ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"Adquisición de datos y control de temperatura para ensayo de carga constante en polímeros"

TESIS DE GRADO:

Previo a la obtención del Título de:

INGENIERO EN ELÉCTRICIDAD ESPECIALIZACIÓN "ELECTRONICA Y AUTOMATIZACIÓN INDUSTRIAL"

Presentado por:

Alex Barcos Sinche

Carlos Castro Mendoza

GUAYAQUIL – ECUADOR

Año: 2008

A G R A D E C I M I E N T O

A la ESPOL por la formación académica recibida, A nuestros profesores por el conocimiento impartido durante nuestros años de estudio, A nuestros compañeros por ser ejemplo de solidaridad y compañerismo.

Alex Barcos S.

A la ESPOL por abrirme sus puertas y a los profesores por toda la enseñanza académica como también por haber compartido sus experiencias con nosotros.

Carlos Castro M.

D E D I C A T O R I A

A mi madre por ser ejemplo de virtud y dedicación, A mi padre ejemplo de trabajo incansable, a mis hermanos la alegría de mi vida.

Alex Barcos S.

Esta tesis esta dedicada a mi madre, padre y hermanos por siempre estar junto a mí y darme su apoyo incondicionalmente.

Carlos Castro M.

TRIBUNAL DE GRADUACION

Ing. Holger Cevallos	Ing. Hugo Villavicencio V.
SUBDECANO DE LA FIEC	DIRECTOR DE TESIS
PRESIDENTE	

Ing. Alberto Manzur H.	Ing. Carlos Valdiviezo A.
MIEMBRO PRINCIPAL	MIEMBRO PRINCIPAL

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de este proyecto de graduación nos corresponden exclusivamente, y el patrimonio intelectual de la misma a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL"

(Reglamento de Graduación de la ESPOL)

Alex Barcos S.

Carlos Castro M.

INDICE GENERAL

ANEXOS

INDICE DE FIGURAS

CAPITULO 1

1 ANTECEDENTES

1.1 Objetivos del proyecto

1.1.1 Objetivo general

Desarrollar aplicaciones en donde se requiera el intercambio de información entre una aplicación desarrollada en Visual Basic y un dispositivo externo al ordenador.

Conocer el puerto serial USB (Universal Serial Bus), sus características, aplicaciones, ventajas y desventajas para la adquisición de datos y control de dispositivos en el área industrial.

Profundizar en el mundo de los microcontroladores de la familia PIC18Fxxxx de Microchip, los mismos que en la actualidad son una herramienta poderosa en el diseño electrónico de diversas áreas.

Ampliar conocimientos en la programación orientadas a eventos tal como en Visual Basic, y un reforzamiento en los conocimientos adquiridos en cursos básicos de lenguaje de alto nivel como el C y que fueron asimilados en el transcurso de nuestra carrera.

1.1.2 Objetivos específicos

Medir la deformación de una muestra de plástico sometida a carga constante.

Almacenar los datos en el circuito para que estos sean transmitidos al computador al final de la prueba.

Desarrollar una interfaz gráfica amigable y segura para que el usuario tenga la facilidad de descargar toda la información adquirida durante el ensayo.

Implementar un control de lazo cerrado para asegurar que la prueba se la realiza a una determinada temperatura, la misma que será definida por el usuario.

Garantizar la adquisición y almacenamiento de datos durante fallas del suministro de energía eléctrica.

1.2 Identificación de la problemática.

En la actualidad existen varios tipos máquinas para realizar el ensayo de tensión constante, las máquinas en su forma mas simple realizan la tarea de mantener una muestra de polímetro a carga constante y poseen algún tipo de visualizador para que el operario tome apunte cada cierto período de tiempo de cuanto ha sido la elongación del espécimen, fecha y hora, y en otros casos la temperatura del ambiente que se encuentra la muestra.

La máquina de ensayo Creep descrita para este proyecto tiene dos modificaciones, la primera no posee tecnología suficiente en la adquisición de los datos, y la segunda opción posee mas recursos de los que necesita, haciéndolo mas costoso.

1.2.1 Tecnología utilizada.

Una de las máquinas para realizar el ensayo de tensión constante realiza la adquisición de los datos a través de un indicador analógico, el cual es supervisado cada cierto tiempo por un operario, el mismo que toma nota de los valores que se presentan en los indicadores del estiramiento y temperatura del ambiente de la muestra.

Existe otra forma menos manual para este trabajo pero más costosa que es la obtención de los parámetros por un software denominado Labview. Para realizar la tarea ya citada, el sistema necesita una tarjeta adicional de adquisición de datos que es la encargada de obtener las variables de la prueba para luego llevarlo por un acondicionador de señal, para finalmente pasarlos por filtros y al conversor analógico-digital, además esta tarjeta posee múltiples canales para adquisición de datos. La transferencia de información desde la tarjeta electrónica al computador es a través del estándar RS232. Por último el software se encarga de guardar y graficar las curvas características del ensayo.

1.2.2 Limitaciones operacionales.

En la primera modificación de la máquina de ensayo de polímetros se es necesario tener a un operario, el que será el encargado de registrar los datos de la prueba. La probabilidad que el operario cometa un error en la toma de los datos es alta debido al uso de los indicadores analógicos.

El uso de Labview implica un sistema más costoso, no solo porque se es necesario una tarjeta electrónica y software adicional, sino porque se requiere el uso de un computador, el cual debe de estar encendido todo el tiempo que dure la prueba del polímero. Como ya se dijo este puede llegar a durar hasta 42 días, consumiendo recursos energéticos innecesarios (tener encendida el ordenador durante el ensayo).

1.3 Justificación para los cambios.

Se plantea el diseño de una tarjeta de adquisición de datos y control de temperatura con lo cual podemos prescindir del operario y disminuir notablemente el error en la toma de los parámetros del ensayo.

También evitamos mantener un computador (el cual posee muchos otros recursos que no se los están aprovechando), una tarjeta con multicanales y el software, logrando de esta manera bajar el costo del sistema. El software planteado en el proyecto es mucho más personalizado y amistoso que el ya citado anteriormente.

CAPITULO 2

2 TEORIA

2.1 Ensayo de carga constante en polímeros

2.1.1 Introducción

El método consiste en la medición del estiramiento del plástico como función del tiempo, sometido bajo a una carga constante y con determinadas condiciones ambientales (temperatura, humedad). Los datos obtenidos del ensayo pueden servir para comparar materiales, en el diseño de partes fabricadas o para caracterizar plásticos durante un período de funcionamiento sometido a ciertas condiciones normales de operación.

2.1.2 Deformación progresiva bajo carga constante (Curva Creep)

Cuando un material plástico es sujeto a una carga constante este se deforma de manera continua. La deformación inicial es aproximadamente calculada por su módulo deformación-carga aplicada. El material continuará deformándose lentamente en un tiempo indefinido o hasta que se produzca una ruptura en el mismo.

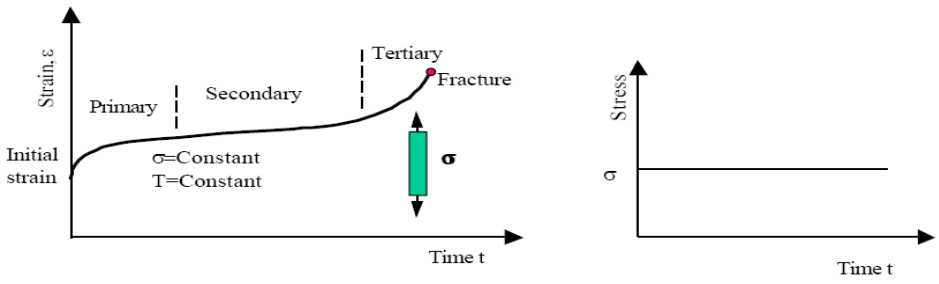


Figura 2.1 Curva de deformación bajo carga constante

La curva creep posee tres regiones, la primera es la etapa en donde comienza la aplicación de una carga constante (tensión), la característica aquí es el rápido decremento de la curva con el tiempo. Luego esta alcanza un estado estable el cual es llamado como el estado secundario seguido por un rápido incremento y fallo del material (tercera región). A este fenómeno de deformación bajo carga constante en el tiempo es denominado Creep. La curva descrita es ideal, algunos materiales no pasan por la región secundaria, mientras la tercera etapa solo ocurre a altos niveles de tensión. El grado de creep depende de muchos factores tales como el tipo de material, magnitud de la carga, temperatura y tiempo. La norma estándar de este método es ASTM D2990.

Si la carga aplicada es retirada antes que ocurra la ruptura del material (denominado ensayo Creep), el espécimen intenta recuperar su forma original lentamente, pero en la mayoría de los casos no sucede esto y mantiene algún tipo de deformación. La magnitud de esta deformación permanente depende tanto de cuanto tiempo estuvo expuesta la muestra, de la tensión aplicada, y la temperatura.

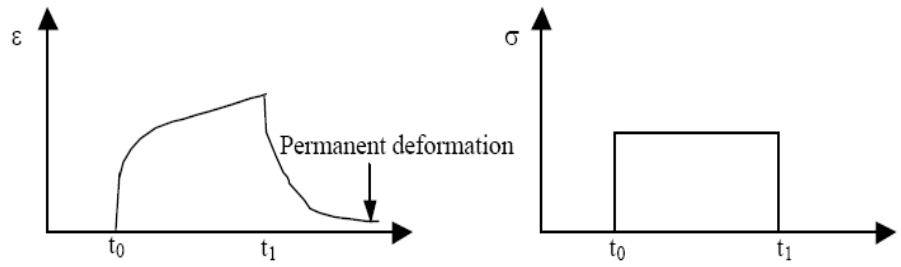


Figura 2.2 Curva de deformación permanente.

El ensayo Creep de ruptura es básicamente similar a la prueba Creep con la excepción que la prueba continúa hasta que el material falla. Debido a que una mayor tensión es aplicada, el tiempo de la prueba será menor. Basados en los datos obtenidos (tiempo de falla del material con una cantidad específica de tensión), se puede determinar una tensión segura de operación durante un tiempo determinado.

2.1.3 Descripción del ensayo.

La prueba inicia escogiendo el espécimen con las características específicadas en la norma AST D638-03, la muestra es tomada desde los extremos más delgados por unas pinzas, un extremo esta fijamente agarrado a un soporte, por el otro lado, la pinza de ese extremo se sujeta a un peso. El espécimen quedara dentro de un cuerpo, aislando el espécimen del exterior, logrando tener un ambiente al cual podremos controlar la temperatura. Luego que el espécimen queda entre la pinza fija y la pinza sujeta al peso (en estado de reposo) se retira la base del peso, en este momento empieza el

estiramiento de la muestra. En el tiempo la muestra se va deformando progresivamente pero no llega nunca a la ruptura (no es el caso, pero existe una norma para este ensayo). El técnico en materiales es el encargado de escoger los parámetros de la prueba tales como peso, temperatura, tipo de espécimen y cada que tiempo y cuanto durará, además se es necesario registrar en cuanto se deforma en el tiempo el material. Una vez transcurrido el tiempo del ensayo el especialista procederá a analizar lo datos obtenidos, realizando la matemática pertinente y gráficando puntos para obtener la curva Creep (deformación plástica).

2.2 Familia PIC 18

2.2.1 Introducción a la familia de microcontroladores PIC18Fxxxx

Los PIC18Fxxxx es una familia de alto rendimiento, CMOS, son microcontroladores de 16 bits con conversores analógico-digital integrados. Esta familia ha mejorado en las características del núcleo, tienen 32 niveles en su pila y múltiples fuentes de interrupciones tanto internas como externas. La separación del bus de datos e instrucciones de la arquitectura Harvard permite un amplio rango palabras de instrucción de 16 bits en comparación con la de los datos que es de 8 bits. Las instrucciones se ejecutan en su solo ciclo de reloj a excepción de ciertos ramales de programa que necesitarán dos ciclos de reloj, en total existen 77 instrucciones que están disponibles. Además, esta familia tiene características especiales para reducir los

componentes externos y así reducir costos, mejorando la confiabilidad del sistema y reduciendo el consumo de energía, por último estos incluyen un detector de bajo y alto voltaje programable (HLVD).

2.2.2 Descripción del microcontrolador PIC18F4550

2.2.2.1 Configuraciones del oscilador.

En el PIC18F4550 incorpora diferentes sistemas de reloj, el módulo USB utiliza la fuente primaria para cumplir las específicaciones Low Speed y Full Speed en donde se provee frecuencias de 6 y 48 MHZ, además posee un sistema de post-escaladores y pre-escaladores para lograr que el módulo USB y el microcontrolador con sus otros periféricos puedan trabajar a frecuencias menores o a la misma frecuencia del USB.

La fuente secundaria es gobernada por el Timer1, en donde no es necesario tener conectado un oscilador entre los pines del oscilador primario.

Un bloque interno de oscilación es también otras de las fuentes de reloj. Esta compuesta por un oscilador de 8 MHZ, la misma que medíante un post-escalador se logra obtener un rango de frecuencias desde 31 KHZ hasta 8 MHZ. Además tiene otro oscilador (RC) de 31 KHZ, el cual puede ser usado como fuente de reloj para aplicaciones especiales como el Watch-Dog.

La operación del oscilador es controlada a través de dos registros de configuración y dos registros de control. Los registros de configuración, CONFIG1L y CONFIG1H, seleccionan el modo del oscilador y las opciónes

de los post-escaladores y pre-escaladores. Los registros de control son OSCCON y el OSCTUNE, el primero es donde habilitamos la fuente de reloj para el microcontrolador (primaria, secundaria, interna) y la frecuencia de trabajo del oscilador interno (si se ha habilitado esta opción), el segundo registro selecciona si la fuente de 32 KHZ es del oscilador RC o desde el de 8 MHZ

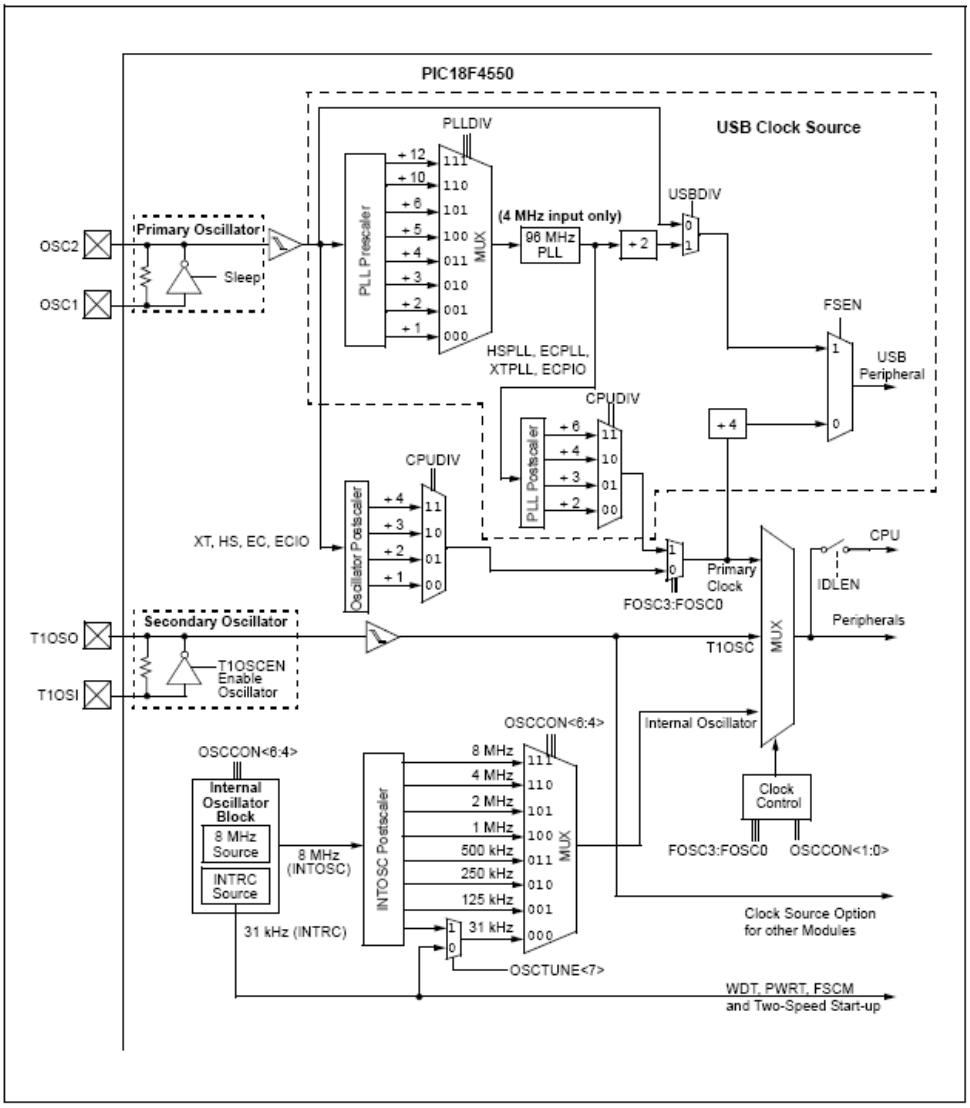


Figura 2.3 Diagrama de bloques del módulo de oscilación del PIC

2.2.2.2 Organización de la memoria.

Existen tres tipos de memoria en esta familia: Memoria de Programa, RAM de Datos y EEPROM de datos.

Los PIC18 tienen implementado un PC (contador de programa de 21 bits con lo cual obtenemos 2Mbytes espacios de memoria de programa, el PIC18F4550 tiene 32Kbytes de memoria Flash. El PC esta conformado por 3 registros de 8 bytes PCL, PCH y PCU, siendo PCL el registro menos significativo, y PCU es el mas significativo, únicamente en el PCL se es posible escribir o leer directamente, mientras que para los demás se logra medíante el uso del PCLATH y PCLATU respectivamente.

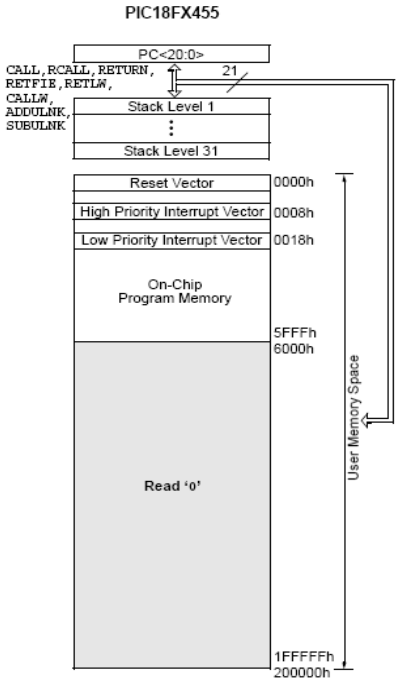


Figura 2.4 Mapa de la memoria de programa del PIC

Tiene dos vectores de interrupción, el vector reset que se encuentra localizado en la dirección 0000h. El vector de interrupción ocupa dos posiciones 0008h y 0018h.

Internamente el reloj interno o externo es dividida para cuatro, obteniendo pulsos que van desde Q1, Q2, Q3 y Q4 (Ver gráfica), los mismos que forman un ciclo de instrucción en donde se ejecuta y decodifica la instrucción desde Q1 hasta Q4.

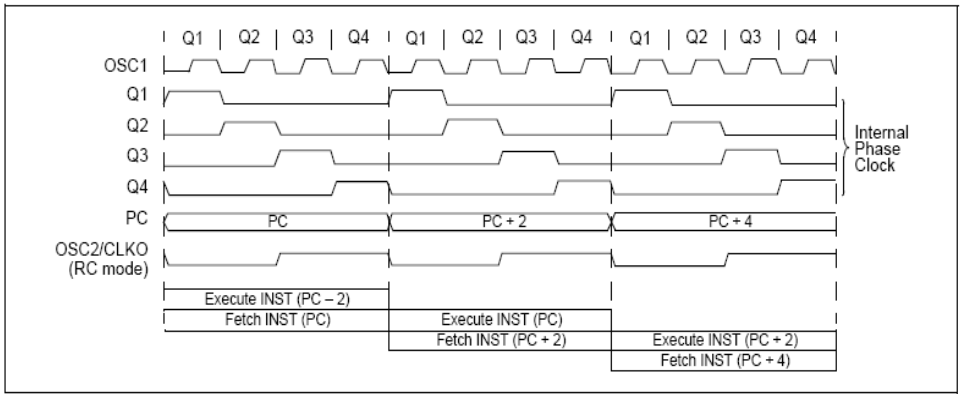


Figura 2.5 Ciclo de instrucción del microcontrolador

Cada registro de la memoria de datos tiene una dirección de 12 bits, permitiendo 4096 bytes. La memoria de datos esta dividida en 16 bancos, los cuales solamente 8 están implementados en el PIC18F4550 (2048 bytes). Los bancos 4 hasta el 7 son utilizados por el módulo USB mientras el módulo este habilitado, de otro modo estos pueden ser utilizados como registros de propósito general. Existen dos tipos de registros que son los registros de funciones especiales (SFRs) y los de propósito general (GPRs), siendo

utilizados para el control y estado de funciones del controlador y periféricos (SFRs), y para el almacenamiento de datos en las aplicaciones del usuario.

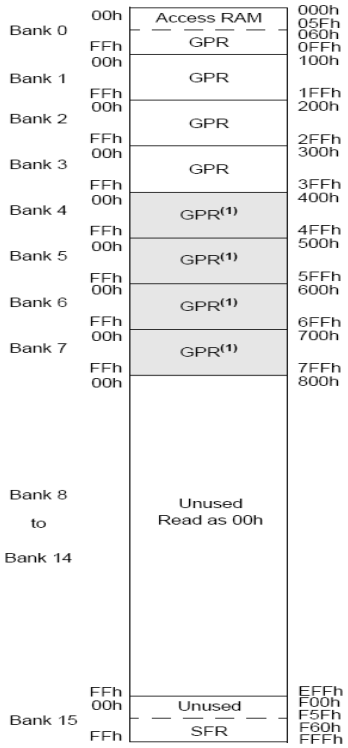


Figura 2.6 Mapa de la memoria de programa del microcontrolador

2.2.2.3 Interrupciones

Los PIC18F4550 tienen múltiples fuentes de interrupciones y una característica de prioridad, esta última permite a la fuente de interrupción tener un nivel de prioridad alto o bajo. Los vectores de interrupción de prioridad alta y baja están situados en las posiciones 000008h y 000018 respectivamente.

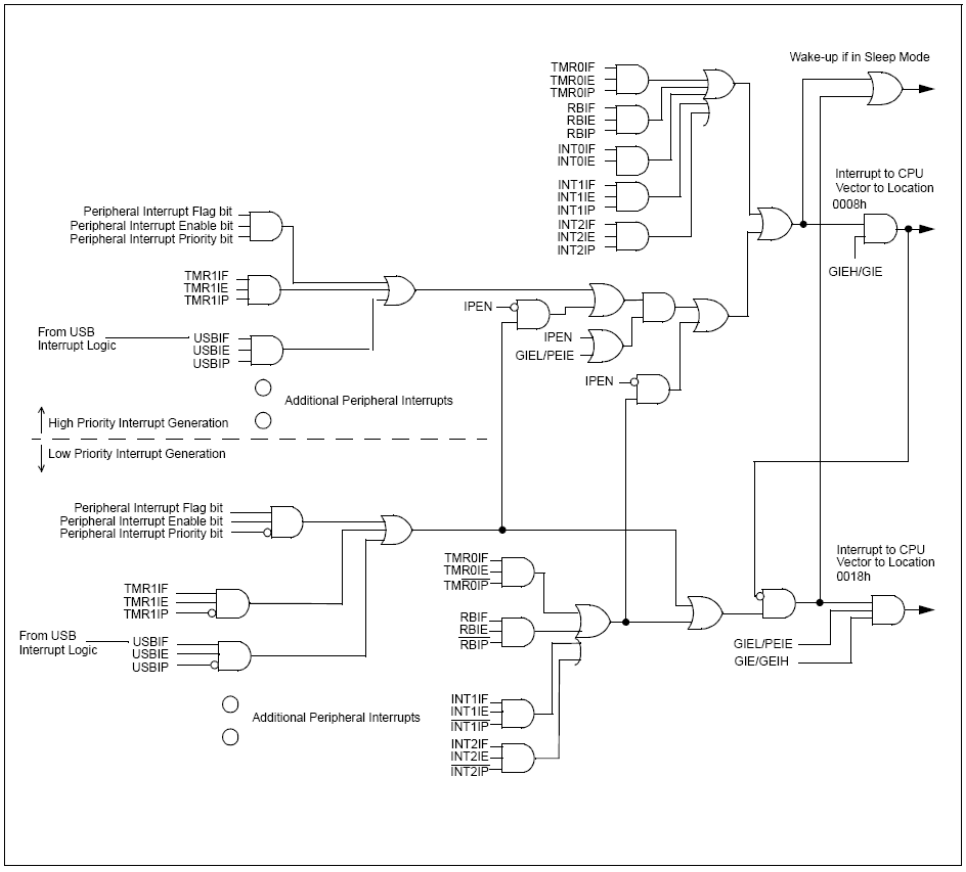


Figura 2.7 Lógica de las interrupciones del microcontrolador

Cada fuente de interrupción tiene tres bits para controlar su operación. La función de estos bits son:

- Bit de bandera indica que un evento de interrupción ha ocurrido.
- El bit habilitador permite al programa desviarse a la dirección del vector de interrupción cuando el bit de bandera es activado.
- La prioridad alta y baja es seleccionada por el bit de prioridad.

Existen diez registros usados para controlar las operaciones de interrupción. Estos registros son:

INTCON (1-2-3). Son registros en el pueden escribir y leer, contiene varios de los bits de habilitación, prioridad y banderas.

PIR (1 y 2). Contiene los bits de bandera para las interrupciones de los periféricos.

PIE (1 y 2). En este registro encontramos los bits habilitadores para las interrupciones de los periféricos.

IPR (1 y 2). Los bits de prioridad están contenidos en este registro. Usar los bits de prioridad requiere que los habilitadores de prioridad estén activos.

RCON. Esta relacionado con la reiniciación del dispositivo por diversas causas. También posee un bit que habilita la interrupción de prioridad.

2.2.2.4 Módulo del Timer0

El módulo puede trabajar como contador o Timer tanto de 8 como de 16 bits. En el modo de timer, el incremento se hace con cada flanco de reloj, es importante notar que hay que tener en cuenta del valor que posea la prescala. En modo contador, su incremento se deberá a los flancos del pin T0CKI, además se puede escoger el tipo de flanco que se necesite. Produce una interrupción en el momento que pasa de FFh a 00h (FFFFh a 0000h si es de 16 bits).

La prescala es programable por software. En el modo de 16 bits, el TMR0H no es el byte más significativo del TIMER0, el cual no se es posible leer y

escribir directamente, para tal objetivo se utiliza el registro mencionado (TMR0H), el cual es siempre actualizado con el byte más significativo del TIMER0. El T0CON es el registro que gobierna este módulo, en donde se configura y habilita todas las características mencionadas.

