALL    BINAD99                 ; La convierte a decimal
        MOVWF   PORTB                   ; La saca por el puerto B
        CALL    DISPARCH                ; Llama a la función de disponibilidad de archivo

ESCGARCH2
        BTFSS   BTINC                   ; Detecta si es pulsado el botón incrementar o el botón
        GOTO    ESCGARCH                ; decrementar, si es pulsado alguno salta a ESCGARCH si
        BTFSS   BTDEC                   ; no queda en el ciclo de ESCGARCH2 esperando
        GOTO    ESCGARCH
        BTFSC   BTGBFN                  ; Detecta si es pulsado el botón grabar/finalizar
        GOTO    ESCGARCH2               ; Si no es pulsado continua en el ciclo ESCGARCH2
        BTFSC   LEDOCP                  ; Testea el LED de archivo ocupado
        GOTO    ESCGARCH2               ; Si el archivo esta ocupado vuelve a a ESCARCH2
        CALL    WBTGBFN                 ; Si el archivo no esta llama a WBTGBFN para esperar la pulsación
        RETURN                          ; Termina la función


; FUNCIÓN DISPONIBILIDAD ARCHIVO *************************************************************
; Esta función indica si un archivo existe o no existe en la memoria USB
DISPARCH
        BCF     LEDLIB                  ; Coloca en cero los LEDS de archivo ocupado o disponible
        BCF     LEDOCP
        MOVF    NARCHIVO,W              ; Convierte la variable NARCHIVO en cadena y se guarda
        CALL    BINAUNIC                ; respectivamente las unidades en ARCHUNI y las decenas
        MOVWF   ARCHUNI                 ; en ARCHDEC
        MOVF    NARCHIVO,W
        CALL    BINADECC
        MOVWF   ARCHDEC
        CALL    DIRARCHIVO              ; Aplica el comando DIR sobre el archivo
        CALL    VDSPIRN                 ; Lee la primera respuesta que debe ser un 0x0D
        CALL    VDSPIRN                 ; Lee la segunda respuesta, si esta es igual a 'E' el
        SUBLW   'E'                     ; archivo existe ya que este empieza por 'E' si no es
        BTFSS   STATUS,Z                ; igual a 'E' el archivo no existe
        GOTO    DISPARCH1               ; Si no es igual a 'E' salta a DISPARCH1
        BSF     LEDOCP                  ; Si es igual a 'E' enciende el LED de archivo ocupado
        CALL    VDWNM                   ; Espera a que el VDRIVE1 mande el resto de datos
        RETURN                          ; Termina la función
DISPARCH1
        BSF     LEDLIB                  ; Si no es igual a 'E' enciende el LED de archivo libre
        CALL    VDWWNM                  ; Espera a que el VDRIVE1 mande el resto de datos
        RETURN                          ; Termina la función


; FUNCIÓN TESTEAR BOTON INCREMENTAR **********************************************************
; Testea si se pulsó el botón incrementar y comprueba que no se pase de 99
TESTBTINC
        BTFSC   BTINC                   ; Testea si el botón incrementar esta pulsado
        RETURN                          ; Si no esta pulsado retorna
```

```
        INCF    NARCHIVO,F              ; Si esta pulsado incrementa NARCHIVO
        MOVF    NARCHIVO,W
        SUBLW   d'100'                  ; Mira si NARCHIVO es igual a 100
        MOVLW   d'1'
        BTFSC   STATUS,Z
        MOVWF   NARCHIVO                ; Si es igual a 100 coloca a NARCHIVO en 1
        CALL    Retardo_50ms            ; Retardo de 10ms para evitar pulsaciones falsas
TESTBTINC1
        BTFSS   BTINC                   ; Espera a que se deje de pulsar el botón
        GOTO    TESTBTINC1
        RETURN                          ; Termina la función