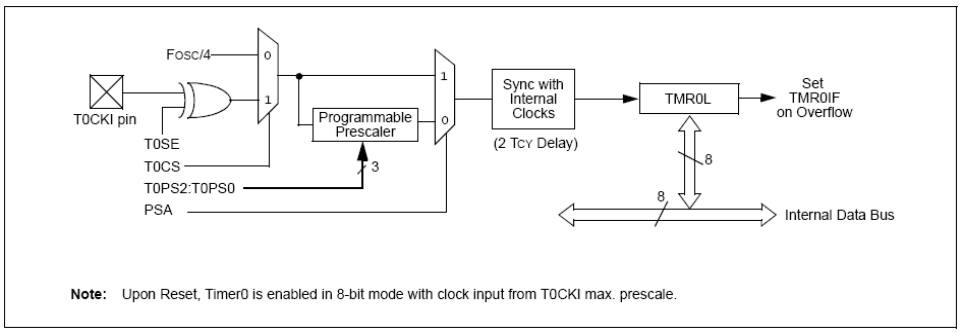


Figura 2.8 Diagrama de bloques del Timer 0 en modo 8 bits

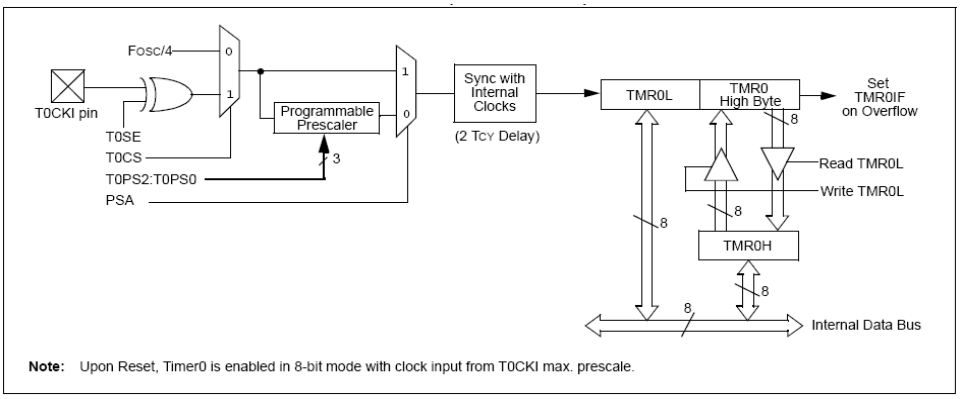


Figura 2.9 Diagrama de bloques del Timer 0 en modo 16 bits

2.2.2.5 Módulo del Timer1

Este módulo puede trabajar como un temporizador o contador asíncrono o asíncrono de 8 o 16 bits e incluso tiene la opción de trabajar como un reloj de

tiempo real (RTC). El módulo incorpora su propio reloj de bajo consumo, el mismo que puede ser utilizado por el microcontrolador para su operación. Tiene registros de 8 bits, los cuales pueden ser escritos o leídos.

Al igual que el TIMER0, produce una interrupción al pasar de FFFFh hasta 0000h. Este módulo es controlado por el registro T1CON.

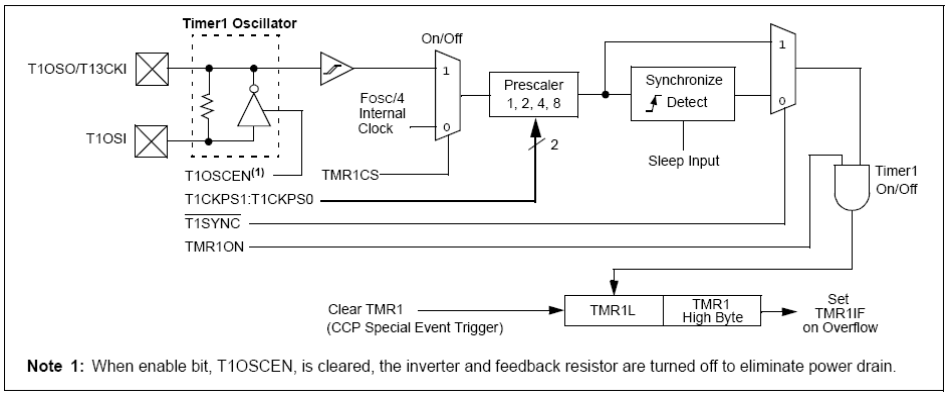


Figura 2.10 Diagrama de bloques del Timer 1 en modo 8 bits (PIC)

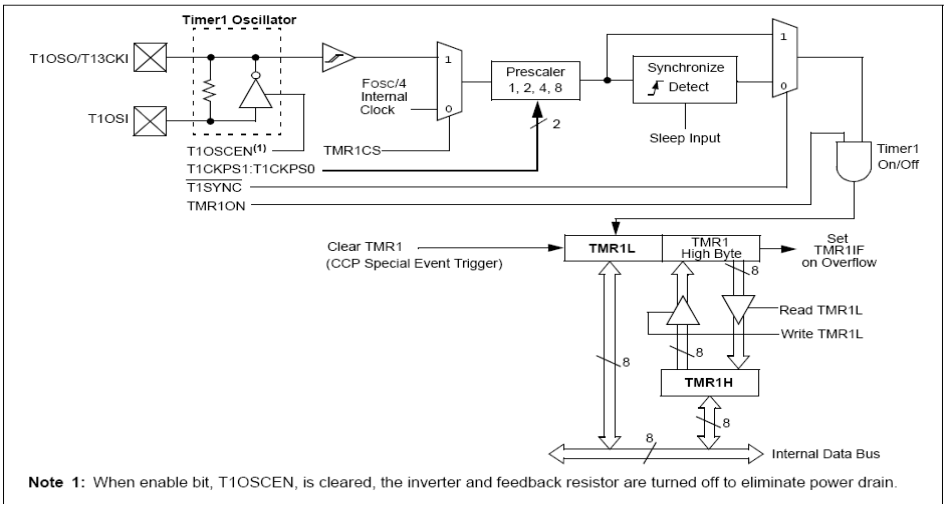


Figura 2.11 Diagrama de bloques del Timer 1 en modo 16 bits

2.2.2.6 Módulo del Timer2

En el módulo encontramos un bloque de pre-escala que es configurado en el TMR2 (1:1, 1:4 y 1:16). Además el módulo posee dos registros de 8 bits que son el TMR2 y el PR2. El primero es donde se producen incrementos medíante los flancos de reloj que van desde 00h hasta FFh, para luego compararse con el PR2 en cada ciclo de reloj (registro de período), cuyo valor es cargado por software. En el momento que TMR2 alcanza a PR2 se produce una salida, la misma que esta disponible para el módulo CCP, donde esta es utilizada como base de tiempo para la operación en modo PWM, además produce que el TMR2 vuelva al valor de 00h. La misma salida pasa por un bloque de postescala que va desde 1:1 hasta 1:16 (seleccionada con 4 bits del registro T2CON), pudiéndose utilizar la salida de este bloque como una interrupción activando su bit habilitador en el registro PIE1. El módulo TMR2 es gobernado por el registro TMR2.

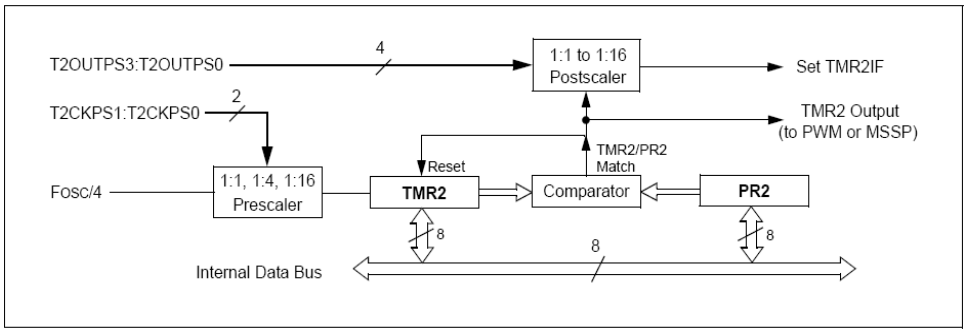


Figura 2.12 Diagrama de bloques del Timer 2

2.2.2.7 Módulo del Timer3

El módulo Timer3 puede operar tanto como timer o contador (síncrono o asíncrono). Tiene dos registros de 8 bits (TMR3H y TMR3L) que pueden ser escritos o leídos. De igual forma que en el Timer1 el byte más significativo no se es posible leer o escribir directamente por lo que se utiliza el TMR3H, obteniendo mediante este registró la actualización del byte más significativo del Timer3 cada vez que se lee o escribe en el TMR3L. Tiene la opción de seleccionar el reloj interno o el oscilador del TMR1. Otra característica es la interrupción que se produce en el momento que se produce un desborde (FFFFh a 0000h), en el registro PIR2 se habilita esta opción.

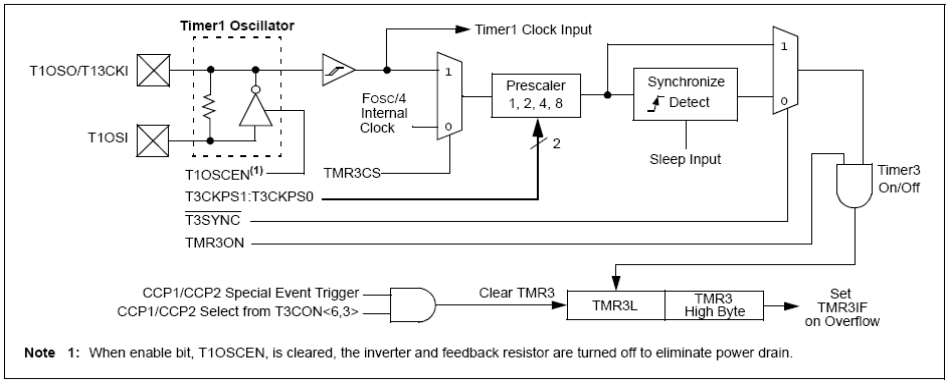


Figura 2.13 Diagrama de bloques del Timer 3 en modo 8 bits

Si el módulo CCP2 es configurado para generar un evento especial en modo comparación, la señal producida podrá resetear el Tiemer3, o también podrá ser usada para empezar una conversión analógica-digital si el módulo A/D esta habilitado. El T3CON gobierna el módulo.

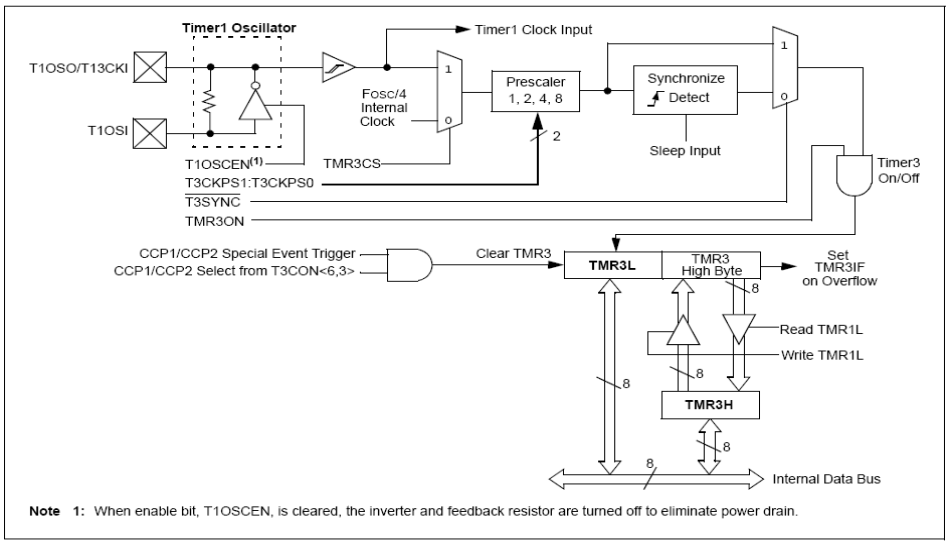


Figura 2.14 Diagrama de bloques del Timer 3 en modo 16 bits

2.2.2.8 Módulo USB

Este microcontrolador puede trabajar en modo Full Speed y Low Speed, y es compatible con la interfaz serial USB (SIE USB). Un transceiver interno opera como la interfaz entre SIE y el bus USB, además posee un regulador de 3.3V interno. Existe la posibilidad de colocar un transceiver externo, previamente desactivando el interno, una fuente externa es también posible colocar.

EL control y estado del USB es mediante los siguientes registros:

UCON. Contiene los bits necesarios para el comportamiento del módulo durante las transferencias.

UCFG. Antes de empezar la comunicación, el hardware debe haber sido configurado, la mayoría de estas configuraciones se las realiza por medio de este registro, en donde podemos establecer si va a trabajar en Full Speed o

Low Speed, si las resistencias pull-up y transceiver internos serán habilitadas.

USTAT. Es un registro de solo lectura en donde se reporta el estatus de una transacción dentro del SIE, en el momento que SIE produce una transferencia, este buffer debe ser leído para conocer su estatus, contiene el número de endpoint, la dirección etc.

UPEn. Cada uno de los 16 endpoints tiene un registro de control independiente, aquí podemos habilitar el handshake del buffer, las transacciones OUT, IN o SETUP etc.

UADDR. Contiene la dirección USB que el periférico debe decodificar cuando es activado.

USB RAM

Los datos USB se mueven entre el núcleo del microcontrolador y el SIE por medio del espacio de memoria conocido como USB RAM de 1 KB, el mismo que esta localizado entre el banco 4 al 7, el banco 4 es utilizado para el control de los endpoint (Buffer Descriptor), mientras que los restantes bancos están disponibles para datos USB.

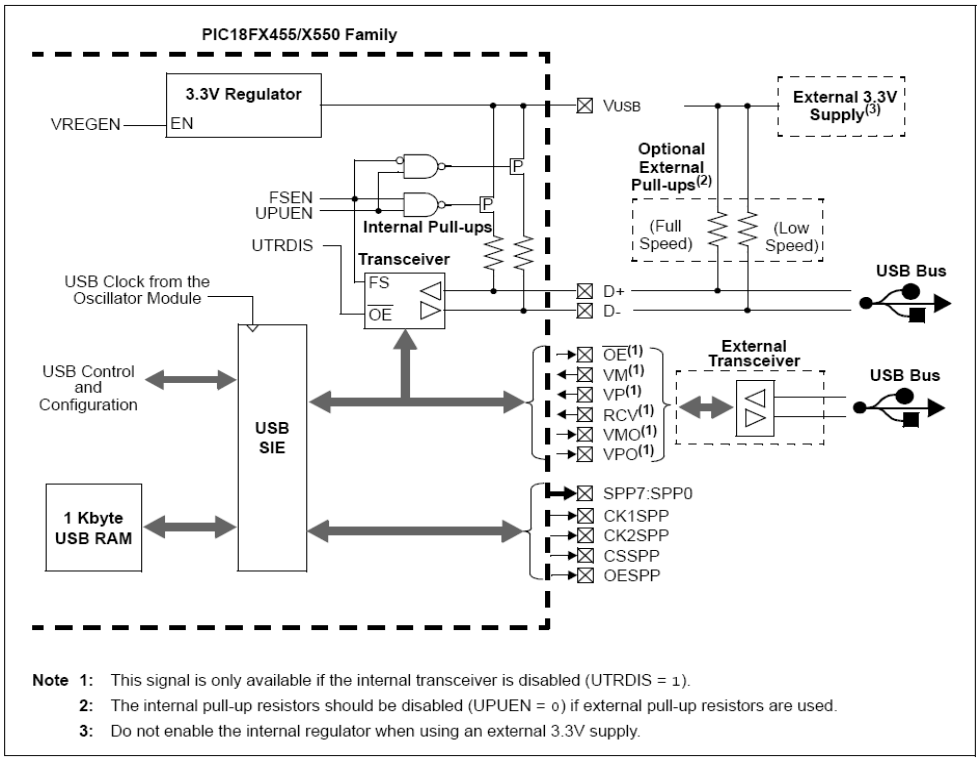


Figura 2.15 Diagrama de bloques del módulo USB

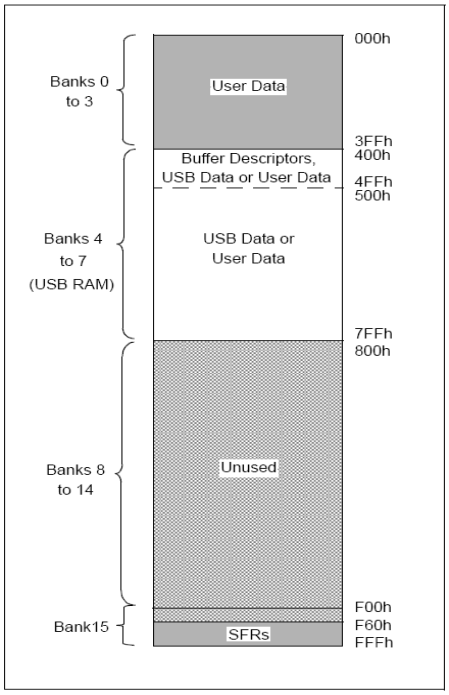


Figura 2.16 Implementación de USB RAM en la memoria de datos

Interrupciones.

El módulo USB puede generar múltiples interrupciones, para acomodar estas múltiples fuentes, el módulo tiene su propia estructura lógica de interrupciones similar a la que tiene el microcontrolador. Las interrupciones son habilitadas con un conjunto de registros de control, siendo estas canalizadas en una sola petición de interrupción localizada en el registro PIR2<5> (USBIF). Esta estructura esta conformada por dos niveles, el nivel alto consiste en todas las interrupciones de estado, mientras el nivel inferior son constituidas por condiciones de error en el USB.

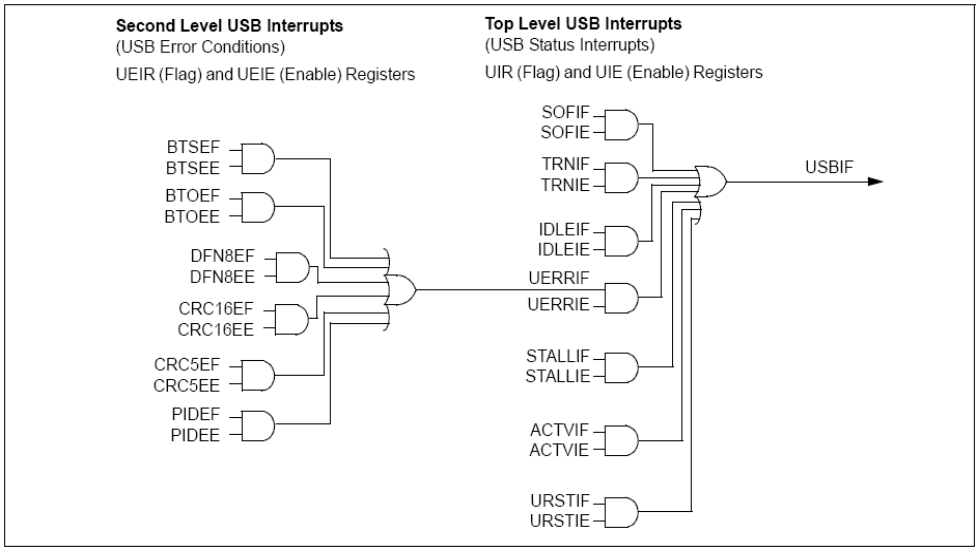


Figura 2.17 Estructura lógica de las interrupciones USB

UIR. Es el registro que contiene los bits de bandera de las interrupciones de estado, los mismos que deben ponerse a ´0´ mediante firmware.

UIE. Aquí encontramos los bits habilitadores del registro UIR.

UEIR. Contiene los bits de bandera de cada uno de las fuentes de error dentro del periférico USB.

UEIE. Este registro posee los bits que habilitan las fuentes de interrupción de error del USB.

2.3 USB

2.3.1 Introducción al USB

USB (Universal Serial Bus) es una interfaz suficientemente versátil para un amplio rango de dispositivos. Los estándares USB incluyen ratones, teclados, escáneres, impresoras y dispositivos de audio y video, también unidades de adquisición de datos, sistema de control y otros tipos de dispositivos con funciones especiales.

2.3.1.1 Beneficios para el usuario.

Una interfaz para varios dispositivos. USB es lo suficientemente versátil para ser utilizado en diferentes clases de dispositivos. En lugar de tener un tipo de conector diferente para cada dispositivo, se utiliza un conector común. En el momento que un usuario conecta un periférico USB al PC, Windows lo detecta y carga el driver apropiado para la aplicación. No hay la necesidad de resetear el sistema antes de usar el periférico. Con USB no hay la necesidad de agregar algún tipo de tarjeta de expansión para cada periférico, una típica computadora tiene cuatro o más puertos, además si existiera el caso de usar

mas de estos puertos solo es necesario conectar un hub USB para tener un mayor número de puertos. El conector USB esta dispuesto de tal forma que no se pueda caer en error al conectarlo. Cinco metros es la máxima distancia, y si se quiere incrementar su distancia se puede utilizar un hub. Su conector es pequeño y compacto en contraste con el típico RS-232 y los conectores paralelos.

El usuario puede conectar y desconectar un periférico USB cuando lo desee, si el sistema esta o no energizado sin dañar el dispositivo o el PC, el sistema operativo detecta cuando un periférico es vinculado y esta listo para su uso. La interfaz USB incluye dos líneas de alimentación que provee al equipo de +5 Voltios del computador o hub que este conectado, logrando que sea más compacto.

El bus del USB puede soportar 3 niveles de transferencia de información HIGH SPEED a 480 Megabits/sec., FULL SPEED a 12 Megabits/sec. y LOW SPEED a 1.5 Megabits/sec. Los PC actuales soportan los tres tipos. La velocidad del bus describe la rapidez que la información viaja en el bus, además se envían información acerca del estado y control señales de error.

La especificación USB 1.0 definió LOW SPEED y FULL SPEED, el primero fue incluido por dos razones, una de estos argumentos era el uso de ciertos periféricos tales como el ratón donde se necesita que el cable posea flexibilidad para tener suficiente movilidad, para este modo de operación no es necesario utilizar cables del tipo par trenzado ni cables demasiados

protegidos en donde se perdería flexibilidad. La otra razón es el bajo costo de fabricación. FULL SPEED fue pretendido para reemplazar los periféricos que utilizan RS-232(serial) y el puerto paralelo. HIGH SPEED empezó a ser una opción en USB 2.0.

La confiabilidad del USB es debido al hardware y el protocolo de comunicación. Tanto los driver USB como la recepción de los datos y cables garantizan una interfaz que elimina el ruido que pueda provocar error en los datos. En cambio, el protocolo habilita la detección de errores en la receptor y notifica al transmisor para luego retrasmitir el dato en caso de problemas en la transferencia de información.

Incluso a pesar que USB es una tecnología más compleja que anteriores interfases, los componentes y cables no son costosos. Un dispositivo con una interfaz USB es probablemente igual o menos costoso que un dispositivo equivalente con una clásica interfases o una de las más recientes tal como IEEE-1394

Posee un circuito de bajo consumo y un código que automáticamente puede poner el sistema en reposo logrando de esta forma ahorro de energía, y un tiempo de recarga de baterías mas largo.

USB originalmente utiliza una interfaz cableada, pero ahora existe dispositivos inalámbricos que usa el USB para comunicarse con los PCs.

2.3.1.2 Beneficio para los diseñadores.

USB tiene cuatro tipos de transferencias que hacen la interfaz fiable para muchos tipos de periféricos. Existen transferencias adecuadas para intercambiar grandes y pequeños bloques de información, con o sin tiempos límites. Para datos que no pueden tolerar retardos, USB puede garantizar un ancho de banda o un tiempo máximo entre transferencias. Esta habilidades son especialmente bienvenidas bajo Windows, donde el acceso a los periferias en tiempo real es algunas veces es un desafió. Aunque los sistemas operativos, drivers de los dispositivos y las aplicaciones pueden introducir inevitables retardos, USB lo hace tan fácil posible para lograr transferencias que están cerca del tiempo real.

A diferencia de otras interfases donde ciertas líneas se las señala para trabajar en determinada forma, USB no hace ningún tipo de asunciones, por ejemplo, las líneas de estado y control en el puerto paralelo fueron asignadas con la intención de comunicar el PC con la impresora. Hay cinco líneas que fueron asignadas para ciertas funciones tales como identificar si hay papel en la impresora o si el dispositivo esta ocupado. Cuando los diseñadores comenzaron a usar el puerto para escáner u otro tipo de periférico que transmite grandes cantidades de información al PC fue una gran limitación tener 5 líneas para este fin.

Para comunicación con los periféricos comunes tales como teclado, impresora, ratón, USB ha definido clases que especifican los requerimientos de lo dispositivos y protocolos.

Un sistema operativo que soporta USB tiene 3 requerimientos mínimos que debe cumplir los cuales son:

- Detección del dispositivo USB en la conexión y desconexión.
- Al existir un nuevo dispositivo conectado, este debe ser capaz de descubrir como intercambiar información con este.
- Provee un mecanismo que habilita el software para comunicación entre el hardware del USB y la aplicación que quiere acceder a los periféricos USB.
- Hablando en un nivel más alto, el sistema operativo podría también incluir driver de las clases que habilitan los programas de aplicación para acceder a los dispositivos.

Por otro lado los periféricos para manejar la comunicación incluyen en el hardware microcontroladores, los cuales son los encargados de manejar todos estos detalles. El periférico es responsable de responder las peticiones para enviar y recibir datos usados en la identificación y configuración del dispositivo y para leer y escribir otros datos en el bus. Algunas funciones son codificadas en el hardware y no necesitan ser programadas.

2.3.1.3 Limitaciones

Todas las ventajas que presenta inducen a creer que este es un buen candidato para el uso de algunos periféricos. Pero como toda interfaz tiene sus limitantes como detallamos a continuación.

USB es versátil, pero no esta diseñado para hacer todo, En modo HIGH SPEED lo hace competitivo con la interfaz IEEE-1394a (Firewire) con 400 Megabits/sec, pero la interfaz IEEE-1394b es mucho más rápida todavía, a 3.2 Gigabits/sec.

USB fue diseñado como un bus de expansión fijo con la expectativa que los periféricos estarían cerca, su cable puede llegar a ser hasta 5 metros. A diferencia de otras interfases donde se permite mayor alcance, tales como el RS-232, RS-485 ETHERNET y IEEE-1394b. Se puede incrementar la distancia a 30 metros colocando 5 hubs. Otra opción cuando se necesita transmitir a distancias grandes es cambiar de USB a RS-485.

En la comunicación USB, solo se puede comunicar el periférico con el host, pero no host con host, tampoco lo hace entre periféricos como lo hacen otras interfases como IEEE-1934. USB provee una parcial solución con USB ON-THE-GO, Un dispositivo ON-THE-GO puede funcionar como un periférico y un host de capacidad limitada, dos host se pueden comunicar mediante vía PC a PC donde estos se conectan con un cable de red entre ellos y se transfiere información desde cada dispositivo en máquinas diferentes.

USB no puede transmitir información simultáneamente a diferentes dispositivos en el bus, los envía individualmente. ETHERNET o IEEE-1394 posee esta habilidad.

Los antiguos computadores y periféricos no tienen puertos USBs, una solución es hacer una conversión de USB a la anterior interfaz, pero esto es útil cuando se esta manejando un solo tipo de periférico tal como pasa en el puerto paralelo con la impresora.

Si se necesita tener puertos USB en máquinas donde no la posean, esto se lo puede lograr agregando una tarjeta (controlador host) e instalando un sistema operativo que soporte USB.

2.3.1.4 Componentes

Los componentes físicos del USB consiste de circuitos, conectores y cables entre un host y uno mas dispositivos. El host es un PC u otra computadora que contiene el controlador Host y el Hub. Estos dos componentes trabajan juntos para habilitar el sistema operativo para comunicarse con los dispositivos en el bus. El controlador Host da formato a los datos para la transmisión en el bus y traducir los datos recibidos a un formato que los componentes del sistema operativo puedan entender. Puede también realizar otras funciones relacionadas al manejo de la comunicación. El Hub raíz tiene uno o más conectores para vincularse con los dispositivos. En combinación, el hub raíz y el controlador host detecta cuando el dispositivo ha sido conectado o removido, lleva las peticiones desde el controlador y

pasa los datos entre el dispositivo y el controlador host. Los dispositivos son los periféricos y hubs adicionales que se conectan en el bus. Un Hub tiene una o más puertos para conectar los dispositivos. Cada dispositivo debe contener circuito y código que conozca como comunicar con el Host. Las especificaciones USB definen los cables y conectores de los dispositivos y hubs.

De manera general un puerto tiene una determinada localización física y dirección, la misma que esta disponible para adicionar circuitos. Usualmente los circuitos terminan en conectores que vinculan un cable a un periférico. Un software puede monitorear los puertos leyéndolos o escribiéndolos. USB difiere de otra clases de puertos debido a que estos utilizan un solo camino para llevar los datos, en otras palabras solo se necesita un bus para los diferentes puertos, pero un solo puerto puede transmitir datos a las vez, situación que no ocurre por ejemplo con el RS-232, en el cual se puede transmitir dos a las vez, pero necesita un cable diferente para cada conector, compartiendo el mismo acho e banda.

2.3.1.5 Tareas del host.

Para la comunicación con un dispositivo USB se necesitan de dos cosas, una es el hardware el cual esta compuesto por el controlador host y hub raíz, y el otro es el software el cual se encarga de brindar los mecanismos para la comunicación con el hardware del USB. El host tiene que conocer los dispositivos que están en el bus y sus capacidades, también debe hacer lo

posible para asegurar que todos los dispositivos puedan enviar y recibir datos.