; FUNCIÓN TESTEAR BOTON DECREMENTAR ***************************************************************
; Testea si se pulsó el botón decrementar y comprueba que no se pase de 1
TESTBTDEC
        BTFSC   BTDEC                   ; Testea si el botón incrementar esta pulsado
        RETURN                          ; Si no esta pulsado retorna
        DECF    NARCHIVO,F              ; Si esta pulsado decrementa NARCHIVO
        MOVLW   d'99'
        BTFSC   STATUS,Z                ; Mira si NARHCIVO es igual a cero
        MOVWF   NARCHIVO                ; Si es igual a cero coloca a NARCHIVO en 99
        CALL    Retardo_50ms            ; Retardo de 10ms para evitar pulsaciones falsas
TESTBTDEC1
        BTFSS   BTDEC                   ; Espera a que se deje de pulsar el botón
        GOTO    TESTBTDEC1
        RETURN                          ; Termina la función



; FUNCIÓN ESCRIBIR EL NOMBRE DEL ARCHIVO**********************************************************
; Esta función escribe el nombre del archivo en el VDRIVE1
WNARCHIVO
        MOVLW   'E'
        CALL    VDSPIW
        MOVLW   'C'
        CALL    VDSPIW
        MOVLW   'G'
        CALL    VDSPIW
        MOVLW   '_'
        CALL    VDSPIW
        MOVF    ARCHDEC,W
        CALL    VDSPIW
        MOVF    ARCHUNI,W
        CALL    VDSPIW
        MOVLW   '.'
        CALL    VDSPIW
        MOVLW   'D'
        CALL    VDSPIW
        MOVLW   'A'
        CALL    VDSPIW
        MOVLW   'T'
        CALL    VDSPIW
        RETURN

; FUNCIÓN ABRIR ARCHIVO PARA ESCRITURA ************************************************************
; Abre el archivo para su escritura
ABRIRARCHIVOE
        CALL    VDOPW                   ; Llama la función OPW
        CALL    WNARCHIVO               ; Escribe el nombre del archivo en el VDRIVE1
        MOVLW   0x0D                    ; Manda un retorno de carril
        CALL    VDSPIW
        RETURN
```

```
; FUNCIÓN DIR SOBRE ARCHIVO ******************************************************************************
; Aplica un DIR sobre el archivo para saber si este existe
DIRARCHIVO
        CALL    VDDIRF                  ; Llama a la función DIRF
        CALL    WNARCHIVO               ; Escribe el nombre del archivo en el VDRIVE1
        MOVLW   0x0D                    ; Manda un retorno de carril
        CALL    VDSPIW
        RETURN

; FUNCIÓN ESCRIBIR 512 BYTES *****************************************************************************
; Da la orden al VDRIVE1 para que escriba 512 bytes en la memoria USB, 512=0x00000200
ESCR512B
        CALL    VDWRF
        MOVLW   0x00
        CALL    VDSPIW
        MOVLW   0x00
        CALL    VDSPIW
        MOVLW   0x02
        CALL    VDSPIW
        MOVLW   0x00
        CALL    VDSPIW
        MOVLW   0x0D
        CALL    VDSPIW
        RETURN

; FUNCIÓN CERRAR ARCHIVO *********************************************************************************
; Esta función cierra el archivo en el que se estaba escribiendo
CERRARARCHIVO
        CALL    VDCLF                   ; Llama la función VDCLF
        CALL    WNARCHIVO               ; Escribe el nombre del archivo en el VDRIVE1
        MOVLW   0x0D                    ; Manda un retorno de carril
        CALL    VDSPIW
        RETURN

; ZONA DE LIBRERIAS **************************************************************************************

        INCLUDE "D:\Electronica\Tesis\Librerias\RETARDOS.INC"
        INCLUDE "D:\Electronica\Tesis\Librerias\VDRIVE.INC"
        INCLUDE "D:\Electronica\Tesis\Librerias\BINAD99.INC"

; FIN DEL CODIGO *****************************************************************************************
        END



;       ================================================================
```