Afortunadamente, el hardware del controlador host y sus drivers en Windows hace mucho del trabajo de manejo del bus, cada dispositivo conectado al host tiene su propio driver que habilita la aplicación para comunicarse con el dispositivo, en otros casos el fabricante entrega el correspondiente software.

Las aplicaciones no tienen que preocuparse por los detalles de comunicación, lo único que deben hacer es enviar y recibir los datos con los comandos destinados a este fin, variando estos del lenguaje de programación utilizado.

Al energizar el PC el host se entera de todos los dispositivos conectados a el, y comienza un proceso denominado enumeración, en el cual se le asigna una dirección a cada dispositivo. Cuando se remueve un dispositivo o se vincula otro, el host realiza una nueva enumeración y lo remueve o lo vincula a su lista de los dispositivos disponible para aplicación.

El controlador host maneja el flujo en el bus, puede ocurrir que algunos dispositivos quieran transmitir a la vez, para lo cual el host divide el tiempo disponible en segmentos llamados frame o microframe, dando a cada transmisión una porción del frame. Transferencias que ocurren en un específico momento están garantizadas para tener el tiempo que ellos necesitan para cada frame. Durante la enumeración, los driver de los dispositivos piden el ancho de banda que estos necesitarán para alguna

transferencia. Si el ancho de banda no esta disponible, el host no permitirá comenzar la comunicación, los driver podrían pedir un pequeño ancho de banda o simplemente esperar hasta que este disponible.

Cuando se transfieren datos, el host agrega un bit de error, en la recepción de datos, el dispositivo realiza cálculos en los datos y compara los resultados con el bit de error recibido. Si el resultado no concuerda, el dispositivo no admite los datos recibidos y el host conoce que debería retransmitir de nuevo. El host puede recibir otras indicaciones tales como que no puede recibir ni enviar datos, el host puede informar al driver del dispositivo de el problema y el driver puede notificar a la aplicación para que este pueda tomar las acciones apropiadas. Poseen dos cables, uno es el +5V y el otro es de tierra, el host le suministra energía al dispositivo en el momento de su conexión y trabaja con el dispositivo para conservar energía lo mas posible.

El atender tareas es el principal trabajo del host, el cual es intercambiar datos con los periféricos, En algunos casos, el driver solicita al host atender los datos recibidos o enviados en intervalos definidos, mientras en otros el host comunica solo cuando una aplicación u otro software pide una transferencia. Los driver reportan algún problema a la aplicación apropiada.

2.3.1.6 Tareas de los periféricos

Las tareas de los periféricos son reflejos del host, Cuando el host empieza la comunicación, el periférico debe responder, Este no puede comenzar una comunicación por si solo, debe esperar y responder a una comunicación del

host. El controlador USB del periférico maneja muchas de las responsabilidades del dispositivo en el hardware. La cantidad de soporte requerida por firmware del dispositivo varía con el chip.

Cada dispositivo monitorea la dirección contenida en cada comunicación en el bus, si la dirección no concuerda con la almacenada en el dispositivo, el dispositivo ignora la comunicación. Pero al concordar, el dispositivo almacena los datos en el buffer de recepción y produce una interrupción para indicar que los datos han llegado. En casi todos los chips estas funciones ya están incluidas en su hardware. Por lo tanto, el firmware no tiene la necesidad de tomar acción o decisión hasta que el chip ha detectado una comunicación conteniendo la dirección del dispositivo.

En el momento que se energiza el dispositivo, el host solicita cierta información del dispositivo tales como las características o el estado del periférico, esto ocurre durante la enumeración, incluso también puede enviar peticiones en algún tiempo después de la enumeración. Hay once tipos de peticiones estándares, sin embargo el dispositivo no tiene que responder cada una de estas. El periférico luego de la petición coloca los datos o información del estado en el buffer de transmisión para enviársela al host.

Similarmente al host, el dispositivo también posee un bit de error que se envía con los datos, en la recepción de los datos se hace el cálculo para chequear el error, la respuesta del dispositivo o falta de esta, le dice al host si

debe transmitir una vez más. Estas funciones son implementadas en el hardware del controlador y no se es necesario programarlas.

El dispositivo puede tener su propia fuente u obtenerla del equipo al que se lo vincula. Para estos últimos cuando no hay actividad en el bus durante 3 milisegundos entra en estado de bajo consumo.

Los periféricos poseen los endpoint que son buffers que almacenan múltiples datos, típicamente son memoria de datos o registros en el controlador del chip, los datos almacenados en los endpoints son datos que se han recibido o están esperando para ser transmitidos al host. Según las especificaciones USB son las únicas secciones de los dispositivos para ser fuente o receptores de flujos de información. Esto quiere decir que los endpoints solo pueden tener comunicación en solo sentido, a excepción del endpoint de control de transferencia que es bidireccional. Hay dos características que tienen los endpoints, cada uno de estos tiene un número que va desde 1 hasta 15, y la dirección que es dada arbitrariamente por el host. Un IN endpoint es el que tiene los datos a enviar a el host y un OUT endpoints almacena los datos que recibidos del host. El endpoints de control posee un para de IN y OUT endpoints que comparten el mismo número. Cada dispositivo debe tener un endpoint 0 configurado como endpoint de control. Cada transacción en el bus empieza con un paquete que contiene el número de endpoint y el código que indica la dirección del flujo de datos (IN, OUT, SETUP) y si se esta inicializando una transferencia de control.

Una transacción SETUP es una del tipo OUT, además un dispositivo siempre tiene que aceptar e identificarlo , debido a que esta tiene las peticiones que debe responder el dispositivo. Cada transacción debe contener la dirección del dispositivo y la dirección del endpoint, cuando el dispositivo recibe un paquete OUT con la dirección del dispositivo, el endpoint almacena los datos y el hardware produce una interrupción, esta rutina de interrupción procesa los datos y toma las acciones respectivas, Si en cambio recibe un paquete IN con la dirección del dispositivo y lo datos están listos para ser enviados, el hardware los envía desde el endpoint especificado hacia el bus, provocando una vez más una rutina de interrupción, la cual dejará lista para la próxima transacción.

2.3.2 Transferencias

Una transferencia esta compuesta por transacciones y esta a la vez se conforman por paquetes de información. Existen cuatro tipos de transferencias, las cuales son de Control, Interrupción, Bulk e Isocronica.

2.3.2.1 Tipos de transferencias

USB esta diseñado para manejar diferentes tipos de periféricos, por lo tanto los requerimientos son diversos (Tiempo de respuesta, monitoreo de señales de error etc.) Es por eso que existen cuatro tipos de transferencias, donde se puede elegir cual es el más acto para determinado propósito.

Transferencia Control Habilita al Host para enviar a los dispositivos peticiones definidas por las especificaciones USB, las mismas que son usadas por el controlador Host para aprender del dispositivo (Enumeración). La transferencia de control es también usada para llevar peticiones definidas por una clase o un fabricante en particular.

Transferencia Interrupción. Aquí podemos transferir información en un tiempo determinado, Además estos dispositivos están siendo observados periódicamente notificando al Hub de su vinculación o no. El teclado o ratón de una PC utilizan este tipo de transferencia.

Transferencia Isócronica. Es tipo de transferencia ha garantizado el tiempo. Es útil cuando se requiere una velocidad de transmisión o recepción de datos constante o en un tiempo específico, donde errores ocasionales pueden ser tolerados. Los datos transmitidos pueden ser de audio o video que necesitan ser tratados en tiempo real.

2.3.2.2 Elementos de la transferencia

Las transferencias consisten en una o más transacciones, estos están definidos por el trabajo que realizan y por la dirección del flujo de datos. Las transacciones están compuestas por paquetes de información, denominados Token, Data y Handshake.

Tipos de transacciones

Setup. Envía las peticiones de una transferencia de control al dispositivo.

OUT. Esta transacción envía datos o información del estatus al dispositivo.

IN. Los utiliza el Host para recibir datos o información de status del dispositivo

Tipos de paquetes

Cada transacción está compuesta por paquetes que ocurren de manera secuencial. Cada paquete está conformado por el PID (Packet identificación) y podría también tener, dependiendo de la transacción, la dirección del endpoint, información del status, o un número de frame, los mismos que a continuación detallamos.

Token. Lo utiliza el Host para empezar una comunicación, el PID del paquete indica el tipo de transacción. Además tiene un uso adicional que es el de llevar el marcador SOF (Start of Frame), el cual es el tiempo de referencia que el host envía en intervalos de 1 milisegundo. A full speed y 125 microsegundos a high speed, con lo cual puede sincronizarse.

Data. En este tipo de paquete el Host o el dispositivo pueden transferir datos.

Handshake. Es donde el Host o dispositivo pueden enviar información de reconocimiento luego que se ha transmitido datos. El PID que contiene los códigos de estado tales como ACK, NACK, STALL.

Tipos de PID en los paquetes Handshake

ACK (acknowledge) Indica que el host o el dispositivo recibieron datos sin errores.

NAK (negative acknowledge) Esto significa que el dispositivo está ocupado o no hay datos para enviar, Cuando el Host envía datos en el momento que el

dispositivo esta ocupado, este retorna un NAK en el paquete handshake. Por otro lado cuando el host solicita datos del dispositivo y este no tiene nada que enviar, es este caso también retornará un NAK. En ambos casos esto es una condición temporal, donde el host podrá intentar después.

STALL Esto PID se puede dar por tres razones: la petición de control no es soportada por el dispositivo, la petición fallo, endpoint falla. Una petición falla cuando esta es soportada por el dispositivo pero no puede ser realizar la acción correspondiente. Endpoint falla (condición HALT) es cuando el buffer queda inhabilitado para recibir o enviar datos.

2.3.2.3 Transferencia de Control

Cada dispositivo de ser capaz de soportar por defecto el pipe en el endpoint 0, los dispositivos pueden tener pipes adicionales para la transferencia de control, pero en realidad no es necesario, incluso si el dispositivo necesita enviar una gran cantidad de información. El host le asigna un ancho de banda para la transferencia de control de acuerdo al número y tamaño de peticiones, y no por el número de endpoint. El agregar mas pipes para este tipo de transferencia no ofrece ninguna ventaja.

Este tipo de transferencias pueden ser de escritura o de lectura, esto depende del tipo de petición que el HOST realice.

Cada transferencia esta formada por transacciones o etapas (SETUP, DATA, STATUS), las mismas que están compuestas por paquetes de datos (TOKEN, DATA, HANDSHAKE). Tanto la transacción del setup como del

status siempre están presentes, la transacciones de datos es opcional, esta se utilizará cuando alguna petición requiera llevar datos.

En las transferencias de escritura, los datos de la transacción de datos viajan desde el host hasta el dispositivo, si no hay este tipo de transacción, es igualmente considerada como una transferencia de escritura. En la transferencia de lectura los datos de la transacción ya mencionada viajarán desde el dispositivo hacia el host. Dependiendo de la velocidad del dispositivo varía el tamaño de los paquetes de datos en las transacciones de datos. Para el modo low speed es de 8 bytes, en cambio si el dispositivo trabaja a full speed el tamaño puede ser de 8, 16 32, 64 bytes, en high es speed es de 64 bytes. Estos llevan información de los bytes transferidos en la transacción de datos y no lo PID y CRC.

Etapa Setup

La etapa del SETUP tiene dos propósitos, una es identificar la transferencia como de control. La segunda es transmitir las peticiones y otra clase de información que el dispositivo necesitará para completar las peticiones.

El dispositivo necesita aceptar y reconocer estas etapas, si se encuentran en medio de una transferencia de control el dispositivo debe abandonar dicha acción, para reconocer la nueva transferencia de control. A continuación se detalla los diferentes paquetes de información de esta transacción.

TOKEN

Propósito: Identificar el receptor y la transacción como SETUP.

Fuente: HOST

PID: SETUP

Contenidos adicionales: dirección del endpoint y dispositivo.

DATA

Propósito: Transmitir las peticiones e información relacionada.

Fuente: HOST

PID: DATA0

Contenidos adicionales: Contiene 8 bytes en 5 campos los mismos que en seguida detallamos.

bmRequestType: Específica la dirección del flujo de datos, tipo de petición y de receptor.

bmRequestType<7>: Dirección del flujo de datos en la transacción de datos.

0: Host > dispositivo (OUT) o no hay transacción de datos.

1: Dispositivo > host (IN)

bmRequestType<6,5>: Específica el tipo de petición.

00: Es una de las peticiones estándar del USB.

01: La petición es definida por una clase específica.

10: Petición definida por un fabricante para un producto particular.

bmRequestType<4-0>: Son bits del receptor que define si la petición es dirigida a:

00000: el dispositivo.

00001: una interfaz específica.

00010: Endpoint.

00011: Otro elemento en el dispositivo.

bRequest: Es un byte que específica la petición, teniendo este un único valor dependiendo de del tipo de petición bmRequestType<6,5>.

00: bRequest específica una de las peticiones estándar USB.

01: bRequest específica una petición definida por una clase del dispositivo.

00: bRequest específica una petición definida por el fabricante

wValue: Son dos Bytes que utiliza el host para pasar información a el dispositivo, el contenido de estos Bytes dependerá del tipo de petición.

wIndex: Son dos Bytes que utiliza el host para pasar información a el dispositivo. Un uso específico es para un índice u offset tales como el número de interfaz o endpoint.

01: La petición es definida por una clase específica.

10: Petición definida por un fabricante para un producto particular.

bmRequestType<4-0>: Son bits del receptor que define si la petición es dirigida a:

00000: el dispositivo.

00001: una interfaz específica.

00010: Endpoint.

00011: Otro elemento en el dispositivo.

bRequest: Es un byte que específica la petición, teniendo este un único valor dependiendo de del tipo de petición bmRequestType<6,5>.

00: bRequest específica una de las peticiones estándar USB.

01: bRequest específica una petición definida por una clase del dispositivo.

00: bRequest específica una petición definida por el fabricante

wValue: Son dos Bytes que utiliza el host para pasar información a el dispositivo, el contenido de estos Bytes dependerá del tipo de petición.

wIndex: Son dos Bytes que utiliza el host para pasar información a el dispositivo. Un uso específico es para un índice u offset tales como el número de interfaz o endpoint.

Como un índice de un endpoint

Windex<0-3>: Indica el número del endpoint.

Windex<7=0>: Endpoint de control o OUT.

Windex<7=1>: Endpoint IN.

Como un índice de una interfaz

Windex<0-7>: Número de la interfaz.

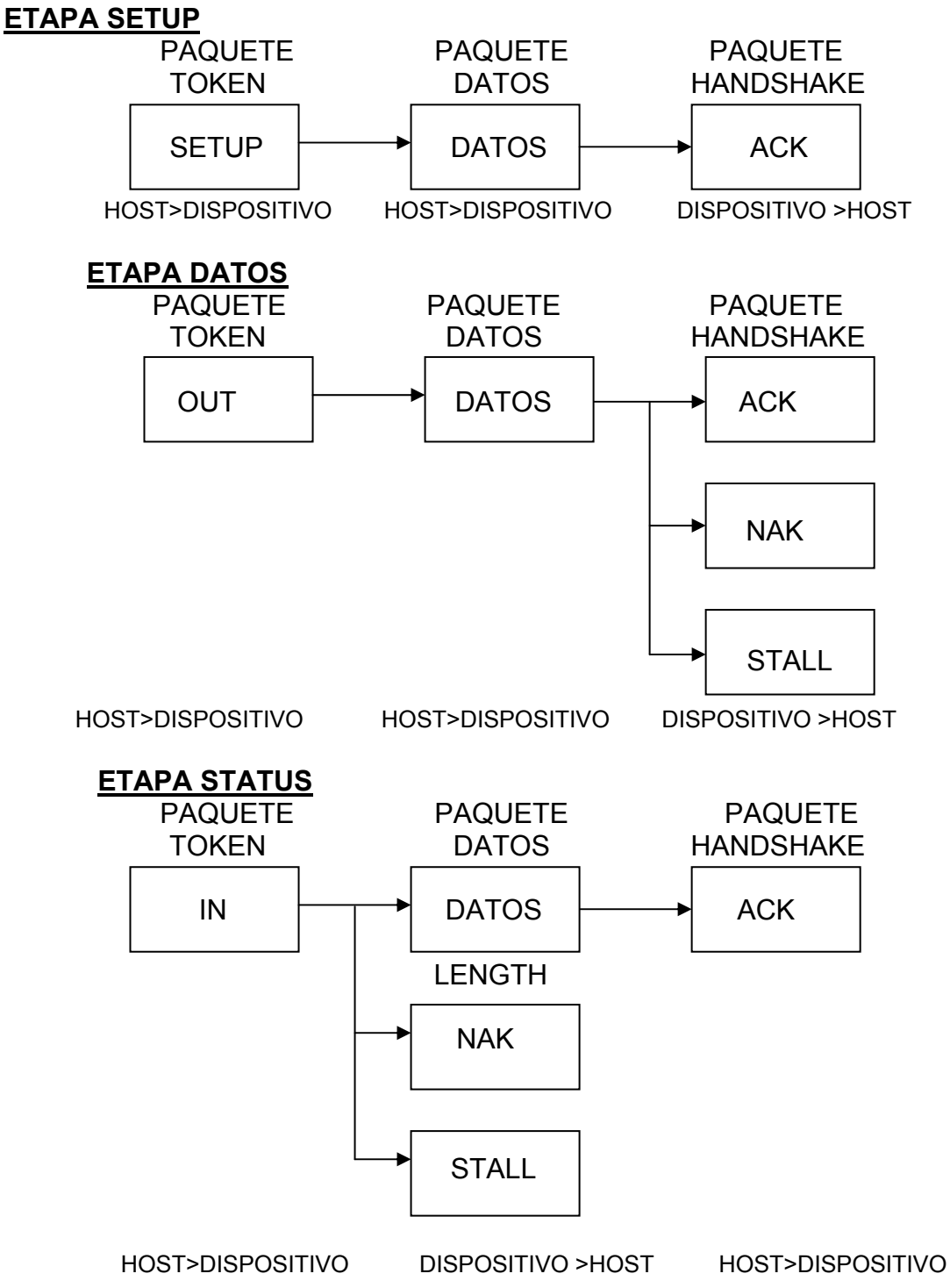


Figura 2.18 Transferencia de escritura USB

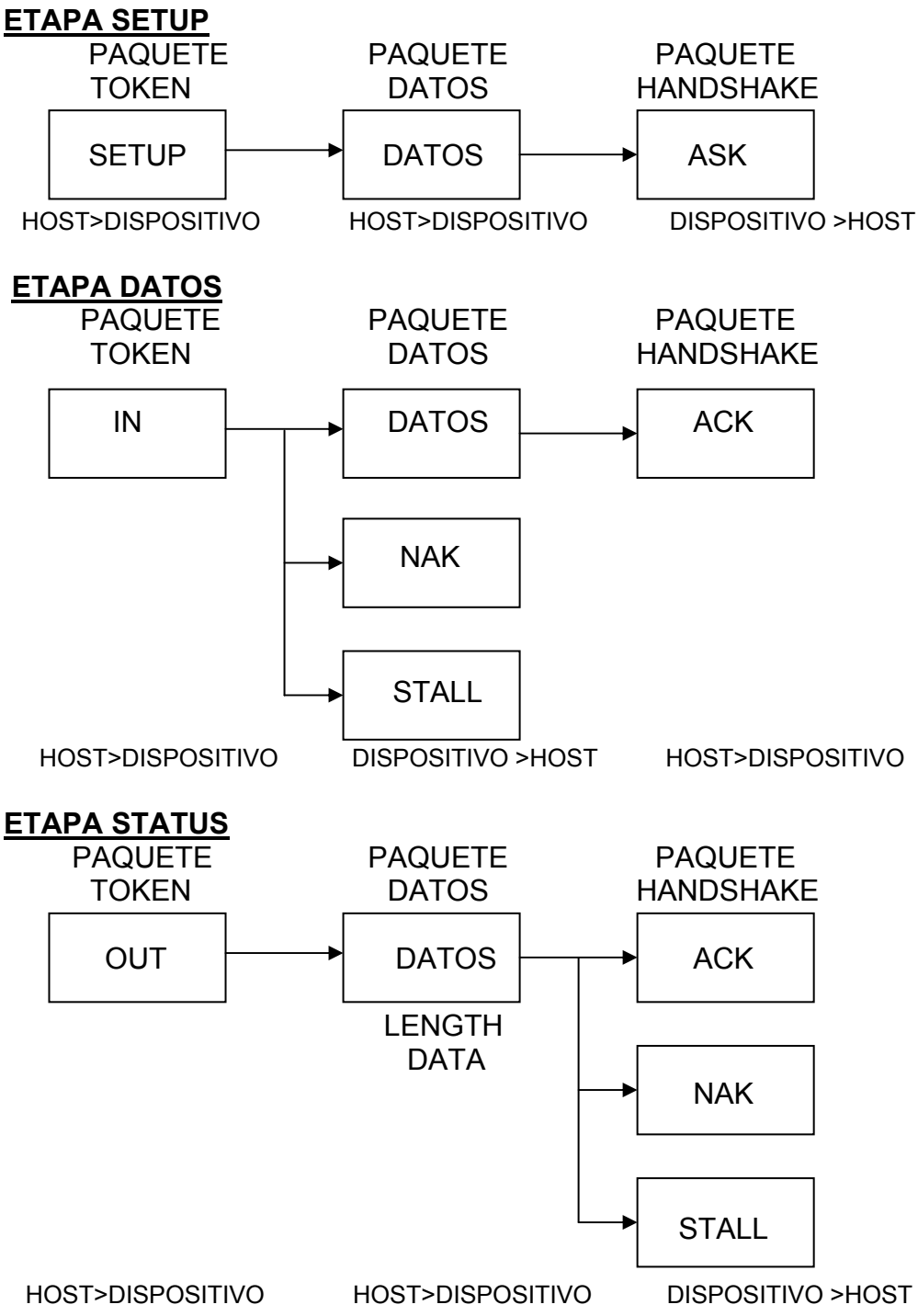


Figura 2.19 Transferencia de lectura USB

00000: el dispositivo.

00001: una interfaz específica.

00010: Endpoint.

00011: Otro elemento en el dispositivo.

bRequest: Es un byte que específica la petición, teniendo este un único valor dependiendo de del tipo de petición bmRequestType<6,5>.

00: bRequest específica una de las peticiones estándar USB.

01: bRequest específica una petición definida por una clase del dispositivo.

00: bRequest específica una petición definida por el fabricante

wValue: Son dos Bytes que utiliza el host para pasar información a el dispositivo, el contenido de estos Bytes dependerá del tipo de petición.

wIndex: Son dos Bytes que utiliza el host para pasar información a el dispositivo. Un uso específico es para un índice u offset tales como el número de interfaz o endpoint.

Como un índice de un endpoint

Windex<0-3>: Indica el número del endpoint.

Windex<7=0>: Endpoint de control o OUT.

Windex<7=1>: Endpoint IN.

Como un índice de una interfaz

Windex<0-7>: Número de la interfaz.

wLength: En estos bytes se almacena el número de bytes de datos en la etapa de transacción de datos. Para una transferencia de escritura. Wlength tiene el número exacto de bytes que el host quiere transferir. En cambio para una transferencia de lectura Wlength es el máximo valor y el dispositivo puede retornar este número de bytes o menos, en cambio si este cambio esta en cero significa que no hay datos

HANDSHAKE

Propósito: Transmitir el reconocimiento del dispositivo.

Fuente: Dispositivo.

PID: ACK

Contenidos adicionales: Consiste únicamente del PID.

Si el dispositivo detecta error en la recepción del SETUP o DATA este no retornará el handshake

Etapa Datos

Cuando ocurre este tipo de transacción pueden estar compuestas por una o más del tipo IN o OUT. El número máximo de bytes en una transacción en el endpoint0 esta especificado por el descriptor del dispositivo.

Cuando la transacción es del tipo de IN los datos viajan desde el dispositivo al host, por ejemplo cuando con la petición Get_Descriptor donde se le envía el descriptor solicitado por el host. Por otro lado, en la transacción OUT se pasan datos desde el host hacia el dispositivo, un ejemplo de esto es la petición de la clase HID, Set_Report, en donde el dispositivo lleva un reporte

a un dispositivo. En cambio cuando en el campo wLength es cero, no se lleva a cabo la etapa de datos, en la petición de Set_Configuration el host pasa el valor de la configuración al periférico en wValue de la etapa de setup en el paquete de datos, este es un ejemplo cuando no ocurre la transacción de datos.

Si se requiere enviar una mayor cantidad de datos que pueda envíarse en una sola transacción, se podrá entonces realizar varias de estas transacciones. El número de transacciones requeridas se la obtiene dividiendo entre el valor que contiene wLength y wMaxPacketSize en el descriptor del endpoint. Por ejemplo cuando en la petición Get_Descritor, si wLength es 18 y wMaxPacketSize es 8, entonces tendremos 3 transacciones de datos.

La transacción SPLIT en la etapa de datos se utiliza cuando el dispositivo es low speed o full speed, aún si está conectado a un bus del tipo high speed. Para dispositivos con high speed tenemos el protocolo PING.

TOKEN

Propósito: Identificar el receptor y la transacción como IN o OUT.

Fuente: HOST

PID: Si se solicita enviar datos al host, el PID es IN. Pero si los datos se dirigen hacia el dispositivo el PID será OUT.

Contenidos adicionales: dirección del endpoint y dispositivo.

DATA

Propósito: Transmitir los todos los datos o parte de estos especificados en el campo wLength de la transacción setup (paquete de datos).

Fuente: Dependiendo del PID del paquete token los datos se enviarán desde el host hasta el dispositivo o en el sentido contrario.

PID: El primer dato es DATA1, al existir más de un datos, estos se alternan DATA0/ DATA1

Contenidos adicionales: El paquete de datos o el de cero datos.

HANDSHAKE

Propósito: El receptor de datos retorna información acerca del estatus.

Fuente: Puede ser tanto del dispositivo como el host, depende del PID del paquete token.

PID: Algunos dispositivos retornan ACK, NAK, STALL, NYET. El host solo puede retornar ASK.

Contenidos adicionales: Consiste únicamente del PID.

Si el receptor detecta error en los paquetes token o datos, este no retornará el handshake.

Etapa Status

En esta transacción es donde se detecta si tuvo existo o fallo la transferencia, no de una transacción. Este es similar a los paquetes denominados handshake, de hecho, en algunos casos (tal como sucede después de la recepción del primer paquete de un descriptor del dispositivo durante la enumeración) el host podría empezar una transacción de status antes de

finalizar otra de datos (sin ser completada), en el momento que se detecta el token del status.

TOKEN

Propósito: Identificar el receptor e indica la dirección de los paquetes de datos de esta transacción.

Fuente: HOST

PID: Es opuesta a la dirección de la anterior transacción de datos. Por ejemplo si la anterior etapa de datos tuvo un PID como OUT o no estuvo presente la misma, entonces el PID será IN.

Contenidos adicionales: dirección del endpoint y dispositivo.

DATA

Propósito: Habilita el receptor de datos de esta etapa para indicar el status de la transferencia.

Fuente: Si el PID es IN entonces el dispositivo enviará información al host, en cambio si el es OUT el host le enviará información al dispositivo.

PID: DATA1.

Contenidos adicionales: El host envía el paquete 0-length, En cambio el dispositivo puede enviar los paquetes 0-length (existo), NAK (ocupado), STALL (endpoint inhabilitado).

Para la mayoría de las peticiones, 0-length enviado por el dispositivo indica que la acción solicitada ha sido tomada. Una excepción es Set_Address, en

el cual el dispositivo implementa después que la etapa de status ha sido completada.

HANDSHAKE

Propósito: El que envía los datos (En la etapa de datos) indica el estatus de la transferencia.

Fuente: El receptor de los paquetes de datos en la etapa de status. Si el PID del paquete token es IN, entonces el host envía el paquete handshake. En cambio si el PID es OUT, entonces el dispositivo pasa el handshake al host.

PID: La respuesta del dispositivo podría ser ACK (existo), NAK (ocupado), STALL (la petición no es soportada o el endpoint esta inhabilitado). El host envía ASK si no hubo errores en los datos.

Contenidos adicionales: Consiste únicamente del PID.

Si el receptor detecta error en los paquetes token o datos, este no retornará el PID del handshake.

Control de errores

Los dispositivos no siempre podrán culminar las peticiones en la transferencia de control. El firmware no podría soportar la petición, o el dispositivo no podría estar habilitado para responder porque el firmware ha fallado o el endpoint esta en la condición Halt (inhabilitado).

Por estas condiciones, el dispositivo notifica al host enviando un STALL en el paquete handshake. El dispositivo puede enviar en primera instancia un

ACK, para luego enviar STALL en la transacción de datos o status. Logrando cancelar la transferencia en la que se encuentra.

Al ocurrir una falla el host vuelve a realizar la etapa de setup, pero al no poder el dispositivo recuperarse este tomará medidas mas drásticas tal como solicitarle al hub del dispositivo resetear su puerto.

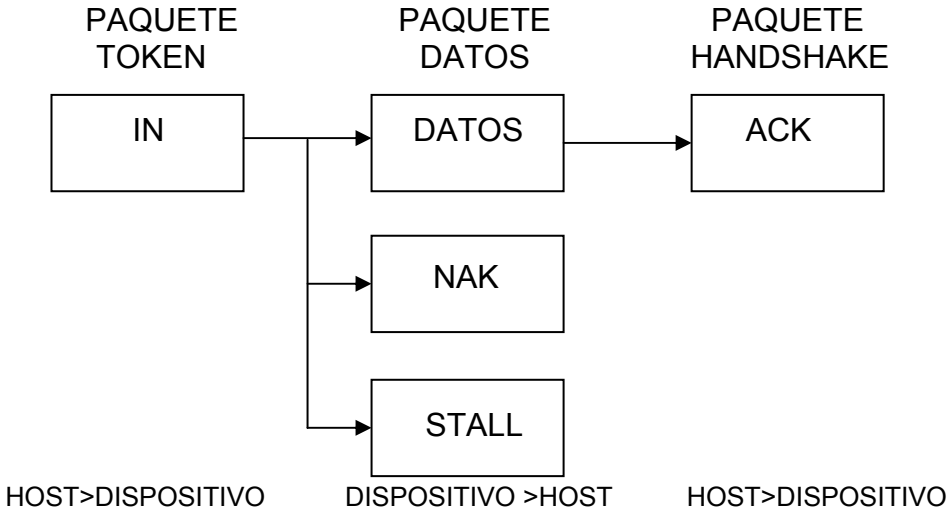
2.3.2.4 Transferencia BULK.

Todo dispositivo USB emplea la transferencia de control para el proceso de enumeración. Por otro lado la transferencia BULK es la escogida por nosotros para la comunicación con la aplicación, debido a que se necesita transferir grandes cantidades de información y el ancho de banda en el momento de descarga estará disponible.

La transferencia BULK es útil para transferir datos cuando el tiempo no es algo crítico, este tipo de transferencia puede enviar grandes cantidades de datos sin obstruir el bus USB debido a que espera hasta que otras transferencias ocurran, en el momento que el bus se encuentre libre, este tipo de transferencia puede ser muy rápido. El uso de transferencias BULK incluyen el envió de datos desde el host hasta una impresora, o desde el escáner a una PC, etc.

Una transferencia BULK consiste de una o más transacciones (IN ó OUT), las transacciones pueden ser todas del tipo IN o todas del tipo OUT, es decir, en una sola dirección, transferir datos en dos direcciones requiere pipes diferentes, uno para cada dirección.

Transacción IN



Transacción OUT

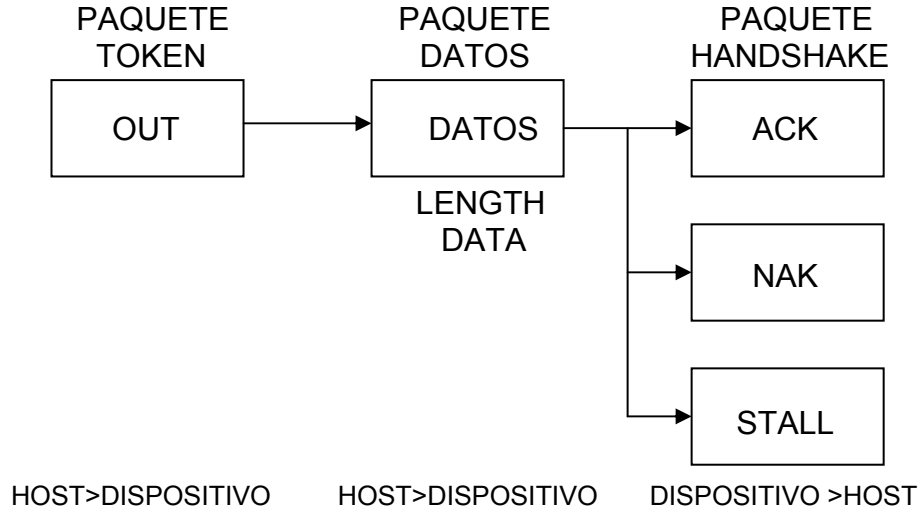


Figura 2.20 Transferencia BULK

Una transferencia BULK finaliza cuando una transacción contiene un número de bytes menor que el tamaño máximo del paquete de datos (endpoint). La especificación USB no define un protocolo para detallar la cantidad de datos

en una transferencia BULK sin embargo, cuando se lo requiera podrá usarse de una clase o fabricante específico para la transferencia de datos.

Este tipo de transferencia en modo Full Speed puede tener un tamaño máximo de 8, 16, 32 o 64 bytes. Para High Speed, el tamaño máximo es de 512 bytes. Durante la enumeración, el host lee el tamaño máximo de cada endpoint BULK del descriptor del dispositivo. Si la cantidad de datos a transferir es mayor que este tamaño máximo descrito anteriormente, este podrá ser enviado usando múltiples transferencias.

El controlador host garantiza que la transferencia se complete de manera paulatina, pero no reserva algún ancho de banda, por lo cual, si el bus se encuentra con otras actividades, la transferencia podrá tomar su tiempo. Por otro lado si el bus esta inactivo, la transferencia puede usar el ancho de banda que le asigne el host, siendo así uno de los más rápidos en transferir datos. La transferencia BULK puede transferir hasta 1216 bytes por frame, es decir 1216 megabytes/sec. dejando un 18% del ancho de banda del bus para otros usos.

Este tipo de transferencia utiliza un detector de error, si un dispositivo no retorna un paquete handshake, el host trata hasta dos veces más. El host también retorna en la recepción un NAK (paquete handshake). El driver del host se asegura que todos lo datos son recibidos sin error.

2.3.3 Enumeración

Antes que la aplicación pueda comunicarse con el dispositivo, el Host necesita aprender del dispositivo en el proceso de enumeración. Es en este proceso en donde se le asigna una dirección al dispositivo, se lee su descriptor, el Host asigna y carga un driver y se selecciona una configuración de los requerimientos de energía, Endpoint y otras características. A continuación se detalla una típica secuencia de eventos que ocurren durante este proceso.

2.3.3.1 Pasos de la Enumeración

1. **El dispositivo se vincula con el puerto USB.** Se energiza el dispositivo, puerto USB puede estar en el conectado a un Hub interno o un externo.
2. **El Hub detecta el dispositivo.** Esto lo logra mediante el monitoreo constante de las señales del puerto (D+, D-), El Hub posee resistencia pull down cada una de estas líneas, por otro lado el dispositivo tiene resistencias pull up en D+ si es Full Speed, y si es Low Speed esta resistencia esta en D-.
3. **El Host aprende del nuevo dispositivo.** Cada Hub usa su endpoint de interrupción para reportar un evento en el puerto. En el aprendizaje, el Host envía el Get_Port_Status para conocer algo más del evento. La información que retorna al Host le dice cuando el dispositivo es nuevamente vinculado.

4. **El Hub detecta si el dispositivo es Low Speed o Full Speed.** Justo antes el Hub resetee al dispositivo, el Hub determina si el dispositivo es Low o Full Speed examinando las líneas D+ y D-, verificando cuales de estas señales tiene un alto cuando el bus esta inhabilitado. Esta información es enviada por Hub hacia el Host en respuesta de un Get_Port_Status.
5. **El resetea el dispositivo.** El Host envía un Set_Port_Feature al Hub, solicitándole que coloque un nivel bajo en las líneas D+ y D- durante un período de 10 milisegundos.
6. **El Host detecta si un dispositivo Full Speed soporta High Speed.** Mientras se produce el reset de las señales D+ y D-, el dispositivo que soporta High Speed envía un estado denominado K, el Hub, que puede también soportar este modo, responde con estados K y J alternadamente. En la detección de las parejas KJKJKJ, el dispositivo remueve sus resistencia pull up (Full Speed) y realiza todo lo necesario para la comunicación High Speed. Si el Hub no responde, el periférico seguirá comunicándose en Full Speed.
7. **El Hub establece un enlace entre dispositivo y el Bus.** El Host solicita conocer si el dispositivo salio del estado de reset, para ello, envía una petición denominada Get_Port_status, si es necesario, El Host repetirá la petición hasta que el dispositivo haya salido del reset. Una vez fuera de este proceso, el dispositivo esta listo para responder

a la transferencia de control en el endpoint 0, La comunicación con el Host realiza usando la dirección 00h.

8. **El Host envía la petición Get_Descriptor para conocer el máximo tamaño de los bloques de información en el Pipe0.** El Host envía la petición a la dirección cero del dispositivo en el endpoint0. El Host solo puede enumerar a un dispositivo a la vez y el dispositivo responderá a la comunicación dirigida a la dirección 00h únicamente, inclusive si existen algunos dispositivos vinculados a la vez. enviando esta petición el Host comienza la etapa de Status de la transferencia.
9. **El Host asigna una dirección.** Aquí el controlador Host asigna una única dirección al dispositivo enviándolo por la petición Set_Address. Pero el dispositivo termina la etapa de status de la petición usando la dirección 00h e implementa la nueva dirección.
10. **El Host aprende de las habilidades del dispositivo.** El Host envía un Get_Descriptor una vez más a la nueva dirección para leer el descriptor del dispositivo.
11. **El Host asigna y carga un driver al dispositivo.** Luego del aprendizaje del dispositivo mediante sus descriptores. El host trata de obtener el driver mas apropiado para el periférico, para poder de esta manera establecer la comunicación con este. Esto se hace comparando información de los archivos INF del PC con datos obtenidos por el descriptor (Vendor ID, Product ID). Si no se logra

obtener de esta manera entonces, Windows trata de comparar el dispositivo con alguna clase o subclase. Si el dispositivo ha sido enumerado previamente, Windows puede buscar información en el sistema de registro en lugar de buscar los archivos INF.

12. **El Host selecciona una configuración.** Una vez que se aprende del dispositivo mediante sus descriptores, el driver solicita una configuración enviando la petición Set_Configuration con la información del número de configuración. Algunos dispositivos pueden soportar una sola configuración, pero existen otros que soportan múltiples configuraciones. En este caso, el driver puede decidir con cual se queda dependiendo de la información que posea acerca de cómo el dispositivo será utilizado. O el driver puede solicitar al usuario que hacer, o simplemente escoge la primera configuración. El dispositivo lee la petición y habilita la configuración solicitada, entrando así al estado de configuración y habilitando la interfaz del dispositivo.

2.3.3.2 Descriptores

Los descriptores son estructuras de datos o bloques de información que utiliza el Host para aprender del dispositivo. Todo dispositivo USB debe responder a las peticiones estándares USB. El dispositivo debe almacenar la información en el descriptor y responder peticiones por el descriptor.

A continuación se detalla los descriptores más comunes:

Device Descriptor.

Este contiene información básica del dispositivo. Este descriptor es el primero que lee el Host, enviando la petición Get_Descriptor, el descriptor incluye información del dispositivo, su configuración, su clase. Están conformados por los campos que a continuación nombramos, y los mismos que ocurren con la secuencia colocada.

bLength. El tamaño en bytes del descriptor.

bescriptorType. El tipo de descriptor (para el dispositivo es 01h).

bcdUSB. Versión del USB en código BCD.

bDeviceClass. Específica la clase del dispositivo, estos tienen valores reservados desde 1 hasta FEh, FF es específica del vendor y definida por el vendor.

bDeviceSubclass. Este campo específica una subclase dentro de un grupo (clase).

bDeviceProtocol. Específica el un protocolo definido por una clase o subclase.

bMaxPacketSize0. Es el tamaño máximo de paquetes del Endpoint0 (Low Speed =8; Full Speed=8, 16, 32, 64; High Speed= 64).

idVendor. Vendor ID es un único identificador de los fabricantes de productos USB.

idProduct. El Propietario del vendor ID asigna a un dispositivo un Product ID para diferenciarlos.

bcdDevice. Es un valor en formato BCD del dispositivo, este es asignado por el fabricante.

iManufacturer. Es un índice que apunta a describir el fabricante. Puede ser cero si no existe el descriptor del fabricante.

iProduct. Este también es un índice que describe al producto. Puede tomar el valor de cero si no existe el string decriptor.

iSerialNumer. Contiene el número de serie para de esta forma diferenciarse de otros dispositivos.

bNumConfigurations. El número de configuraciones que soporta el dispositivo.

Configuration Descriptor.

Luego que el Host obtiene información del Device Descriptor, el Host puede solicitar los descriptores de la configuración, interfaz y endpoint. Cada dispositivo puede tener una o más configuraciones, las mismas que especifican las características y habilidades del dispositivo, tal como los requerimientos de energía y el número de interfases que soporta. Este descriptor se conforma por los siguientes campos:

bLength. Es el tamaño de Bytes del descriptor.

bDescriptorType. El tipo de descriptor (Configuración es 02h).

wTotalLength. El número de Bytes del descriptor y todos sus sub-descriptor.

bNumInterfaz. El número de interfases de una determinada configuración, mínimo es uno.

bConfiguractionValue. Identifica la configuración mediante las peticiones Get_Configuration y Set_Configuration.

iConfiguration. Es un arreglo que describe la configuración, este valor es cero si no hay el String Descriptor.

bmAttributes. Si el bit 6=1, entonces el dispositivo tiene su propia alimentación, en cambio si toma el valor de cero quiere decir que se alimenta del Bus. El bit 5 indica si el dispositivo tiene características remotas para salir del estado de suspendido. Entra en este estado si no hay actividad durante 3 ms. Los demás bit no son usados. El bit 7 debe tomar el valor de 1(en la versión 1.0 este bit indicaba que la energía la tomaba del bus).

bMaxPower. Específica el consumo de corriente del dispositivo, este valor es la mitad del número en miliamperios, por ejemplo si el periférico requiere 200 mA. el valor que tomará este byte será de 100. La máxima corriente que el dispositivo puede demandar es 500mA.

Interfaz Descriptor.

Está conformado por nueve campos, contiene información de la clase, subclases y protocolo, y el número de endpoint que la interfaz utiliza. Una configuración puede tener algunas interfases activas a la vez. Pueden estar asociadas a una misma función como también no pueden estar relacionadas.

bLength. Es el tamaño de Bytes del descriptor.

bDescriptorType. El tipo de descriptor (Configuración es 04h).

bInterfazNumber. Identifica la interfaz, si existe algunas interfases, cada una de estas tendrán sus propios descriptores.

bAlternatingSetting. Cuando la configuración soporta múltiple interfases el descriptor tiene el mismo valor en bInterfazNumber y un único valor en bAlternatingSetting

bNumEndpoints. Contiene el número de endpoint que la interfaz utiliza, sin contar el endpoint 0, los dispositivos que solo soportan el endpoint 0 este campo contiene el valor cero.

bInterfazClass. Específica la clase de la interfaz.

bInterfazSubclass. Específica una subclase dentro de una clase.

bInterfazProtocol. Describe el protocolo para una clase o subclase definida anteriormente.

iInterfaz. Es un índice que describe la interfaz, puede tomar este campo el valor de cero si el string descriptor no existe.

Endpoint Descriptor.

Cada endpoint especificada en el Interfaz Descriptor tiene su descriptor, a excepción del endpoint0. El Host obtiene este descriptor solicitando al Configuration Descriptor la configuración del endpoint. A continuación se detalla los seis campos que lo conforma.

bLength. Es el tamaño de Bytes del descriptor.

bDescriptorType. El tipo de descriptor (Endpoint es 05h).

bEndpointAddress. Contiene la dirección y el número de endpoint, Los Bit 0 hasta 3 contienen la dirección del endpoint, mientras si el Bit 7 = 0 entonces es de tipo OUT, por otro lado si este Bit toma el valor de 0 entonces su dirección es del tipo IN.

bmAttributes. Los Bit 1y 0 especifican el tipo de transferencia, 00 es la transferencia de control, 01 es Isocronica, 10 es Bulk y por último 11 es del tipo interrupción. Si la transferencia es isocronica los bits 2 hasta el 7 son utilizados, de otro modo deben ser cero.

wMaxPacketSize. Es el número máximo de Bytes que el endpoint puede transferir en una transacción.

bInterval. Puede indicar la máxima latencia de los indicadores de las interrupciones del endpoint o los máximos rangos del NAK.

String Descriptor. Contiene índices o caracteres que describen a los fabricantes, al producto, número de serie, configuración, interfaz. Los campos se detallan enseguida:

bLength. Es el tamaño de Bytes del descriptor.

bDescriptorType. El tipo de descriptor (String es 03h).

wLANGID. Es Usado únicamente en el descriptor0. Cuando el Host solicita el String Descriptor, el byte menos significativo del campo wValue es un valor índice. Un valor de cero representa una función especial, la cual solicita una identificación del lenguaje.

bString. Cualquier otro valor en el índice puede contener caracteres, tal como el nombre del fabricante.

2.3.4 Descripción del CDC

2.3.4.1 Introducción

La interfaz serial RS-232 ya no es un puerto muy común en las computadoras personales, este es un problema para aplicaciones que involucran el uso de este estándar para comunicarse con los sistemas externos. Una solución es migrar a aplicaciones con interfaz USB, existen algunos métodos para convertir una interfaz RS-232 a una USB, cada una requiere diferentes niveles de experiencia. Una manera sencilla de lograrlo es emular al RS-232 sobre el bus USB, logrando de esta manera una ventaja que consiste en que la aplicación reconocerá la conexión USB como un puerto RS-232, evitando al escoger este camino realizar cambios al existente software.

2.3.4.2 especificación CDC

La especificación CDC (Communication Device Class) define algunos modelos de comunicaciones, incluido la emulación del puerto serie. La especificación CDC describe un modelo de control abstracto para la simulación serial sobre el bus USB. Se requiere dos interfases, la primera es CCI (Communication Class Interfaz), la misma que utiliza un endpoint de interrupción del tipo IN, esta interfaz es usada para notificar al host USB del

estado de una conexión RS-232 desde un dispositivo emulado. La segunda es DCI (Data Class Interfaz), la cual utiliza un endpoint bulk del tipo IN otro del tipo OUT, esta interfaz es utilizada para transferir datos que normalmente serían transferidos sobre un puerto real RS-232.

Peticiones estándares.

Las peticiones estándares para el modelo de control abstracto son las siguientes:

SEND_ENCAPSULATED_COMMAND. Emite un comando en el formato del protocolo de control soportado.

GET_ENCAPSULATED_RESPONSE. Solicita una respuesta en el formato del protocolo de control soportado.

SET_LINE_CODING. Configura el rango del DTE (equipo Terminal de datos, por ejemplo el PC), bit de parada, paridad etc.

GET_LINE_CODING. Solicita el actual rango del DTE (equipo Terminal de datos), bit de parada, paridad etc.

SET_CONTROL_LINE_STATE. Señal del RS-232 usada para decirle al dispositivo DCE (Tarjeta de adquisición) que el dispositivo DTE esta presente.

2.4 Control de Temperatura.

2.4.1 Sensores de temperatura.

Muchos procesos industriales requieren el control preciso de la temperatura para producir resultados de calidad o prevenir sobrecalentamientos, rupturas, explosiones y otros tipos de problemas. Como por ejemplo, se necesitan altas temperaturas para ablandar materiales tales como el plástico o metales, como también se es necesario mantener temperaturas muy bajas para conservar productos como sucede en las fábricas procesadoras de alimentos. De otro lado la condición de sobrecalentamiento se puede dar en un sistema de lazo cerrado como en las calderas, en donde puede provocar una excesiva presión. Actualmente se disponen de sensores para dicho fin, los mismos que nombramos a continuación

Termostato. Son sensores del tipo ON-OFF que conmutan automáticamente de un estado al otro cuando la temperatura a su alrededor alcanza un valor determinado. Constan generalmente de un bimetal, es decir una pieza formada por dos metales con distinto coeficiente de dilatación térmica. Este tipo de sensores se fabrican para detectar temperaturas desde -75°C hasta +540°C y son también muy utilizados como dispositivos de protección en circuitos eléctricos, como por ejemplo los relés térmicos.

Dispositivos termo-resistivos (RTD). Son dispositivos basados en la variación normal que experimenta la resistencia de un conductor metálico puro con la temperatura, como resultado del cambio de su resistividad y sus

dimensiones. Esta variación es directa, es decir, que si la temperatura aumenta o disminuye, la resistencia también aumenta o disminuye en la misma proporción. Se dice entonces, que son dispositivos con coeficiente de temperatura positivo (PTC). El elemento sensor es típicamente un alambre fino de platino o un delgada película del mismo material aplicada a un sustrato cerámico, otros materiales utilizados pueden ser el níquel, el cobre y el molibdeno. Las RTD, principalmente las versiones de platino, se caracterizan por su precisión y su amplio rango de temperaturas de operación, el cual se extiende desde -250°C hasta +850°C. Tienen también una sensibilidad, estabilidad y repetibilidad muy altas, y ofrecen una respuesta más lineal que las termocuplas. Las RTD se utilizan generalmente con acondicionadores de señal que convierte su salida a un voltaje o a una corriente proporcional a la temperatura. Esta señal de alto nivel puede ser entonces transmitida a un sistema de control.

Termistores. Este tipo de sensores son basados en óxidos metálicos semiconductores que exhiben un gran cambio en su resistencia eléctrica cuando se comenten a cambios relativamente pequeños de temperatura. Pueden ser de coeficiente de temperatura positivo (PTC) o negativos (NTC), siendo estos últimos los más utilizados. Los termistores PTC se construyen a base de óxidos de bario y titanio, y lo NTC a base de óxidos de hierro, cobre, cromo, cobalto, manganeso y níquel dopados con iones de titanio o litio. Los termistores pueden tomar una gran variedad de formas y tamaños, llegando

incluso a ser tan diminuto como la cabeza de un alfiler. La mayoría de termistores se diseñan para trabajar en el rango de -50 °C. a 150 °C. Los termistores ofrecen varias ventajas con respecto a las RTD y las termocuplas. Por ejemplo, son más económicas, estables y confiables, pueden hacerse lo suficientemente pequeñas para permitir la medición puntual. Los termistores son particularmente adecuados para aplicaciones de baja temperatura sobre rangos limitados.

Termocuplas. O termopares son transductores de temperatura constituidos por dos alambres conductores hechos de metales diferentes y soldados por uno de sus extremos formando una unión. Al calentar dicha unión se produce entre los extremos de la termocupla (unión fría) un voltaje proporcional a la diferencia de temperatura entre la unión caliente y cualquiera de las uniones frías, las cuales deben estar a una misma temperatura de referencia, generalmente 0°C. Estos elementos se fabrican a base de metales o aleaciones metálicas especiales, como platino, hierro, cobre, rodio, renio, etc. Dependiendo de la combinación o calibración particular de metales utilizados, la termocuplas reciben diferentes nombres o designaciones (J, K, T, E, etc.)

Sensores de temperatura infrarrojos. También denominados pirómetros de radiación, son dispositivos de no contacto que miden la temperatura de cuerpos calientes a partir de la radiación térmica emitida en forma natural por los mismos. Se utiliza en los casos en los cuales resulta imposible o peligroso el uso de un termistor, una termocupla u otro tipo de sensor de

contacto. Por ejemplo en el caso de procesos industriales donde se manejan temperaturas muy superiores a las de fusión del transductor, de cuerpos calientes muy pequeños, inaccesibles o en movimiento. Estos sensores están basados en el concepto de que todos los cuerpos s temperaturas superiores al cero absoluto (-273°C), producen radiación térmica en cantidad dependiente de su temperatura y sus propiedades físicas. Esta energía se incrementa a medida que el objeto se torna mas caliente.

Sensores de temperatura de silicio. Los sensores de silicio son circuitos integrados que aprovechan la variación predecible del voltaje de la unión base-emisor (V_{BE}) de los transistores bipolares para realizar mediciones confiables y exactas de temperatura. Se caracterizan por su pequeño tamaño y son especialmente apropiados para aplicaciones de medición y control de temperatura en el rango de -55°C a +150°C. Además, no requieren de etapas de linealización, amplificación ni compensación externas debido a que incorporan en la misma pastilla sus propios circuitos de procesamiento de señales. La mayoría de sensores de silicio proporcionan como salida un voltaje que varia linealmente con la temperatura en grados Kelvin (°K), Celsius (°C) o Fahrenheit (°F)

2.4.2 Tipos de Control.

Los sistemas de control pueden ser clasificados básicamente en sistemas de lazo abierto o manual y lazo cerrado o automático. Un sistema de control es de lazo abierto cuando la variable de salida no afecta la acción de control. En

otras palabras, no se mide la salida (variable controlada) ni se realimenta para compararla con la entrada (señal de referencia). Por tanto, no se utiliza sensores y a cada entrada de referencia le corresponde una condición operativa fija. Como resultado, la precisión del sistema depende de la calibración. Ante la presencia perturbaciones, el mismo no realiza eficientemente la tarea deseada. Se utilizan en aplicaciones relativamente sencillas, donde no se requiere mucha exactitud, no hay perturbaciones internas ni externas, o existe una relación conocida entre la entrada y la salida.

Un sistema es de lazo cerrado cuando utiliza elementos adicionales para medir o sensar la variable de salida, realimentarla procesada a la entrada, compararla con la señal de referencia y utilizar la diferencia (señal de error) para obtener la respuesta de salida deseada. Por tanto, mantiene una relación preestablecida entre la salida y la entrada, y es capaz de auto corregirse en caso de una perturbación externa. La señal de retroalimentación es producida generalmente a partir de un sensor que mide la señal de salida y la convierte en una señal adecuada para ser procesada por el controlador del sistema y comparada con la señal de referencia. Este tipo de control es el más utilizado en control industrial. El controlador de los sistemas de lazo cerrado pueden ser del tipo PI, PID, ON-OFF, entre otros. Un controlador PID tienes tres acciones denominadas Proporcional, Integral y Derivativa. La primera acción se caracteriza porque la señal de control

depende proporcionalmente de la señal de error. La acción anterior siempre presenta error por corregir, la acción integral permite anular este error. En cambio, la acción derivativa ayuda a obtener una respuesta dinámica más rápida. Los controladores PI son útiles en donde el proceso no requiere un tiempo de respuesta rápido. Por otro lado un controlador ON-OFF es aquel donde solo se pude obtener dos posibles valores (máximo y salida). En estos sistemas de control las entradas tienen magnitudes continuas y las salidas son del tipo lógico.

CAPITULO 3

3 Diseño del Sistema

3.1 Descripción general del sistema.

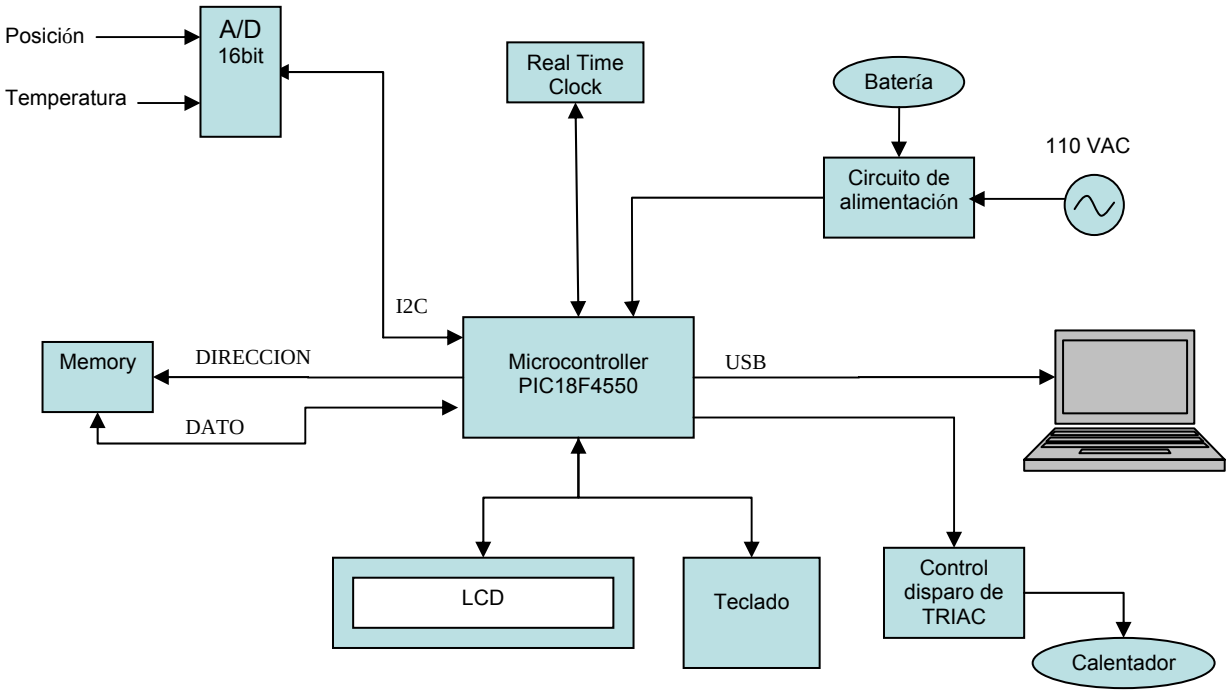


Figura 3.1 Diagrama de bloques del sistema

El sistema (Fig. 3.1) esta compuesto por un módulo de adquisición de datos, el cual será el encargado de obtener las variables del ensayo (temperatura, deformación del espécimen). Posee un sistema de control de temperatura proporcional e integral para el disparo del TRIAC para evitar tener

oscilaciones grandes como se observa en control ON-OFF. Los datos obtenidos de la prueba son almacenados en memorias del tipo RAM, para luego transferirlos mediante tecnología USB al computador. El sistema posee un reloj de tiempo real (RTC) para obtener la hora y fecha de la toma de la muestra que se comunicara con el microcontrolador mediante I2C. Posee un sistema de transferencia de energía eléctrica, que en el momento de falle el suministro de energía general pueda trabajar con una batería y evitar la pérdida de datos. Además el controlador del sistema es implementado por un microcontrolador de la familia 18 de Microchip (PIC18F4550). Para la descarga y configuración de los parámetros se lo realiza a través de una aplicación que fue desarrollada en Visual Basic. Finalmente esta dotado por un visualizador para observar en tiempo real los datos muestreados.

3.1.1 Diagrama de flujo del código fuente del microcontrolador (firmware).

El primer diagrama de flujo ha analizar es el general (Fig. 3.2), este comienza con la inicialización de los módulos internos del microcontrolador, que nos es otra cosa que la habilitación y configuración de los periféricos como los temporizadores (Timer 1 y Timer 2), interrupciones, convertidor A/D interno, etc., seguido después por la inicialización del módulo USB del PIC18F4550, RTC y los convertidores A/D, para finalmente entrar a un lazo infinito.

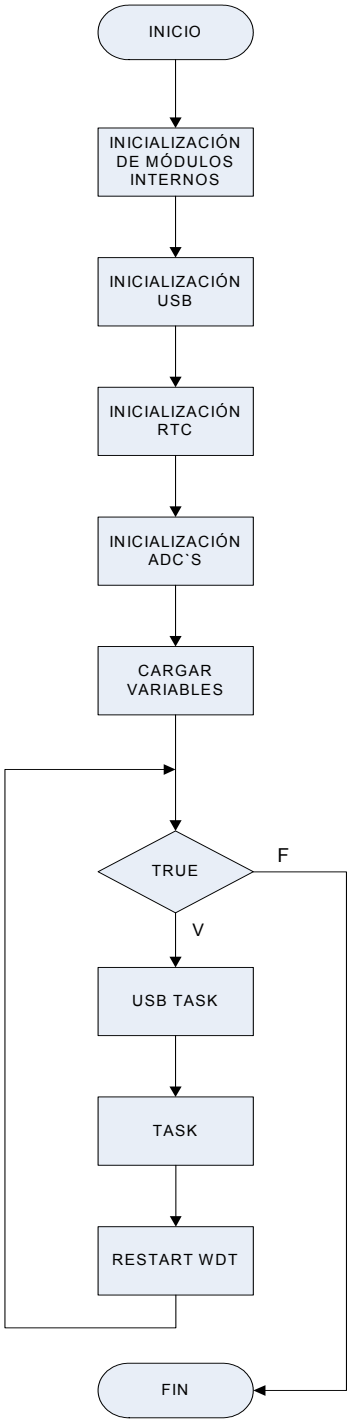


Figura 3.2 Diagrama de flujo general

La primera acción dentro del lazo infinito es chequear si existe conexión entre el dispositivo y el computador. Luego de esto realiza las tareas propias del dispositivo que mas adelante se detallan, finalmente se concluye reiniciando el perro guardián y regresando al inicio del lazo.

En el segundo diagrama de flujo (TASK), como se muestra en la Fig. 3.3, es donde se realizan las tareas que realizará el microcontrolador, cabe resaltar que cada una de ellas se ejecutará cada cierto lapso de tiempo, para esto se ha empleado las características del módulo Timer 2(pre-escalador y post-escalador). Cada tarea posee una bandera que se activará cuando el temporizador llegue a un valor predispuesto para dicha tarea, para finalmente luego de la ejecución del proceso se limpiará.

Antes de iniciar una tarea se utiliza una condicionante para verificar si es el momento para su ejecución, esta condicionante es la bandera descrita anteriormente, por ejemplo si TASK_F1 es verdadera se procesará la tarea 1 (TASK1) de lo contrario preguntará por la siguiente condicionante (TASK_F2) y así sucesivamente.

TAREA	**DESCRIPCION**	**TAREA**	**DESCRIPCION**
TASK1	Ejecución de pantalla	TASK4	Lectura del ADC de longitud
TASK2	Actualización de reloj	TASK5	Lectura del ADC de temperatura
TASK3	Transferencia de datos por USB	TASK6	Estado de batería

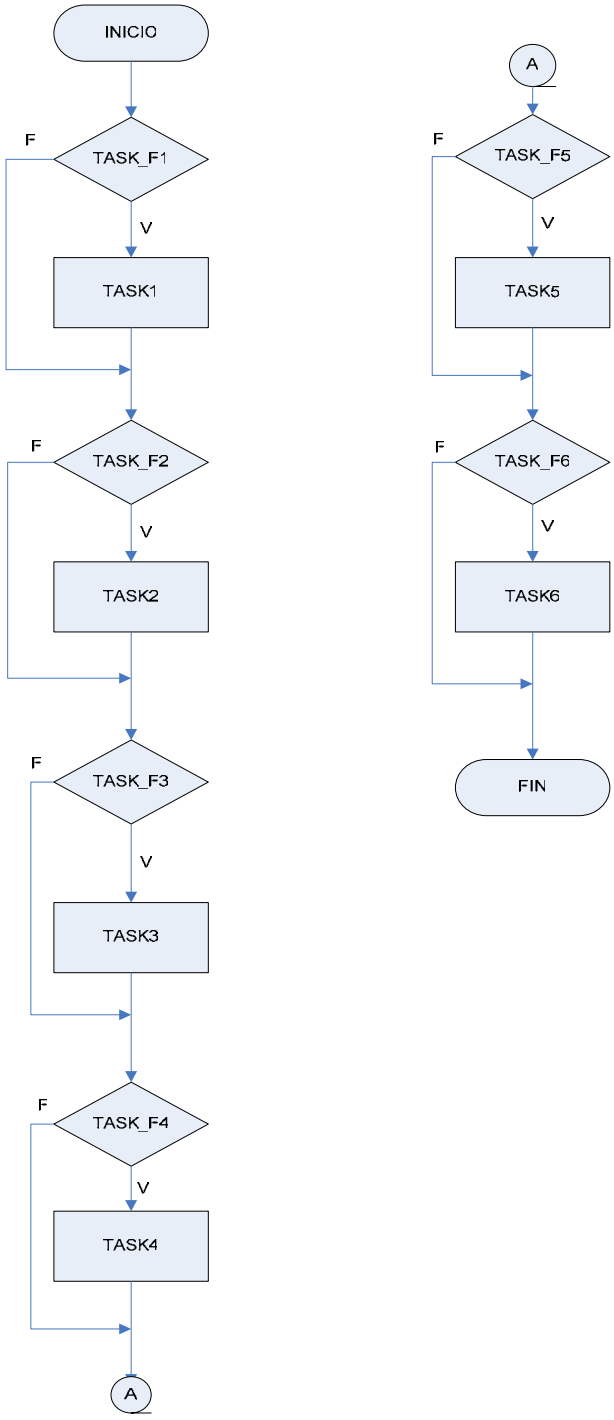


Figura 3.3 Diagrama de flujo de TASK

La función USB TASK que se ejecuta antes de TASK no se detalla debido a que es un proceso propio del compilador utilizado (CCS). A continuación se

detalla cada una de las tareas realizadas por la unidad micro-controladora con su respectivo diagrama de flujo.

TASK1 (Fig. 3.4) es la encargada de ejecutar la pantalla LCD, para dicho propósito se inicia escaneando el estado del teclado, luego de esto controla los indicadores de estado (LED`s de RUN, STOP, Batería baja, falta de energía externa). El último proceso de este diagrama es el encargado de presentar las diferentes pantallas que serán observadas por el usuario, y dependerá del estado del teclado escaneado con anterioridad.

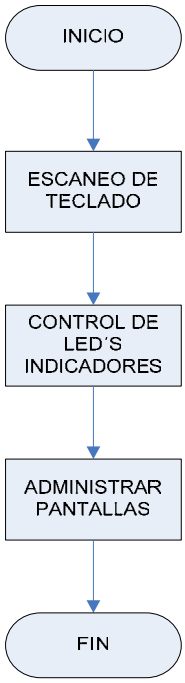


Figura 3.4 Diagrama de flujo de TASK 1

La figura Fig. 3.5 describe a TASK2, la tarea comienza escaneando los datos del reloj de tiempo real (RTC) tales como fecha y hora para colocarlos sus

respectivas variables, quedando listas estas mismas variables que luego las tomará la tarea1 para presentarlas en pantalla.

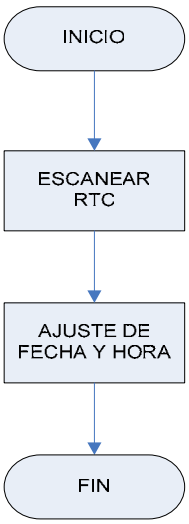


Figura 3.5 Diagrama de flujo de TASK 2

TASK3 (Fig. 3.6) es el encargado de la transferencia de datos por el bus USB, inicia preguntando si existe una conexión en el bus, esto lo realiza la función interna del compilador usb_cdc_connected(), luego de verificar la conexión los datos que se encuentren en el endpoint serán capturados, estos datos son comandos que determinaran de que manera responderá el módulo

Comando	Descripción	Comando	Descripción
Comando A	Actualizar pantalla	Comando C	Configuración de ensayo
Comando H	Actualizar fecha y hora	Comando T	Puesta en marcha o parada del ensayo
Comando E	Exportar datos		

a la aplicación. Por otro lado al no haber conexión entre el módulo y el ordenador saldrá de la tarea tres. En el cuadro siguiente se describe cada uno de estos comandos.

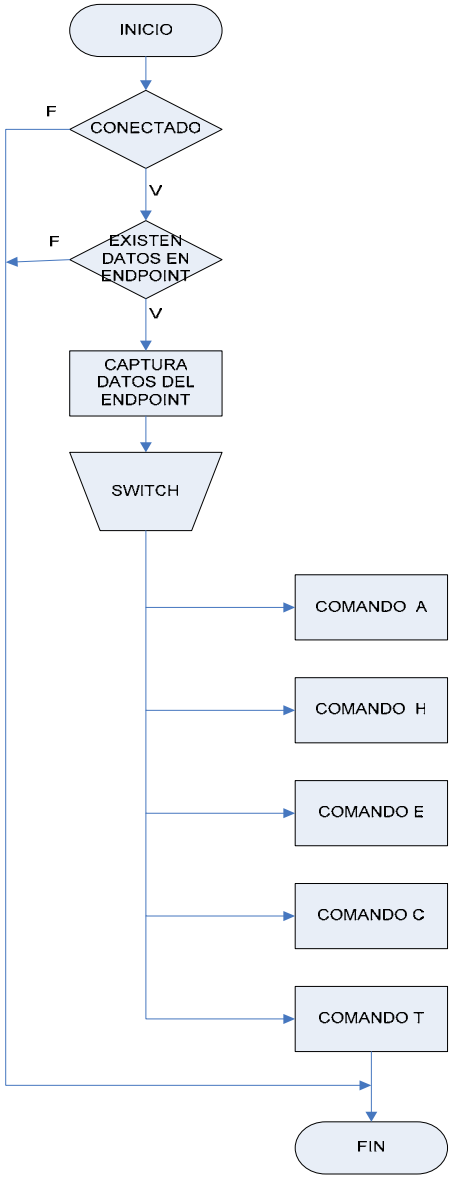


Figura 3.6 Diagrama de flujo de TASK 3

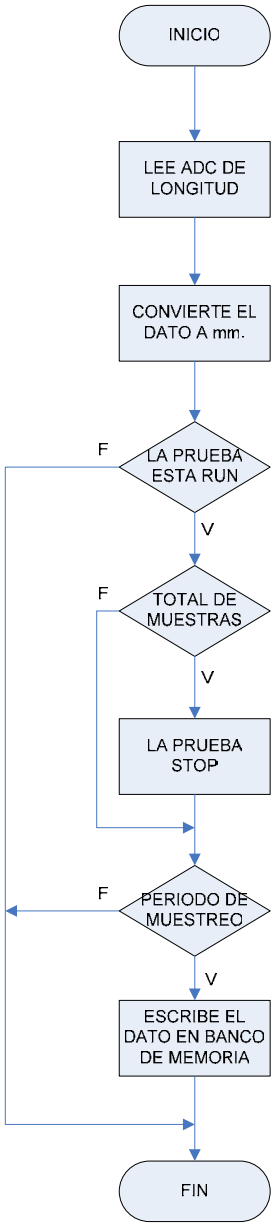


Figura 3.7 Diagrama de flujo de TASK 4

TASK4 es la encargada de leer el convertidor analógico-digital para luego convertir ese dato binario en un valor en milímetros que sería la longitud de la muestra, luego que esto pasa, si la prueba esta en marcha se verifica que sí

se han tomado todas las muestras configuradas en el ensayo, de ser así la adquisición de datos termina, de lo contrario se comprueba el tiempo que se almacenará los datos, esto simplemente es un contador que al llegar a un valor que el usuario define se podrá almacenar todos los parámetros de la prueba. Si el tiempo de muestreo no se cumple no se escribirá en memoria y saldrá de la tarea como lo demuestra la Fig. 3.7.

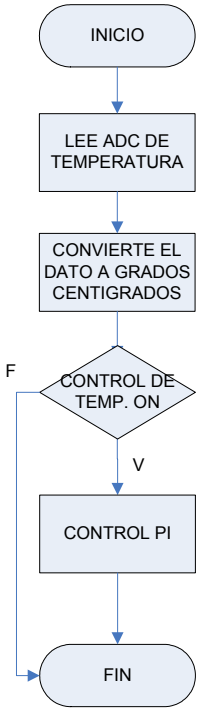


Figura 3.8 Diagrama de flujo de TASK 5

TASK5 (Fig. 3.8) en cambio lee el dato binario de temperatura, de igual manera que la tarea anterior, convierte este dato en grados centígrados, si el control de temperatura esta activado ejecutará un algoritmo destinado para

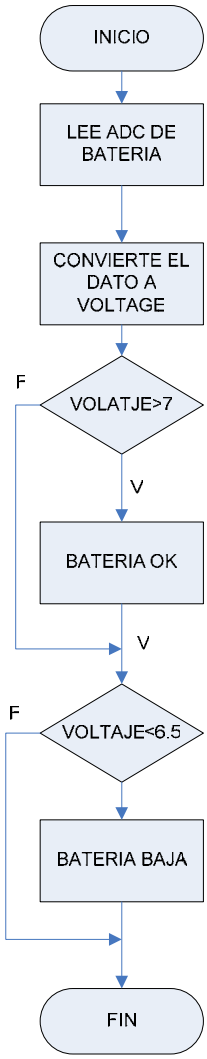


Figura 3.9 Diagrama de flujo de TASK 6

tal objetivo (control PI) y finaliza la tarea, De lo contrario luego de convertir el dato a grados centígrados finaliza la tarea. Cabe recalcar que los datos de temperatura no son escritos en la RAM, esto se realizará solo en la tarea 4.

La última tarea que realiza el microcontrolador (TASK6) es la de monitorear el nivel de voltaje, esto lo consigue leyendo el ADC interno que posee, para luego compararlo y así verificar su estado, produciendo una señal de

advertencia si este nivel esta por debajo de lo preestablecido como lo muestra la figura 3.9.

3.2 Módulo DATACREEP.

3.2.1 Características de adquisición de Datos.

Los parámetros de la prueba se obtienen mediante el módulo de adquisición de datos que esta conformado por dos conversores analógico-digital. Son conversores de precisión con entradas diferenciales de 16 bits de resolución. Poseen un amplificador de ganancia programable y un voltaje de referencia de 2.048V integrado en el chip. Además son compatibles con la interfaz serial I2C y pueden operar con suministro de energía desde 2.7V a 5.5 V

Uno de los conversores es utilizado para explorar la variable mecánica como es el caso de la longitud de la muestra durante el ensayo, la señal analógica la toma del sensor de posición, el cual es un potenciómetro con características lineales.

Para detectar la temperatura del ensayo se ha utilizado un dispositivo termoresistivo como es el caso de una RTD. Este sensor posee características casi lineales en el rango de 0 a 100 ºC. Un acondicionador de señal se ha acoplado después de la RTD, uno de los propósitos de esto es preparar la señal de entrada. Este sistema integrado inyecta una corriente constante de 1mA. al dispositivo termoresistivo para excitarlo y obtener un voltaje proporcional a la temperatura, este dispositivo tiene además integrado

un amplificador operacional con ganancia que es programable y una circuitería de salida para generan el rango de corriente de 4-20mA.

El segundo conversor de las características mencionadas anteriormente recibe la señal que produce el acondicionador, para realizar la conversión respectiva y enviar el dato en formato digital al microcontrolador. Cada conversor asume una dirección diferente que es la única forma de identificarse ante el dispositivo microcontrolador.

3.2.2 Características del control de temperatura.

Para el control de temperatura se escogió un control del tipo PI (proporcional integral), para poder entregar potencia de manera gradual, logrando tener menor oscilación que si estuviera funcionado como un control ON-OFF.

Esto se consigue modulando el tiempo que se mantiene el tiristor encendido, en este caso se tomo un tiempo de ciclo de 10 segundos, durante este ciclo la potencia variará de manera proporcional y se podrá eliminar el error de estado estacionario por la acción integral de este tipo de control.

3.2.3 Comunicación USB del módulo.

Como se citó anteriormente existen dos tipos de comunicación, la utilizada para el proceso de enumeración y para la transferencia de datos. Para realizar esto se ha utilizado una clase para la transferencia de los datos desde la tarjeta de adquisición hasta el PC, la cual es CDC y a la vez esta utiliza las dos interfases necesarias para la comunicación con el dispositivo,

para configurar el dispositivo se utiliza CCI y para la transferencia de los datos se utiliza DCI, Como ya se cito CCI tiene un formato definido y utiliza el endpoint0y DCI utiliza la transferencia bulk y un endpoint del tipo IN. Con esta clase obtenemos un puerto virtual (COM) de RS-232.

3.3 Programa de descarga de datos al P.C.

La aplicación se encargará de configurar los parámetros de la prueba, para esto el usuario tendrá que ingresar parámetros tales como el número de muestras, el tiempo de muestreo, el tiempo de duración del ensayo o si el control de temperatura se ejecutará o no.

Otra función que el programa realizará, una vez finalizada la toma de muestras, es de almacenar los datos que provienen del puerto USB, para finalmente colocarlos en una hoja de Excel para su posterior análisis.

3.3.1 Detalles de los eventos de la Interfaz gráfica de la aplicación.

La aplicación se desarrollo en Visual Basic 6.0, la misma que es una programación orientada a eventos, a diferencia de de otros programas como C en donde su ejecución es de manera estructurada o secuencial, en donde línea por línea de código es leída y ejecutada, en lugar de esto, nuestra aplicación realizará una actividad solo en el momento que ocurra un evento.

La aplicación esta compuesta por dos pantallas, la primera se podrá visualizar ciertos estados de la prueba, como por ejemplo, la temperatura, elongación de la muestra. La segunda pantalla se ingresara para configurar

los parámetros de la prueba. A continuación se detallarán cada uno de los eventos que conforman la ventana principal.

ActHF_Click.- Este evento es el encargado de actualizar el reloj de tiempo real del módulo DataCreep, para esto el programa le envía el comando “H” seguidamente de los parámetros de fecha y hora.

Cfg_test_Click.- Se visualiza la ventana de configuración de los parámetros del ensayo a partir de este evento.

selComm_Click.- Realiza la selección del Comm virtual al cual se conecta el módulo.

conect_Click.- Realiza la conexión o desconexión al Comm virtual, y a la vez habilita led indicadores de conexión y la ventana de configuración de parámetros, al no estar conectado no se podrá cambiar ningún parámetro de la prueba.

ExitCreep_click.- Con esta función se cerrara la ventana principal de la aplicación.

Form_Load.- Inicializa las variables del programa y carga los parámetros desde el registro de Windows.

Form_Unload.- Este evento asegura que el Comm virtual seleccionado quede cerrado y guarda los parámetros del programa en el registro de Windows.

ImportDat_Click.- Envía el comando “E” para indicarle al módulo DataCreep que entregue los datos de la prueba almacenados en el banco de memoria hacia el computador.

TestON_OnClick.- Esta función envía los comandos que inician (Run) o detienen (Stop) la adquisición de datos en el módulo DataCreep (poner en marcha la prueba).

Timer1_Timer.- Aquí se realizan dos tareas, una de ellas es enviar el comando “C” para que el módulo reconozca que los datos que a continuación se le transmite corresponden a los parámetros de la prueba, La otra tarea es la de actualizar el estado de las variables de la prueba, tales como temperatura, estiramiento, etc. mediante el comando A.

MSComm_OnComm.- Se activa si existen datos en el buffer de entrada del Com y llama a la función DataInComm.

DataInComm.- Función para capturar todos los datos que ingresan por el Comm, y además llama la función OnlineDato si la importación de datos esta activada.

OnlineDato.- Esta función interpreta la trama, que transmite el módulo, en la cual están los datos que representan el estado de la prueba.

ImportDato.- Función que procesa los datos que se encuentran en el buffer de entrada del Comm, para luego exportarla a una hoja de Excel.

Timer2_Timer.- Determina si la importación de datos ha finalizado y llama a la función ImportDato para interpretar los datos recibidos.

GetSetting.- Función para obtener parámetros del programa desde el registro de Windows.

SaveSetting.- Guarda los parámetros del programa en el registro de Windows.

La segunda ventana el usuario podrá configurar los parámetros del ensayo como ya se menciono anteriormente, y esta estructurada con los siguientes eventos.

Form_Load.- Carga el formulario de configuración de los parámetros de la prueba.

Cancel_cfgtest_Click.- Este evento ocasiona la salida del formulario de configuración sin guardar los cambios.

ok_cfgtest_Click.- Guarda los cambios realizados a las variables del ensayo y habilita una bandera para que la función Timer1_Timer envié los cambios realizados de la prueba al módulo DataCreep.

i_horas_OnChageUser.- Captura e interpreta los cambios efectuados al parámetro de hora.

i_muestras_OnChangeUser.- Captura e interpreta los cambios efectuados al parámetro que indica el número de muestras.

i_muestreo_OnChangeUser.- Captura e interpreta los cambios efectuados al parámetro que indica el período de muestreo.

iKnobX1_OnPositionChange.- Captura el valor fijado de temperatura en la perilla para el control de temperatura.

iSwitchLedX1_OnChange.- Este evento habilita y deshabilita el control de temperatura.

3.3.2 Pantallas.

La ventana principal (Fig 3.10) esta compuesta por visualizadores del estado de la prueba tales conectado, si la prueba esta o no corriendo, incluso si el control de temperatura esta activado, además se visualiza la temperatura de la prueba y el estiramiento del espécimen de manera sencilla.

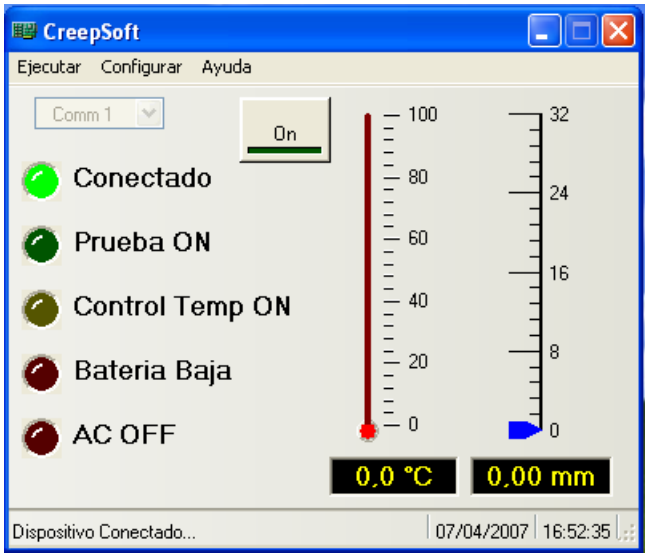


Figura 3.10 Ventana principal del programa de aplicación

Posee menús desplegables en donde se podrá ingresar a la ventana de configuración etc.

En la segunda ventana se observan dos pestañas, la primera es donde se inserta el número de muestras, período de toma de datos y tiempo total de la prueba (Fig. 3.11)

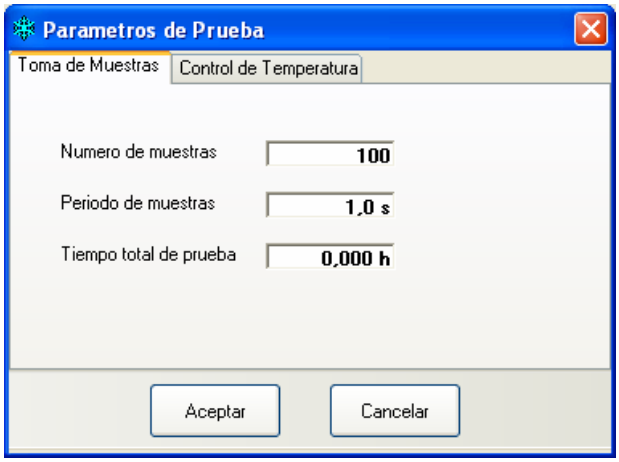


Figura 3.11 Ventana de parámetros de prueba (toma de muestras)

La última pestaña (Fig. 3.12) se establece la temperatura que se mantendrá fija la prueba y además la opción de habilitar el control de temperatura si el caso lo amerita.

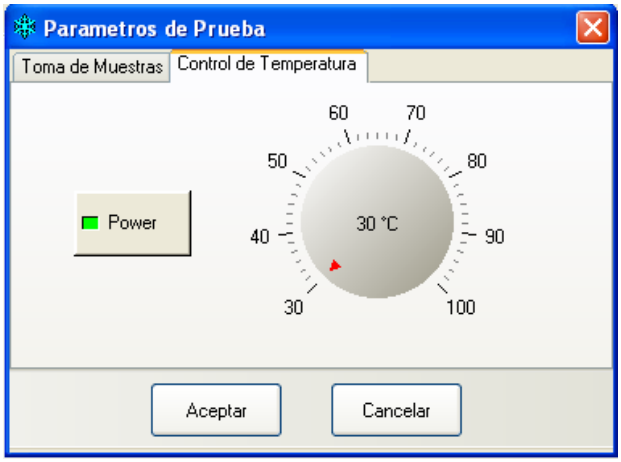


Figura 3.12 Ventana de parámetros de prueba (control de temperatura)

3.4 Alcances del sistema.

Sistema se lo ha diseñado para realizar tres tareas, una de ellas es la de tomar datos del ensayo cuyos parámetros reales son los de estiramiento del material a probar, y temperatura de la prueba. La temperatura podrá ser o no controlada (dependiendo de la prueba) para mantenerse a un valor preestablecido por el usuario y mediante una pantalla LCD se podrá visualizar los parámetros ya citados anteriormente.

Otra característica es la de transferir los datos almacenados, en un banco de memoria que posee el módulo, al ordenador mediante un puerto USB y cable de conexión del mismo estándar. Además tendrá la opción de ver los parámetros del ensayo en línea desde el computador para de esta manera realizar un seguimiento del ensayo.

Por último, el sistema consta de un programa instalado en un computador, que se encargará de la descarga de los datos almacenados en el banco de memoria del módulo, y los colocará en una hoja de Excel para su posterior análisis por personal calificado para dicha tarea, cabe recalcar que el software no realizará cálculo alguno.

CAPITULO 4

4 Implementación del hardware

4.1 Detalles de construcción del módulo DATACREEP.

Para realizar circuitos impresos es necesario diseñar regidos a parámetros y estándares que exige el mercado entre las más importantes tenemos:

Trazo de líneas.- Las líneas de conducción (vía) no deben tener cantos vivos, para evitar que se pierdan en el momento del comido de cobre. Si una vía se dobla a un ángulo V>90º el ángulo debería redondearse o ser truncado como se muestra en la figura 4.1. El fotoploter puede redondear el ángulo A, pero retendrá el rincón brusco en B que resultará tener el riesgo de:

Una bolsa de aire debajo de la mascar de soldadura.

La tinta en la impresión del screen saltará en la mascaras de soldadura.

Diámetro de agujeros y pads.- el diámetro de los agujeros determinado como d y el diámetro del pad de soldadura definido como dy, (Fig. 4.2) están estándarizados por las normas VDE y ANSI en la relación:

$$dy \geq d+0.5 \text{ mm}$$

Se debe considerar que existe una tolerancia en diámetro de las brocas para el proceso de +/- 0.05 y/o desplazamientos de las películas en el insolado, lo cual reducirá el diámetro del anillo del pad, por esta razón no es recomendable trabajar con los límites. Una relación mínima aceptable es:

dy ≥ d+0.8 mm

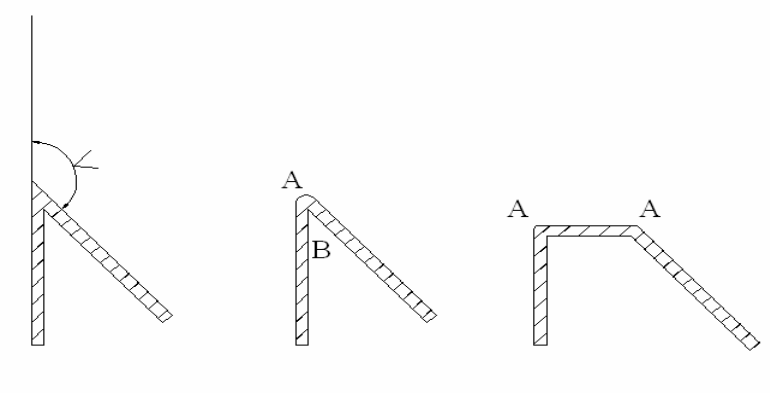


Figura 4.1 Modos de Vía

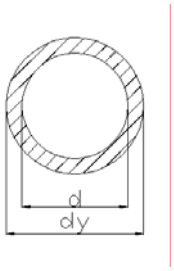


Figura 4.2 Diámetro de pads

Capacidad de Corriente.- En circuitos impresos el manejo de corriente es importante, debido a que una vía muy pequeña podría reventar con un excesivo paso de corriente. Las normas VDE, DIN, ANSI recomiendan:

Placa 18um 4mm-1Amp

Placa 35um 2mm-1Amp

Placa 70um 1mm-1Amp

Distancia de conectores y agujeros.- Para los conectores de fila que requieran un tratamiento especial de níquel-plata/oro es importante diseñarlos con distancias mínimas para realizar estos tratamientos.

Para cualquier tipo de agujero fuera de los conectores una distancia D **≥** 1.0 mm.

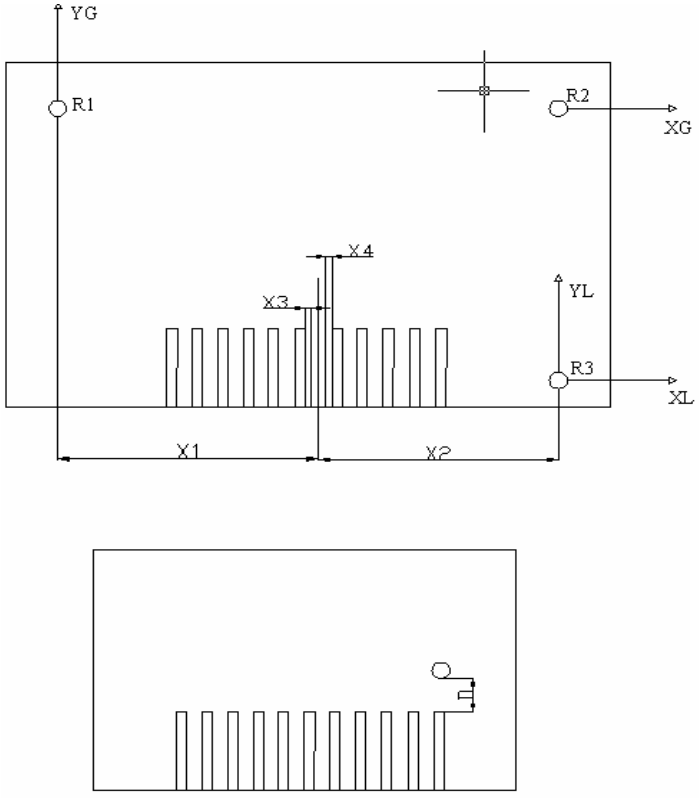


Figura 4.3 Distancia de conectores y agujeros

Un error aceptable entre conector de la cara superior e inferior, para d ≤ 0.2 mm.

Para el ancho del pin del conector tenemos t **≥** 1.5 mm.

El inicio de los pines debe existir un chaflan de 45º de un ancho de 0.3 ≤ a ≤ 0.4 mm.

Conector debe guardar simetría en el centrado Para el mecanizado guardar las respectivas distancias, tomando en cuenta el diámetro del router y simetría, debería mantener un mínimo de X3 = X4 ≥ 0.25 mm.

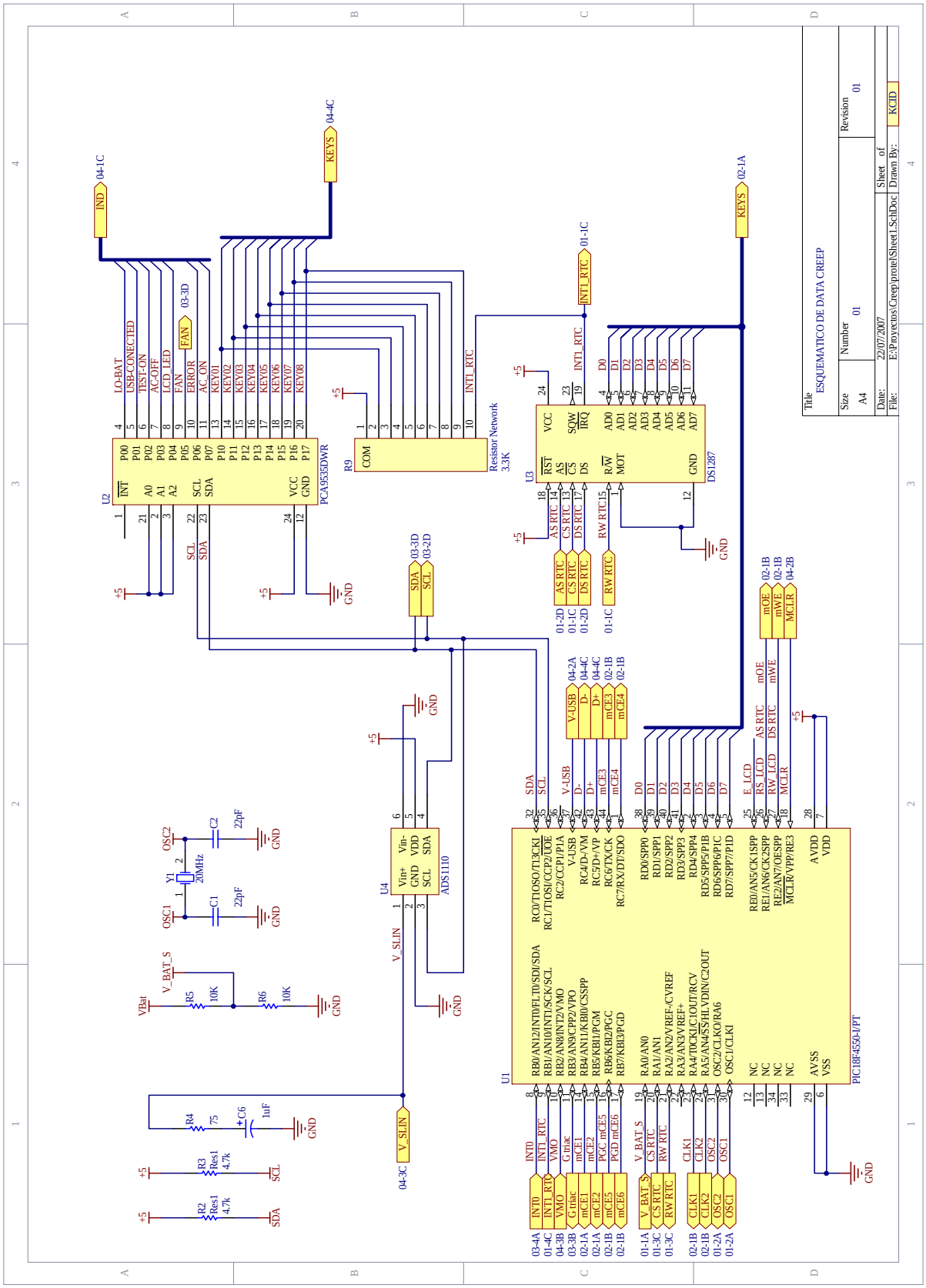


Figura 4.4 Diagrama esquemático de CPU y periféricos

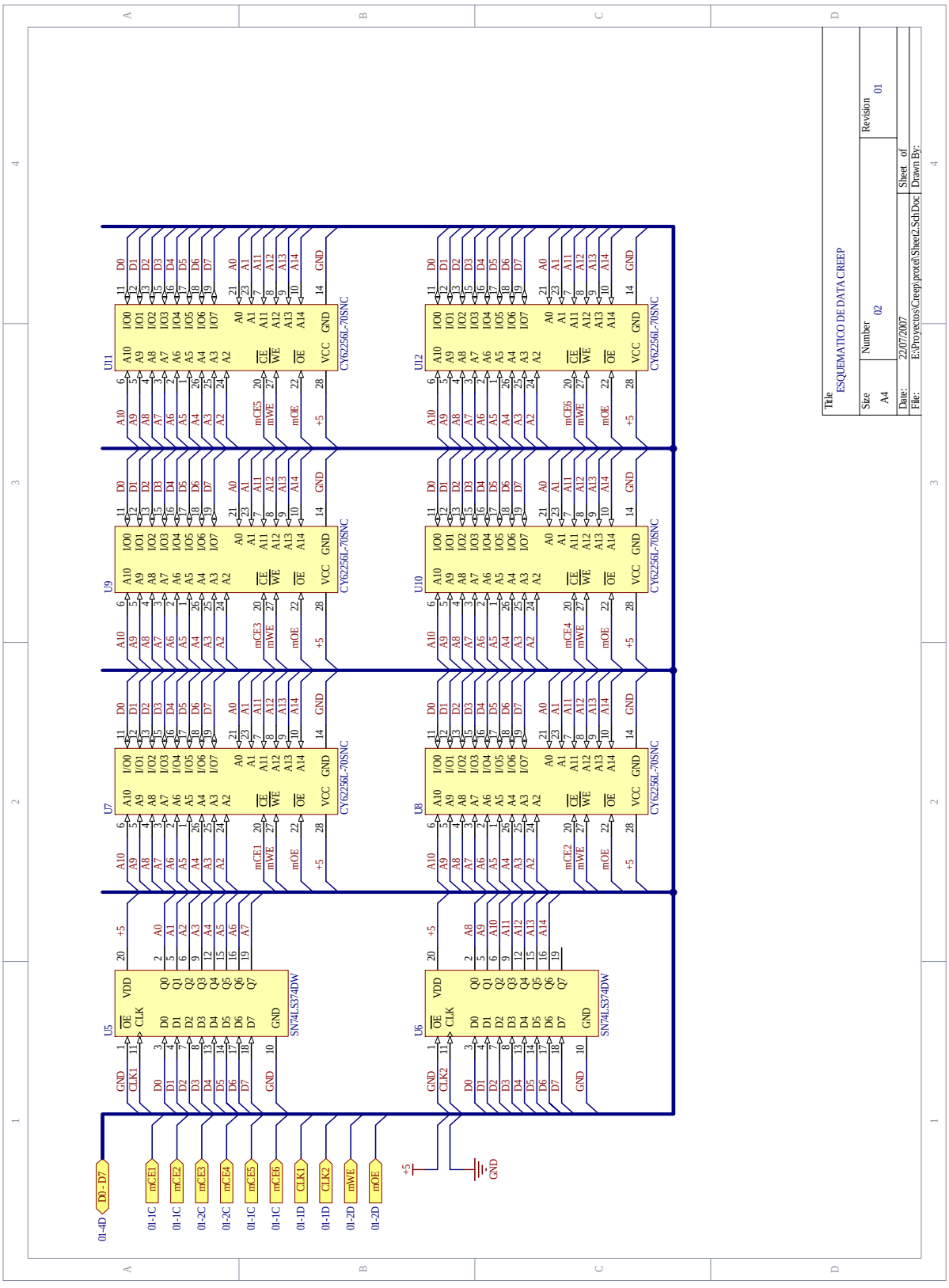


Figura 4.5 Diagrama esquemático del banco de memoria.

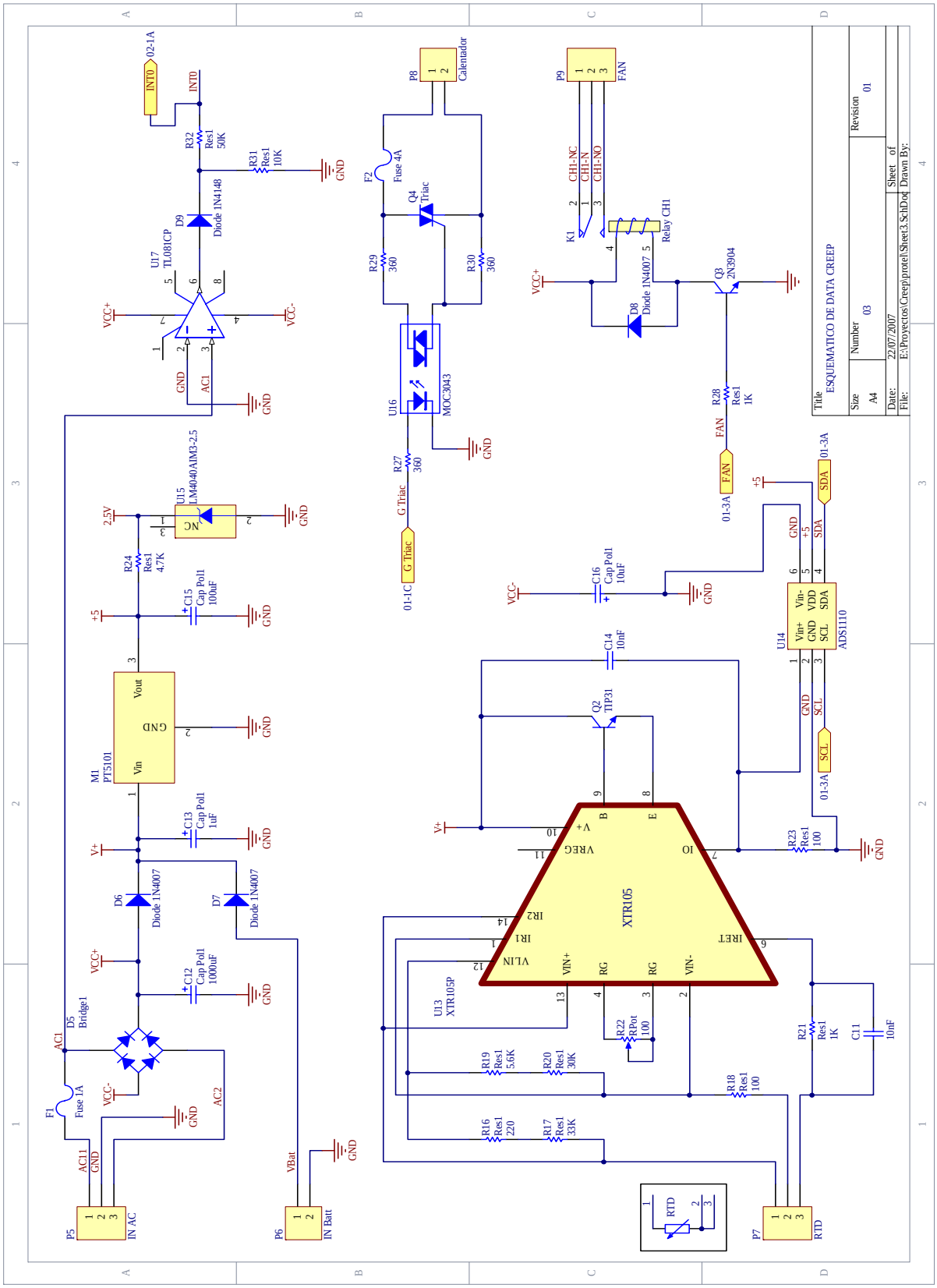


Figura 4.6 Diagrama esquemático de la fuente de alimentación y RTD

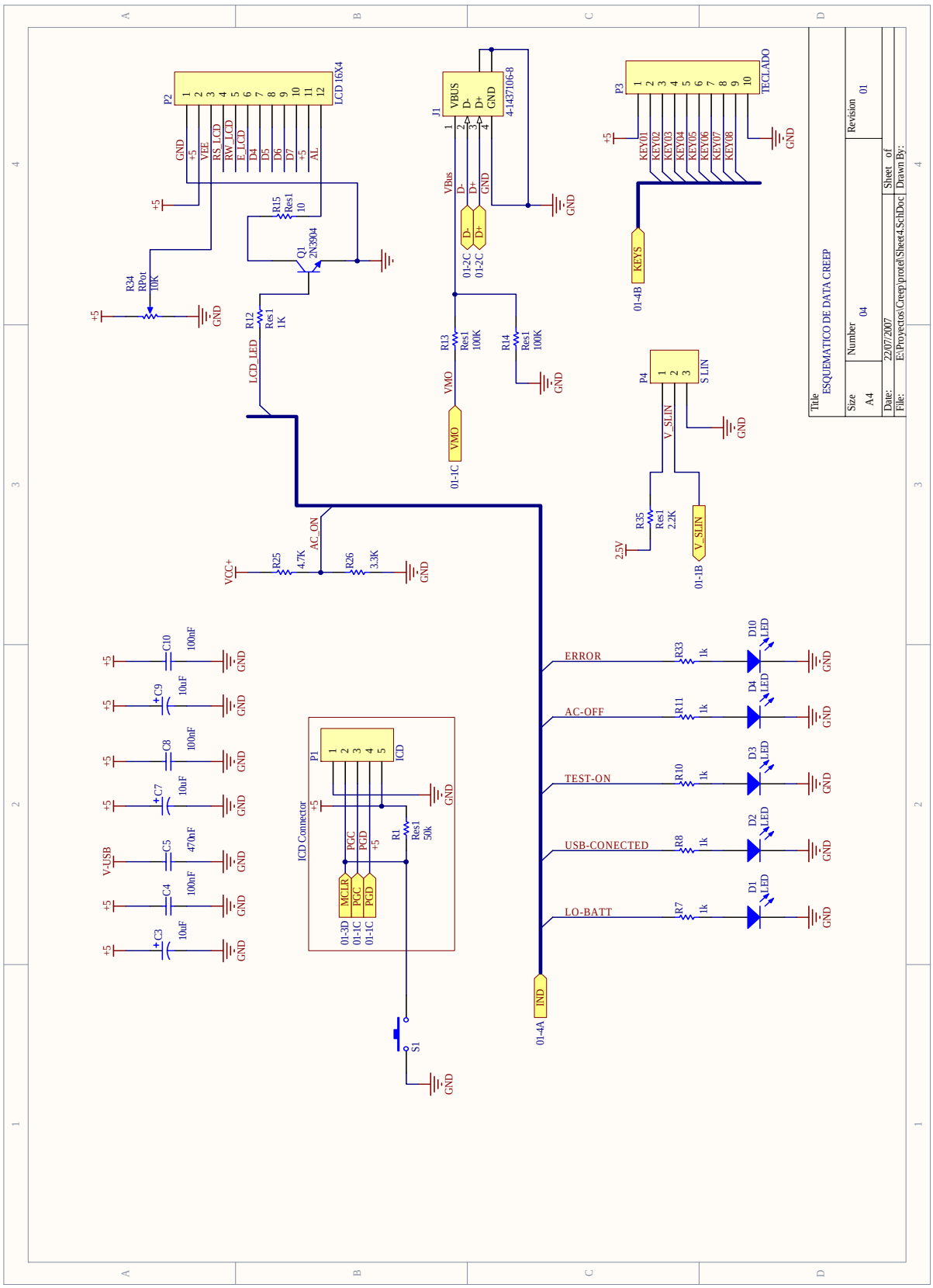


Figura 4.7 Diagrama esquemático de indicadores y conexión de periféricos

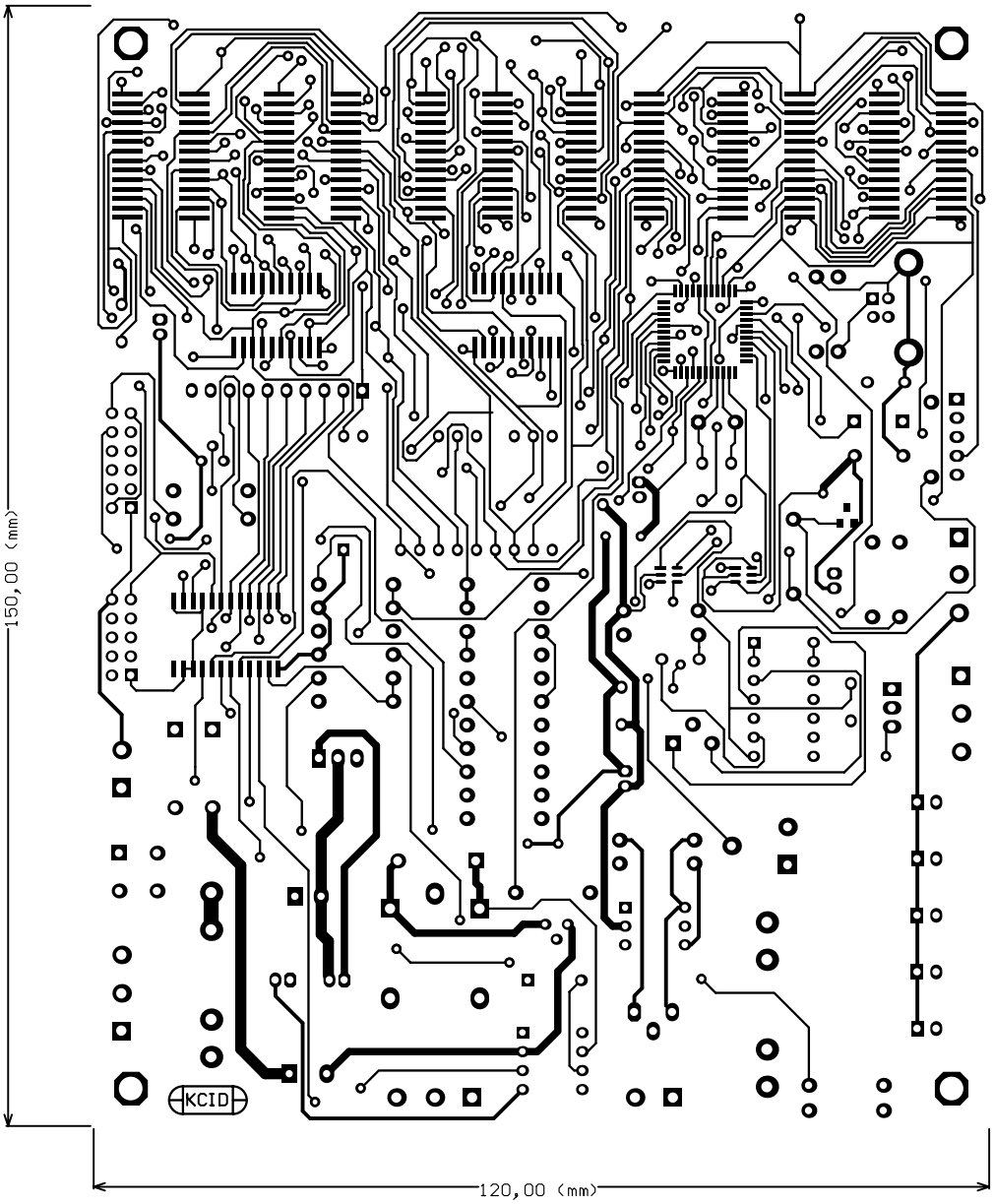


Figura 4.8 Diseño del PCB de la cara superior del circuito.

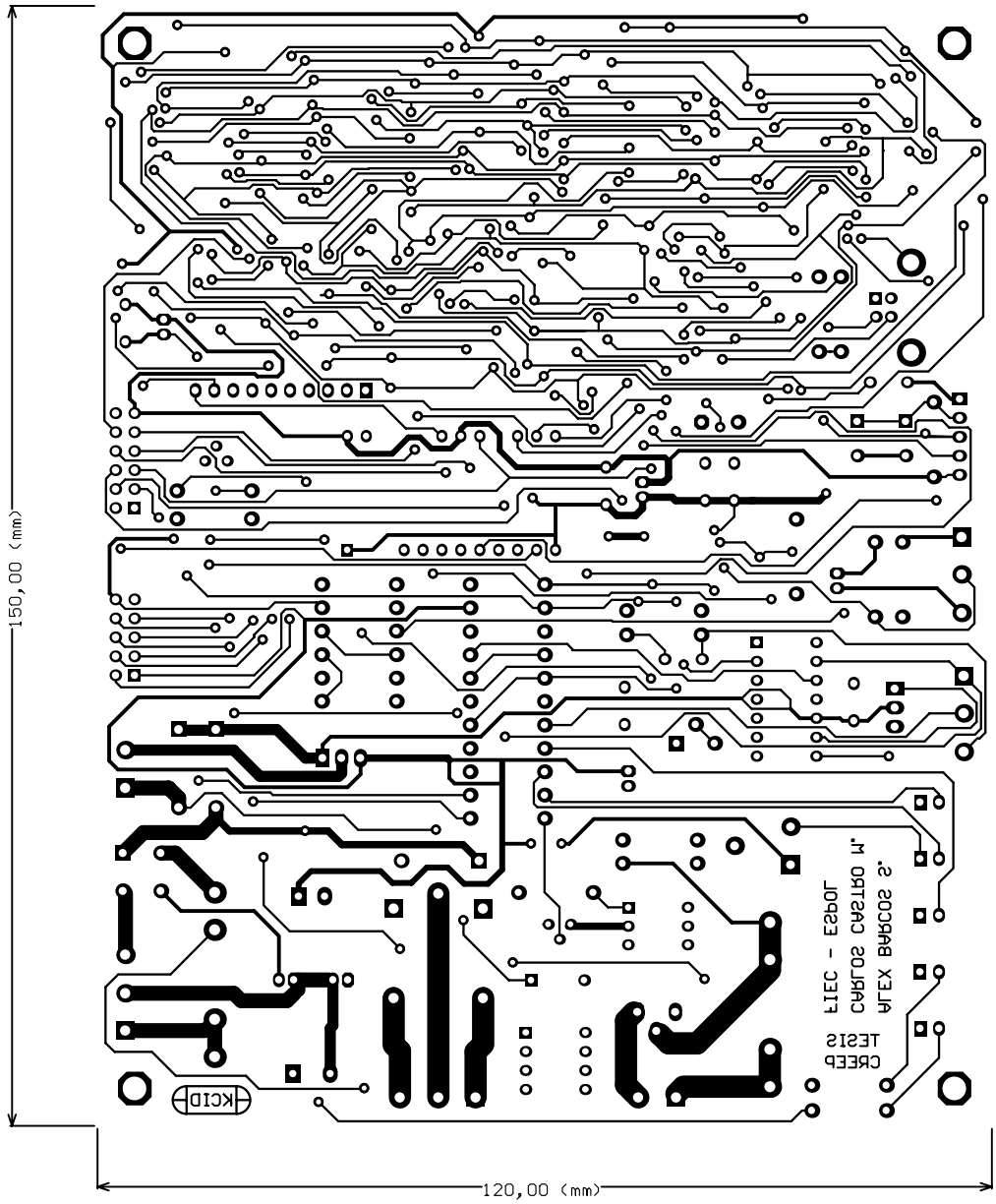


Figura 4.9 Diseño del PCB de la cara inferior del circuito.

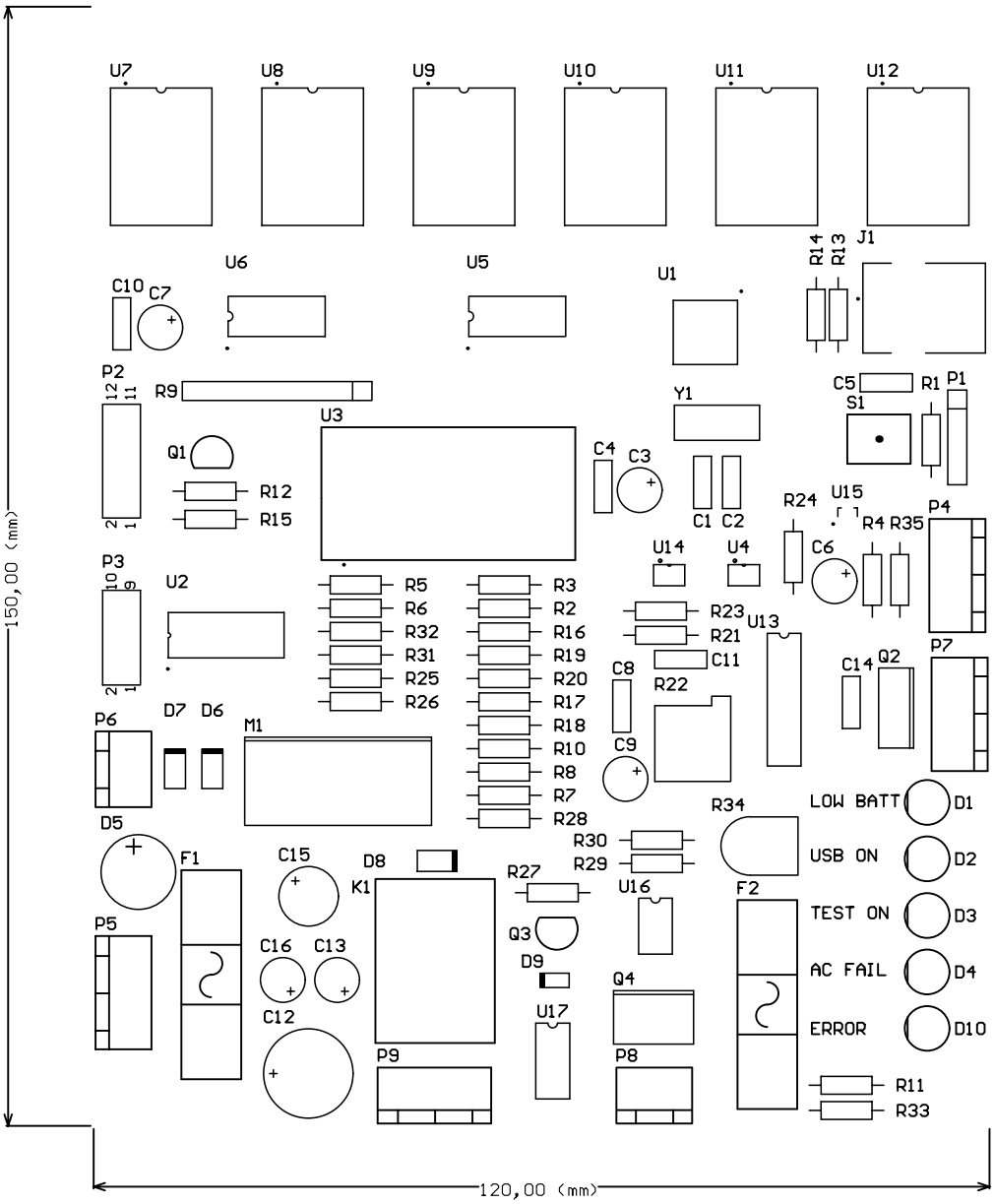


Figura 4.10 Diagrama de componentes de la tarjeta electrónica.

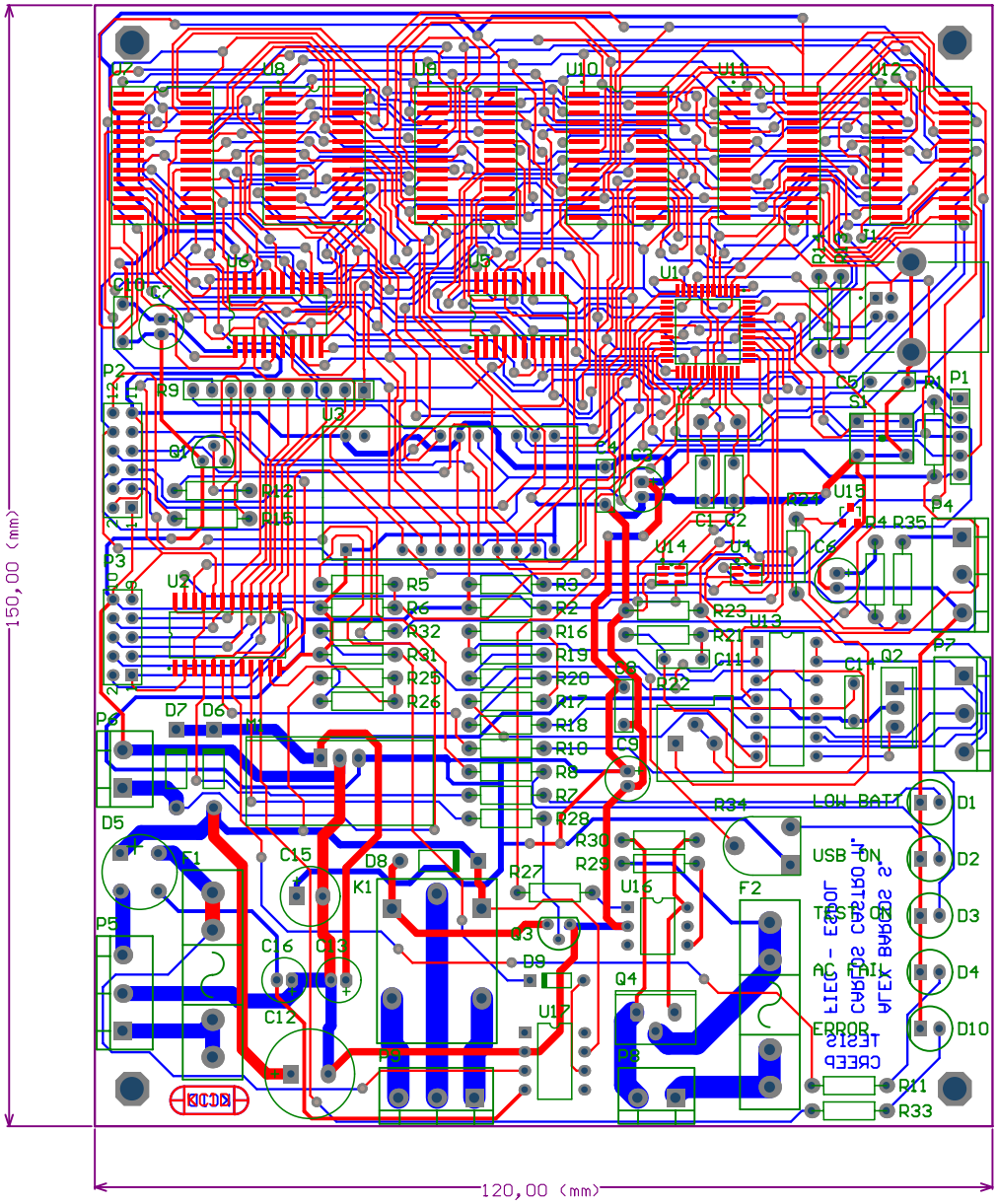


Figura 4.11 Diseño del PCB de las dos caras de la tarjeta electrónica

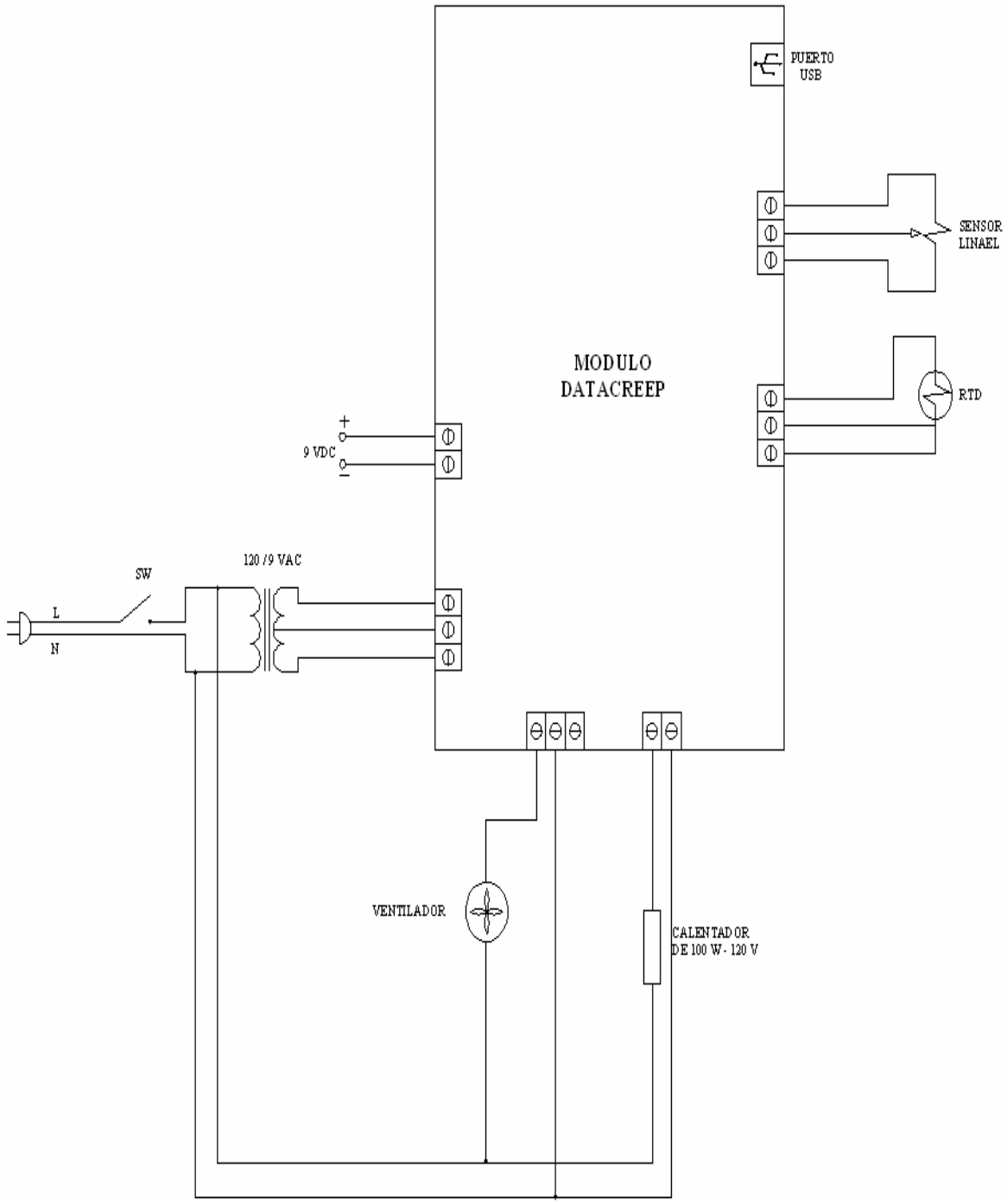


Figura 4.12 Esquemático del módulo

Conclusiones y Recomendaciones

Se logró implementar una tarjeta electrónica capaz de adquirir los datos de la prueba de carga constante en polímeros y además de controlar la temperatura del ambiente en donde se encuentre la muestra.

Con esta tarjeta se logro obtener una solución económica y confiable para este tipo de prueba que es practicada en la actualidad por industrias de plásticos.

Debido a las características internas del microcontrolador escogido, no fue necesario agregar módulos adicionales, tales como para la transferencias de datos mediante USB. Logrando ahorro en la implementación de la tarjeta y que el PCB sea de menores dimensiones.

Se aplicaron las bases adquiridas y se mejoraron los conocimientos en Lenguaje C, que es un programador de alto nivel para desarrollar el firmware del microcontrolador.

Para transferir datos almacenados desde el dispositivo al computador se utiliza el estándar USB logrando de esta manera conocer este tipo de tecnología que hoy en día es muy útil y que en la industria puede ser en ciertos casos útil y lograr reducir costos.

Durante el desarrollo del proyecto se mejoraron los conocimientos adquiridos en electrónica tanto digital como análoga y se obtuvieron conocimiento de herramientas, en el desarrollo de PCB´s mediante el software Altium (Protel)

y en el desarrollo del firmware el compilador de lenguaje C denominado PICC.

La tarjeta de adquisición se podría complementar con el desarrollo del sistema mecánico de la prueba de tensión constante y de esta forma poder realizar el ensayo real de una muestra de polímero.

Una modificación de la tarjeta de adquisición puede ser implementar una tarjeta de memoria flash, en donde se almacenará directamente los datos de la prueba, para luego retirarla y llevarla a un computador que disponga de una ranura para este tipo memoria flash SD.

BIBLIOGRAFÍA

“Everything you need to develope custom USB peripherals”

Jan Alexon third Edition

“USB Desing by Example”

John Hide

“Curso practico de electrónica industrial”

Editorial Cekit S.A.

“Visual Basic for electronics engineering applications”

Vincent Himpe second edition

“C compiler reference manual”

Custom Computer Service, 2002

“USB Design by example”

John Hide, third Edition 2004

“Nota técnica No. 10”

Arian

www.microchip.com

www.ti.com

Anexos

A. Tabla de pruebas y simulación del Ensayo de tensión Constante

Parámetros del ensayo

Temperatura del ensayo: 45ºC

Temperatura ambiente: 24ºC

Número de muestras: 2500

Período de muestra: 1.5 segundos

Tiempo Total de la prueba: 1.042 horas

Debido a la gran cantidad de muestras tomadas en el ensayo de tensión constante, solo se consideraron los puntos de mayor variación para presentarlos en la tabla A1 que a continuación se presenta.

Index	Tiempo	Long	Temp.	Index	Tiempo	Long	Temp.	Index	Tiempo	Long	Temp.
1	1.5	0	24.5	46	69	0	27.3	81	121.5	0.55	33.9
2	3	0	24.5	47	70.5	0	27.5	82	123	0.67	34.1
3	4.5	0	24.5	48	72	0	27.6	83	124.5	0.67	34.3
4	6	0	24.5	49	73.5	0	27.7	84	126	0.67	34.5
5	7.5	0	24.5	50	75	0	27.9	85	127.5	0.67	34.8
8	12	0	24.6	51	76.5	0	28.1	86	129	0.67	35
9	13.5	0	24.6	52	78	0	28.2	87	130.5	0.67	35.2
10	15	0	24.6	53	79.5	0	28.4	88	132	0.67	35.5
12	18	0	24.7	54	81	0	28.6	89	133.5	0.67	35.8
13	19.5	0	24.8	55	82.5	0.12	28.7	90	135	0.67	36
16	24	0	24.9	56	84	0.5	28.9	91	136.5	0.67	36.2
19	28.5	0	25	57	85.5	0.5	29.1	92	138	0.67	36.4
20	30	0	25.1	58	87	0.5	29.2	93	139.5	0.67	36.6
23	34.5	0	25.2	59	88.5	0.5	29.4	94	141	0.67	36.9
24	36	0	25.3	60	90	0.5	29.6	95	142.5	0.67	37.1
25	37.5	0	25.3	61	91.5	0.5	29.8	96	144	0.67	37.3
26	39	0	25.4	62	93	0.5	30	97	145.5	0.67	37.6
28	42	0	25.5	63	94.5	0.5	30.2	98	147	0.67	37.8
29	43.5	0	25.6	64	96	0.5	30.5	99	148.5	0.67	38
30	45	0	25.6	65	97.5	0.5	30.7	100	150	0.67	38.2
31	46.5	0	25.7	66	99	0.5	30.9	101	151.5	0.67	38.5
32	48	0	25.8	67	100.5	0.5	31.1	102	153	0.67	38.7
33	49.5	0	25.9	68	102	0.5	31.3	103	154.5	0.67	38.9
34	51	0	26	69	103.5	0.5	31.6	104	156	0.67	39.2
35	52.5	0	26	70	105	0.5	31.8	105	157.5	0.67	39.4
36	54	0	26.1	71	106.5	0.5	32	106	159	0.67	39.6
37	55.5	0	26.2	72	108	0.5	32.2	107	160.5	0.67	39.8
38	57	0	26.3	73	109.5	0.5	32.4	108	162	0.67	40
39	58.5	0	26.4	74	111	0.5	32.6	109	163.5	0.67	40.2
40	60	0	26.6	75	112.5	0.5	32.8	110	165	0.67	40.4
41	61.5	0	26.7	76	114	0.5	32.9	111	166.5	0.67	40.6
42	63	0	26.8	77	115.5	0.5	33.1	112	168	0.67	40.9
43	64.5	0	26.9	78	117	0.5	33.3	113	169.5	0.67	41.1
44	66	0	27	79	118.5	0.5	33.5	114	171	0.67	41.3
45	67.5	0	27.1	80	120	0.5	33.7	115	172.5	0.67	41.5

Tabla A1. Datos del ensayo de tensión constante

Index	Tiempo	Long	Temp.	Index	Tiempo	Long	Temp.	Index	Tiempo	Long	Temp.
116	174	0.67	41.7	151	226.5	0.67	47.9	186	279	0.67	51.6
117	175.5	0.67	41.9	152	228	0.67	48	187	280.5	0.66	51.7
118	177	0.67	42.1	153	229.5	0.67	48.1	188	282	0.66	51.7
119	178.5	0.67	42.3	154	231	0.67	48.3	189	283.5	0.67	51.8
120	180	0.67	42.5	155	232.5	0.67	48.4	190	285	0.67	51.8
121	181.5	0.67	42.7	156	234	0.67	48.5	191	286.5	0.67	51.9
122	183	0.67	42.9	157	235.5	0.67	48.7	192	288	0.66	51.9
123	184.5	0.67	43.1	158	237	0.67	48.8	193	289.5	0.67	52
124	186	0.67	43.3	159	238.5	0.67	48.9	194	291	0.67	52
125	187.5	0.67	43.5	160	240	0.67	49	195	292.5	0.66	52
126	189	0.67	43.8	161	241.5	0.67	49.1	196	294	0.67	52.1
127	190.5	0.67	44	162	243	0.67	49.3	197	295.5	0.67	52.1
128	192	0.67	44.1	163	244.5	0.67	49.4	198	297	0.67	52.1
129	193.5	0.67	44.4	164	246	0.67	49.5	199	298.5	0.66	52.2
130	195	0.67	44.5	165	247.5	0.66	49.6	200	300	0.67	52.2
131	196.5	0.67	44.8	166	249	0.67	49.7	201	301.5	0.67	52.3
132	198	0.67	45	167	250.5	0.67	49.9	202	303	0.67	52.3
133	199.5	0.67	45.1	168	252	0.67	50	203	304.5	0.67	52.3
134	201	0.67	45.3	169	253.5	0.67	50.1	204	306	0.66	52.3
135	202.5	0.67	45.4	170	255	0.67	50.2	205	307.5	0.67	52.3
136	204	0.67	45.6	171	256.5	0.66	50.3	206	309	0.66	52.4
137	205.5	0.67	45.8	172	258	0.67	50.5	207	310.5	0.67	52.4
138	207	0.67	45.9	173	259.5	0.67	50.5	208	312	0.66	52.4
139	208.5	0.67	46.1	174	261	0.66	50.6	209	313.5	0.66	52.4
140	210	0.67	46.2	175	262.5	0.67	50.7	210	315	0.66	52.4
141	211.5	0.67	46.4	176	264	0.67	50.8	211	316.5	0.67	52.4
142	213	0.67	46.5	177	265.5	0.66	50.9	262	393	0.66	52.2
143	214.5	0.67	46.7	178	267	0.67	51	263	394.5	0.67	52.2
144	216	0.67	46.8	179	268.5	0.67	51.1	264	396	0.66	52.1
145	217.5	0.67	47	180	270	0.67	51.2	265	397.5	0.66	52.1
146	219	0.67	47.2	181	271.5	0.67	51.3	266	399	0.67	52
147	220.5	0.67	47.3	182	273	0.67	51.3	267	400.5	0.66	52
148	222	0.66	47.5	183	274.5	0.67	51.4	268	402	0.66	52
149	223.5	0.67	47.6	184	276	0.67	51.5	269	403.5	0.66	52
150	225	0.67	47.7	185	277.5	0.67	51.5	270	405	0.66	52

Tabla A1. Datos del ensayo de tensión constante

Index	Tiempo	Long	Temp.	Index	Tiempo	Long	Temp.	Index	Tiempo	Long	Temp.
271	406.5	0.66	51.9	305	457.5	0.66	50.7	339	508.5	0.66	49.2
272	408	0.67	51.9	306	459	0.66	50.7	340	510	0.66	49.2
273	409.5	0.66	51.9	307	460.5	0.66	50.6	341	511.5	0.66	49.1
274	411	0.66	51.8	308	462	0.66	50.6	342	513	0.66	49.1
275	412.5	0.66	51.8	309	463.5	0.66	50.5	343	514.5	0.66	49
276	414	0.66	51.8	310	465	0.66	50.5	344	516	0.66	49
277	415.5	0.66	51.7	311	466.5	0.66	50.5	345	517.5	0.67	49
278	417	0.67	51.7	312	468	0.67	50.4	346	519	0.66	48.9
279	418.5	0.66	51.7	313	469.5	0.66	50.4	347	520.5	0.66	48.9
280	420	0.66	51.6	314	471	0.66	50.3	348	522	0.66	48.8
281	421.5	0.66	51.6	315	472.5	0.66	50.3	349	523.5	0.66	48.8
282	423	0.66	51.5	316	474	0.66	50.2	350	525	0.66	48.7
283	424.5	0.66	51.5	317	475.5	0.66	50.2	351	526.5	0.66	48.7
284	426	0.66	51.5	318	477	0.66	50.1	352	528	0.66	48.6
285	427.5	0.67	51.4	319	478.5	0.66	50.1	353	529.5	0.66	48.6
286	429	0.66	51.4	320	480	0.66	50	354	531	0.67	48.5
287	430.5	0.66	51.4	321	481.5	0.66	50	355	532.5	0.66	48.5
288	432	0.66	51.3	322	483	0.66	50	356	534	0.66	48.4
289	433.5	0.66	51.3	323	484.5	0.66	49.9	357	535.5	0.66	48.4
290	435	0.66	51.3	324	486	0.66	49.9	358	537	0.79	48.3
291	436.5	0.67	51.2	325	487.5	0.66	49.8	359	538.5	1	48.3
292	438	0.66	51.2	326	489	0.67	49.8	360	540	1.31	48.2
293	439.5	0.66	51.1	327	490.5	0.66	49.7	361	541.5	1.31	48.2
294	441	0.66	51.1	328	492	0.66	49.7	362	543	1.31	48.2
295	442.5	0.66	51.1	329	493.5	0.66	49.7	363	544.5	1.31	48.1
296	444	0.66	51	330	495	0.66	49.6	364	546	1.31	48.1
297	445.5	0.66	51	331	496.5	0.66	49.6	365	547.5	1.31	48
298	447	0.66	51	332	498	0.66	49.5	366	549	1.31	48
299	448.5	0.66	50.9	333	499.5	0.66	49.5	367	550.5	1.31	47.9
300	450	0.66	50.9	334	501	0.66	49.5	368	552	1.31	47.9
301	451.5	0.66	50.8	335	502.5	0.67	49.4	369	553.5	1.31	47.8
302	453	0.66	50.8	336	504	0.66	49.4	370	555	1.31	47.8
303	454.5	0.66	50.8	337	505.5	0.66	49.3	371	556.5	1.31	47.7
304	456	0.66	50.7	338	507	0.66	49.3	372	558	1.31	47.7
45	67.5	0	27.1	80	120	0.5	33.7	115	172.5	0.67	41.5

Tabla A1. Datos del ensayo de tensión constante

Index	Tiemp	Long	Temp	Index	Tiemp	Long	Temp	Index	Tiemp	Long	Temp
373	560	1.31	47.6	408	612	1.31	46	443	665	1.31	44.5
374	561	1.31	47.6	409	614	1.31	46	444	666	1.31	44.5
375	563	1.31	47.6	410	615	1.31	45.9	445	668	1.31	44.4
376	564	1.31	47.5	411	617	1.31	45.9	446	669	1.36	44.4
377	566	1.31	47.5	412	618	1.31	45.8	447	671	1.75	44.3
378	567	1.31	47.4	413	620	1.31	45.8	448	672	1.81	44.3
379	569	1.31	47.4	414	621	1.31	45.7	449	674	1.92	44.3
380	570	1.31	47.3	415	623	1.31	45.7	450	675	1.92	44.2
381	572	1.31	47.3	416	624	1.31	45.6	451	677	1.92	44.2
382	573	1.31	47.2	417	626	1.31	45.6	452	678	1.93	44.1
383	575	1.31	47.2	418	627	1.31	45.6	453	680	1.92	44.1
384	576	1.31	47.1	419	629	1.31	45.5	454	681	1.92	44.1
385	578	1.31	47.1	420	630	1.31	45.5	455	683	1.92	44
386	579	1.31	47	421	632	1.31	45.4	456	684	1.92	44
387	581	1.31	47	422	633	1.31	45.4	457	686	1.92	44
388	582	1.31	46.9	423	635	1.31	45.3	458	687	1.92	43.9
389	584	1.31	46.9	424	636	1.31	45.3	459	689	1.92	43.9
390	585	1.31	46.8	425	638	1.31	45.2	460	690	1.92	43.9
391	587	1.31	46.8	426	639	1.31	45.2	461	692	1.92	43.8
392	588	1.31	46.8	427	641	1.31	45.1	462	693	1.93	43.8
393	590	1.31	46.7	428	642	1.31	45.1	463	695	1.92	43.8
394	591	1.31	46.7	429	644	1.31	45.1	464	696	1.92	43.8
395	593	1.31	46.6	430	645	1.31	45	465	698	1.92	43.7
396	594	1.31	46.6	431	647	1.31	45	466	699	1.92	43.7
397	596	1.31	46.5	432	648	1.31	44.9	467	701	1.92	43.7
398	597	1.31	46.5	433	650	1.31	44.9	468	702	1.92	43.6
399	599	1.31	46.4	434	651	1.31	44.8	469	704	1.93	43.6
400	600	1.31	46.4	435	653	1.31	44.8	470	705	1.92	43.6
401	602	1.31	46.3	436	654	1.31	44.8	471	707	1.92	43.6
402	603	1.31	46.3	437	656	1.31	44.7	472	708	1.93	43.5
403	605	1.31	46.3	438	657	1.31	44.7	473	710	1.92	43.5
404	606	1.31	46.2	439	659	1.31	44.7	474	711	1.92	43.5
405	608	1.31	46.2	440	660	1.31	44.6	475	713	1.92	43.4
406	609	1.31	46.1	441	662	1.31	44.6	476	714	1.92	43.4
407	611	1.31	46.1	442	663	1.31	44.5	477	716	1.92	43.4

Tabla A1. Datos del ensayo de tensión constante

Index	Tiempo	Long	Temp.	Index	Tiempo	Long	Temp.	Index	Tiempo	Long	Temp.
636	954	1.92	44.4	671	1007	1.92	45	822	1233	2.6	45.3
637	955.5	1.92	44.5	672	1008	1.93	45	823	1235	2.6	45.3
638	957	1.92	44.5	673	1010	1.92	45.1	824	1236	2.6	45.3
639	958.5	1.93	44.5	674	1011	1.93	45.1	825	1238	2.59	45.3
640	960	1.92	44.5	675	1013	1.92	45.1	826	1239	2.94	45.3
641	961.5	1.92	44.5	676	1014	2.19	45.1	827	1241	2.95	45.3
642	963	1.93	44.5	677	1016	2.19	45.1	828	1242	3.01	45.3
643	964.5	1.92	44.6	678	1017	2.23	45.1	829	1244	3.01	45.3
644	966	1.92	44.6	679	1019	2.23	45.2	830	1245	3.01	45.2
645	967.5	1.92	44.6	680	1020	2.23	45.2	831	1247	3.01	45.2
646	969	1.92	44.6	681	1022	2.23	45.2	832	1248	3.01	45.2
647	970.5	1.92	44.6	682	1023	2.23	45.2	833	1250	3.01	45.2
648	972	1.92	44.6	683	1025	2.23	45.3	834	1251	3.01	45.2
649	973.5	1.93	44.7	684	1026	2.23	45.3	835	1253	3.01	45.2
650	975	1.92	44.7	685	1028	2.31	45.3	836	1254	3.01	45.2
651	976.5	1.92	44.7	686	1029	2.6	45.3	837	1256	3.01	45.2
652	978	1.93	44.7	687	1031	2.6	45.3	838	1257	3.01	45.2
653	979.5	1.92	44.7	688	1032	2.6	45.3	839	1259	3.01	45.1
654	981	1.92	44.7	689	1034	2.6	45.4	840	1260	3.01	45.1
655	982.5	1.92	44.7	690	1035	2.6	45.4	841	1262	3.01	45.1
656	984	1.93	44.8	691	1037	2.6	45.4	842	1263	3.01	45.1
657	985.5	1.92	44.8	692	1038	2.6	45.4	843	1265	3.01	45.1
658	987	1.92	44.8	693	1040	2.6	45.4	844	1266	3.01	45.1
659	988.5	1.93	44.8	694	1041	2.6	45.4	845	1268	3.01	45.1
660	990	1.92	44.8	695	1043	2.6	45.4	846	1269	3.01	45.1
661	991.5	1.92	44.9	696	1044	2.6	45.4	847	1271	3.01	45
662	993	1.93	44.9	697	1046	2.6	45.5	860	1290	3.01	45
663	994.5	1.92	44.9	704	1056	2.6	45.6	861	1292	3.01	44.9
664	996	1.92	44.9	764	1146	2.6	45.5	887	1331	3.01	44.9
665	997.5	1.92	44.9	795	1193	2.6	45.4	888	1332	3.01	44.8
666	999	1.92	44.9	796	1194	2.6	45.4	965	1448	3.01	44.7
667	1001	1.92	44.9	797	1196	2.6	45.4	967	1451	3.01	44.8
668	1002	1.92	45	819	1229	2.6	45.4	1068	1602	3.71	44.9
669	1004	1.93	45	820	1230	2.6	45.4	1123	1685	3.88	45
670	1005	1.92	45	821	1232	2.6	45.3	1124	1686	3.88	45.1

Tabla A1. Datos del ensayo de tensión constante

Index	Tiempo	Long	Temp.	Index	Tiempo	Long	Temp.	Index	Tiempo	Long	Temp.
1184	1776	4.73	45	1623	2435	5.45	45	2000	3000	6.15	44.9
1185	1778	4.73	45.1	1624	2436	5.45	45	2030	3045	6.15	44.9
1186	1779	4.73	45.1	1625	2438	5.45	45	2031	3047	6.15	45
1187	1781	4.73	45	1626	2439	5.45	45	2138	3207	7.3	45
1188	1782	4.73	45.1	1627	2441	5.45	44.9	2139	3209	7.3	45.1
1204	1806	4.73	45.1	1628	2442	5.45	44.9	2140	3210	7.3	45
1205	1808	4.73	45	1629	2444	5.45	45	2141	3212	7.3	45.1
1206	1809	4.73	45.1	1705	2558	5.64	44.9	2152	3228	7.3	45.1
1207	1811	4.73	45	1706	2559	6.03	45	2153	3230	7.3	45
1208	1812	4.73	45	1750	2625	6.15	45	2154	3231	7.3	45
1219	1829	4.73	45.1	1751	2627	6.15	45.1	2155	3233	7.3	45.1
1239	1859	4.73	45.2	1781	2672	6.14	45	2156	3234	7.3	45.1
1240	1860	4.73	45.1	1782	2673	6.15	45.1	2157	3236	7.3	45
1261	1892	4.73	45.2	1783	2675	6.15	45	2180	3270	7.3	45
1270	1905	4.73	45.1	1849	2774	6.15	45.1	2181	3272	7.3	45.1
1300	1950	4.73	45.1	1895	2843	6.15	45.1	2229	3344	8.05	45.1
1301	1952	4.73	45	1896	2844	6.15	45	2230	3345	8.05	45
1356	2034	5.45	44.9	1944	2916	6.15	44.9	2231	3347	8.05	45.1
1357	2036	5.45	45	1945	2918	6.15	45	2232	3348	8.05	45.1
1358	2037	5.45	44.9	1946	2919	6.15	45	2233	3350	8.05	45
1445	2168	5.45	44.9	1947	2921	6.15	44.9	2234	3351	8.05	45
1446	2169	5.45	45	1948	2922	6.15	45	2235	3353	8.05	45.1
1512	2268	5.45	45.1	1949	2924	6.15	45	2255	3383	8.05	45
1513	2270	5.45	45	1950	2925	6.15	44.9	2333	3500	8.53	44.9
1519	2279	5.45	45.1	1963	2945	6.15	44.9	2334	3501	8.53	45
1520	2280	5.45	45	1964	2946	6.15	45	2341	3512	8.53	44.9
1572	2358	5.45	45	1965	2948	6.15	45	2368	3552	8.53	44.9
1580	2370	5.45	44.9	1966	2949	6.15	45	2399	3599	8.59	45
1581	2372	5.45	45	1967	2951	6.15	45	2426	3639	8.59	45.1
1618	2427	5.45	44.9	1968	2952	6.15	45	2427	3641	8.59	45.2
1619	2429	5.45	45	1969	2954	6.15	45	2430	3645	8.59	45.1
1620	2430	5.45	44.9	1970	2955	6.15	44.9	2431	3647	8.59	45.1
1621	2432	5.45	44.9	1971	2957	6.15	45	2442	3663	9.33	45
1622	2433	5.45	44.9	1972	2958	6.15	45	2468	3702	9.32	45
670	1005	1.92	45	821	1232	2.6	45.3	1124	1686	3.88	45.1

Tabla A1. Datos del ensayo de tensión constante

Con estos datos obtenemos las curvas de temperatura y deformación que a continuación se presentan.

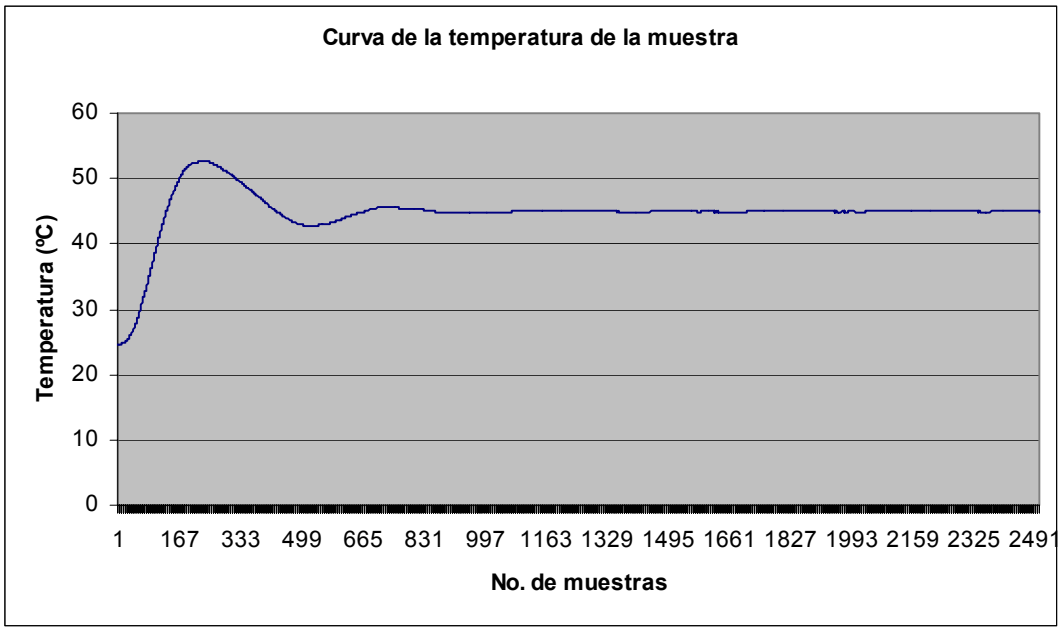


Fig. A1 Curva temperatura del ensayo de tensión constante

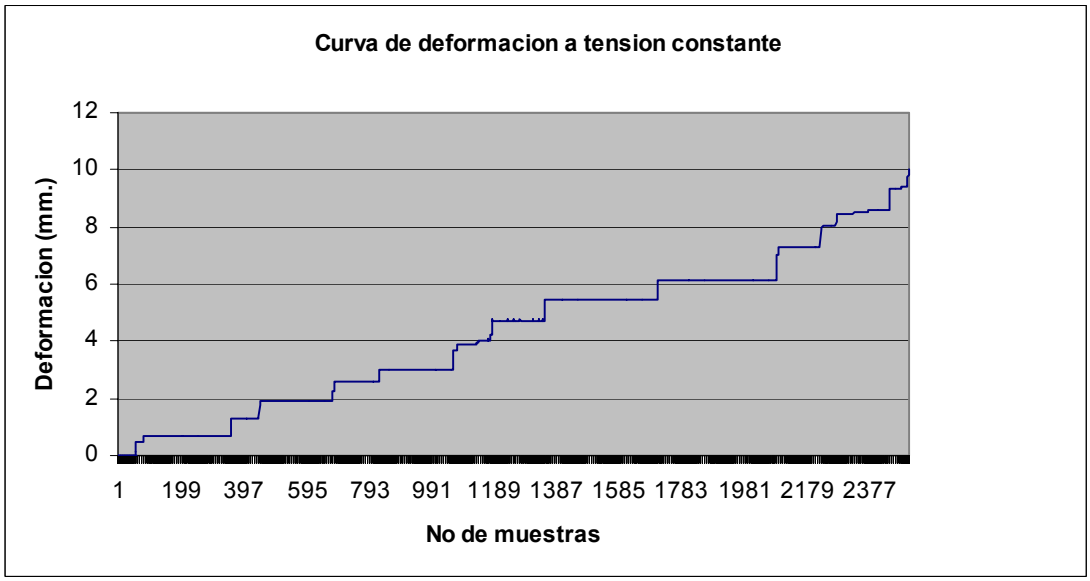


Fig. A2 Curva la deformación del ensayo de tensión constante

B. Código fuente del hardware (firmware).

Código fuente general (firmware).

```
#include "F:\Proyectos\Creep\CCS\Creep Control\Control.h"
#include "F:\Proyectos\Creep\CCS\Creep Control\lcd.h"
#include "F:\Proyectos\Creep\CCS\Creep Control\BQ328x.h"
#include "F:\Proyectos\Creep\CCS\Creep Control\ADS111x.h"
#include "F:\Proyectos\Creep\CCS\Creep Control\UT61256.h"
#include "F:\Proyectos\Creep\CCS\Creep Control\PCA9535.h"
#include "F:\Proyectos\Creep\CCS\Creep Control\eeprom_internal.h"
#include "F:\Proyectos\Creep\CCS\Creep Control\funtions.h"
#include "F:\Proyectos\Creep\CCS\Creep Control\pantallas.h"
#include "F:\Proyectos\Creep\CCS\Creep Control\Task.h"
#include "F:\Proyectos\Creep\CCS\Creep Control\interrupciones.h"

void main()
{
   setup_adc_ports(AN0|VSS_VDD);
   setup_adc(ADC_CLOCK_INTERNAL);
   setup_psp(PSP_DISABLED);
   setup_wdt(WDT_ON);
   setup_timer_0(RTCC_INTERNAL);
   setup_timer_1(T1_INTERNAL|T1_DIV_BY_4);   // donde cada pulso
   dura 1/3 de us, con un maximo de 20ms
   setup_timer_2(T2_DIV_BY_16,149,5);       // interrupcion  cada 1.0ms
   setup_timer_3(T3_DISABLED|T3_DIV_BY_1);
   setup_comparator(NC_NC_NC_NC);
   setup_vref(FALSE);
   enable_interrupts(INT_TIMER2);
   enable_interrupts(INT_EXT1);
   enable_interrupts(GLOBAL);
   setup_low_volt_detect(FALSE);
   setup_oscillator(False);
  //***************************
   delay_ms(250);
   loadmemory();
   usb_init();              //inicializacion del USB
    //***************************
   ext_int_edge( 1, H_TO_L);          // Set up PIC18 EXT2
   SET_TRIS_A( 0b11001001 );      //Cargando config de los reg Tris.
   SET_TRIS_B( 0b00000111 );
   SET_TRIS_C( 0b00111100 );
   SET_TRIS_D( 0b00000000 );
```

```
   SET_TRIS_E( 0b00000000 );
   output_bit(PIN_B3,0);
   memory_init();         //Inicialñisacion de control de memorias
   rtc_init();            //Inicialisacion de módulo RTC
   lcd_init();            //Inicialisacion de LCD
   pca9535_init( pca );
   write_rtc( 0x0A, 0x20 ); //configuring register A
   write_rtc( 0x0B, 0x16 ); //configuring register B
   ADS111x_init( ad_rtd, 0b01001100 );      // RTD   <input
   type="checkbox" name="" value=""> config adc continuos covertion,
   canal0, pga=1 and 16bit
  ADS111x_init( ad_long, 0b01001100 );     // LONG   config adc
  continuos covertion, canal0, pga=1 and 16bit
  rtc_read_time( rtc_date );             //Leyendo hora y fecha por 1st
  //*********************************
  // Inicializando variable index
  index = make16( read_rtc(0x21), read_rtc(0x20));
  index = make16( read_rtc(0x21), read_rtc(0x20));
    //*********************************
  while(true)
  {
     restart_wdt();
     //**********************
     usb_task();
     usb_debug_task();
     //**********************
     Tasks();
  }
}
```

Código fuente de TASK.

```
//Libreria donde se realiza un control de las tareas bajo el princio basico de
los RTOS


int1  Task_f1, Task_f2, Task_f3, Task_f4, Task_f5, Task_f6;
int8  Task_cnt1, Task_cnt2, Task_cnt4;
int16 Task_cnt3, Task_cnt5, luz_cnt;

//Funcion de control de tareas
void  func_task( void )
{
  ////////////////////////////////////////////////
  if( Task_cnt1++ >= 200 )     // se ejecuta cada 250ms
  {
    Task_f1=1;
    Task_cnt1=0;
     Task_f6=1;
  }
  ////////////////////////////////////////////////
  if( Task_cnt3++ >= 25 )      //se ejecuta cada 25ms
  {
    Task_f3=1;
    Task_f6=1;
    Task_cnt3=0;
  }
  ////////////////////////////////////////////////
  if( Task_cnt4++ >= 100 )       //se ejecuta cada 100ms
  {
    Task_f4=1;
    Task_cnt4=0;
  }
  ////////////////////////////////////////////////
  if( Task_cnt5++ >= 500 )       //se ejecuta cada 1000ms
  {
    Task_f5=1;
    Task_cnt5=0;
  }


}
```

```
//*****************************************************************************
//Ejecuion de tareas
void  Tasks( void )
{
  //*******************************************
  //Ejecucion de Pantalla
  if( Task_f1 )
  {
      int1 led1;                        // toma de tiempo
      set_timer1(0);
     ////////////
      write_rtc( 0x0A, 0x20 ); //configuring register A
      write_rtc( 0x0B, 0x16 ); //configuring register B
     read_rtc (0x0C);   //Limpiando interrupciones de RTC
     key = key_code();       //escaneo del teclado
     if(key == 5)                       lcd_luz= 1;
     if(key != 0)                       luz_cnt= 0;
     if( luz_cnt++ > 1500 )lcd_luz=0;   //cerca de 5 minutos, 1 min= 300

//*****************************************
      if( !bit_test( make8( var1, 0), 7) )
      ac_off=1;
      else
      ac_off=0;
      if( usb_enum )  led1=1;
      else
      led1= 0;
      if( status == 'R' ) test_on= 1;
      else
      test_on= 0;
      if( lcd_luz ) bit_set(P0,4);
      else
      bit_clear(P0,4);
      if( low_batt )  bit_set(P0,0);
      else
      bit_clear(P0,0);
      if( test_on ) bit_set(P0,2);
     else
     bit_clear(P0,2);
     if( led1 ) bit_set(P0,1);
     else bit_clear(P0,1);
     if( ac_off ) bit_set(P0,3);
     else
     bit_clear(P0,3);    //semsando la alimentacion por AC
```

```
      write_pca9535(pca, P0);

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

  //*** Cálculo de temperatura tomada del TMP175
  tempc = (float)TMPC/16;


  //*** manejo de pantallas**************************************

    screen();
    T_task1 = get_timer1();
    //********
    Task_f1=0;
}

//*********************************************
// actualizacion de reloj
if( Task_f2 )
{
  set_timer1(0);

  rtc_read_time( rtc_date );
  sec = rtc_date[0];
  min = rtc_date[1];
  hour = rtc_date[2];
  day = rtc_date[3];
  month = rtc_date[4];
  year = rtc_date[5];
  //output_toggle(PIN_B5);

  T_task2 = get_timer1();
  //************
  Task_f2 = 0;
}

//*********************************************
// Transferencias de datos por usb de dato por USB
if( Task_f3 )
{
  set_timer1(0);
  usb_func();
```

```
   T_task3 = get_timer1();
   //**********
   Task_f3=0;
}

//*********************************************
// Lectura el ADC del dato de longitud
if( Task_f4 )
{
    set_timer1(0);
    Fil_L[1]= Fil_L[0];
    Fil_L[0]= ADS111x_data( ad_long );
    ad_L=(Fil_L[1]+Fil_L[0])/2;
    if( ad_L < ad_min )            ad_L= ad_min;
    if( ad_L > ad_max-10)       ad_L= ad_max;
    longitud= (float)32*(ad_L - ad_min)/ad_max;
   if( status=='R' )
   {

    if( index >= N_muestras )
        status= 'S';
    if( ++T_sample_cnt >= N_muestreo )// antes era T_sample
      {
        T_sample_cnt= 0;
        write_muestra();
      }
   }
   T_task4 = get_timer1();
   //************
   Task_f4=0;
}

//*********************************************
// Lectura el ADC del dato de temperatura y manejo de control PI
if( Task_f5 )
{
   set_timer1(0);
   //TMPC = TMP175_temp(0x49);     //sensor tmp175
      Fil_T[1]= Fil_T[0];
      Fil_T[0]= ADS111x_data(ad_rtd);
      ad_T= (Fil_T[1] + Fil_T[0])/2;

    //Cálculo de la temperatura de RTD apartir del ADC del PIC **********
```

```
   temp_RTD = (float)100*(ad_T - 6400 )/( 25600 );
   if( temp_RTD > 100 )
   temp_RTD = 100.5;
   if( temp_RTD < 0 )
   temp_RTD = -0.5;

//************************************************************
   Pb= 15;                          //atencion
   P_i= 3;
//control pi para el calentador ********************************
 if( CtrlTemp == 'V' )
{
    if(cnt_pi==1)
     {
          SP= (int16)10*setTemp;
          PV= (int16)10*temp_RTD;
          error_pi= SP - PV;     // E = SP - PV
          if( labs(error_pi) < 30)// 3 grados de error actua el integral
           {
              saco= saco + error_pi;
              Psaco= (int16)(P_i*saco)/100;
                                                                  }
           else
                                                                      {
           saco=0;
           Psaco=0;
            }
          banda_pi= (int16)Pb*SP/100;
          out_pi= 100*(error_pi + Psaco)/banda_pi;
          if( out_pi>100 ) out_pi=100;
          if( out_pi<0 )              out_pi=0;
          cmp_pi= out_pi/10;
             }
          if( cnt_pi>=10 )            cnt_pi=0;
          if( cnt_pi>cmp_pi )
          output_bit(PIN_B3, 0);
          else
          output_bit(PIN_B3, 1);
          cnt_pi++;
       }
      else
        {
           output_bit(PIN_B3, 0);
                                                                  }
```

```
    //*************************************************************
    T_task5 = get_timer1();
    //************
    Task_f5=0;
  }
    //*********************************************

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

    // Monitoreo de Batería de respaldo

    if( Task_f6 )
    {
       set_timer1(0);
       battery_monitor();
      T_task6 = get_timer1();
      Task_f6=0;
                                              }

}
```

Código fuente de Pantallas.

```
//Libreria donde se realiza el manejo de las pantallas del lcd

int8 lcd_x, lcd_y, lcd_i=1, i_time;
int8 T_time_y=1;
int1 lcd_f1;
int8 scr_tmp;
int16 msn_error, out_main_scr;

////////////////////////////////////////////////////////////////////////////
void screen_init(void);
void screen_main(void);
void screen_menu(void);
void screen_setADC(void);
void screen_task_time(void);
void screen_staBatt(void);
void screen_status_prueba(void);
void screen_parametros(void);
void screen_muestreo(void);
void screen_datos(void);
void screen_error(void);
////////////////////////////////////////////////////////////////////////////

//***************************************************************************
//    Pantallas
void screen( void )
{
  if( scr!=1 )
  {
    if( key!=0 )   out_main_scr= 0;
    if(out_main_scr++ == 900)        // para que si esta fuera de la pantalla
principal
    {                                                          // por mas de
3min regrese a esta AUTOmaticamente
      //scr = 1;
                  key= 3;
    }
  }

  if (scr != scr_tmp) // clear screen cuando hay cambio de pantalla
  printf (lcd_putc, "\f");
```

```
scr_tmp = scr;

SWITCH(scr)
{
  case 0:
    screen_init();
  break;

  case 1:
    screen_main();
    out_main_scr = 0;
  break;

  case 2:
    screen_menu();
  break;

  case 3:
    screen_setADC();
  break;

  case 4:
    screen_task_time();
  break;

  case 5:
    screen_staBatt();
  break;

  case 6:
    screen_status_prueba();
  break;

  case 7:
    screen_parametros();
  break;

  case 8:
    screen_muestreo();
  break;

  case 9:
    screen_datos();
  break;
```

```
      case 10:
        screen_error();
      break;

      DEFAULT:
      break;
    }
}

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

//*****************************************************************************
void screen_init(void)     // scr= 0
{
  if (lcd_i == 1)
    lcd_f1 = 1;

  if (lcd_i == 4)
    lcd_f1 = 0;

  printf (lcd_putc, "\f");
  lcd_gotoxy (lcd_i, 1);
       printf (lcd_putc,"* * *** * **  ");
  lcd_gotoxy (lcd_i, 2);
       printf (lcd_putc,"**  *   * * * ");
  lcd_gotoxy (lcd_i, 3);
       printf (lcd_putc,"**  *   * * * ");
  lcd_gotoxy (lcd_i, 4);
       printf (lcd_putc,"* * *** * **  ");

  if (lcd_f1)
    lcd_i++;

  else
  lcd_i--;

  if (key)
  {
    printf (lcd_putc, "\f");
    scr = 1;
  }
  else
  scr = 0;
```

```
}

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

//*****************************************************************************
void screen_main(void)     // scr= 1
{
  lcd_gotoxy(1,1);
   printf(lcd_putc,"=====SYSADQ=====");
   lcd_gotoxy (7, 2);
  printf(lcd_putc,"%02u/%02u/2%03u", day, month, year);
   lcd_gotoxy (9, 3);
   printf(lcd_putc,"%02u:%02u:%02u\n", hour, min, sec);
   printf(lcd_putc,"Prueba en: ");
   if( status == 'R' )
     printf( lcd_putc,"RUN ");
  else
     printf( lcd_putc,"STOP");
  //***********************************
  if( key==1 )    scr= 2;
   if( key==3 )   scr= 6;
  if( key==4 )    scr= 7;
  if( key==5 )    scr= 4;
  lcd_y = 2;
}

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

//*****************************************************************************
void screen_menu(void)     // scr= 2
{
  lcd_gotoxy (1, 1);
  printf (lcd_putc, "******MENU******\n");
  printf (lcd_putc, " Estado de Batt\n");
   printf (lcd_putc, " Ajuste del ADC\n");
  printf (lcd_putc, " OUT ON");

  //***********************************
  if(key == 2)    scr=1; // hacia screen main
  if(lcd_y==3 && key == 1)    scr = 3;
  if(lcd_y==2 && key==1)                scr = 5;
  if(lcd_y==4 && key==1) { output_toggle(PIN_B3); send_dat=1;
export_index= 0;  scr= 1; }
```

```
   if (key == 3) lcd_y++;
   if (key == 4) lcd_y--;
   if (lcd_y==1 && key == 4) lcd_y = 4;
   if (lcd_y==5 && key == 3) lcd_y = 2;
   lcd_gotoxy (1, lcd_y);
   printf (lcd_putc, "%c", 0x7E);
}

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

//*****************************************************************************
void screen_setADC(void)     //scr= 3
{
    int16   ADtmp;
    ADtmp= ADS111x_data( ad_long );
    lcd_gotoxy(1,1);
   printf(lcd_putc,"-Ajuste del ADC-\n");
   printf (lcd_putc," ADC  0mm: ");
      if(lcd_y==2 && key==1)
      {   printf( lcd_putc,"%04LX*\n", ADtmp);   ad_min= ADtmp;  }
      else
         printf( lcd_putc,"%04LX\n", ad_min);
   printf (lcd_putc," ADC 32mm: ");
   if(lcd_y==3 && key==1)
      {   printf( lcd_putc,"%04LX*\n", ADtmp);   ad_max= ADtmp;  }
      else
         printf( lcd_putc,"%04LX\n", ad_max);
   printf (lcd_putc," Salir");
    lcd_gotoxy (1, lcd_y);
   printf (lcd_putc, "%c", 0x7E);

   //************************************
   if(key==1 && lcd_y==4 )
   {
      writEE_int16( ee_ad_min, ad_min );
      writEE_int16( ee_ad_max, ad_max );
      scr= 2;
   }
   if(key==2)  scr= 5;
   if(key == 3) lcd_y++;
   if(key == 4) lcd_y--;
   if(lcd_y==1 && key==4) lcd_y = 4;
   if(lcd_y==5 && key==3) lcd_y = 2;
}
```

```
//*****************************************************************************

//*****************************************************************************
void screen_task_time(void)      // scr= 4
{
  lcd_gotoxy (1, 1);
  printf (lcd_putc, "T en micro seg\n\n");

  SWITCH (T_time_y)
  {
    case 1:
      printf (lcd_putc, "Task1 %lu    ", T_task1 / 3);
    break;

    case 2:
      printf (lcd_putc, "Task2 %lu    ", T_task2 / 3);
    break;

    case 3:
      printf (lcd_putc, "Task3 %lu    ", T_task3 / 3);
    break;

    case 4:
      printf (lcd_putc, "Task4 %lu    ", T_task4 / 3);
    break;

    case 5:
      printf (lcd_putc, "Task5 %lu    ", T_task5 / 3);
    break;

    case 6:
      printf (lcd_putc, "Task6 %lu    ", T_task6 / 3);
    break;

    DEFAULT:
    break;
}
  lcd_gotoxy(1,4);
  printf( lcd_putc,"%s", usb_dat);

  if(key == 3) T_time_y--;
  if(key == 4) T_time_y++;
  if(T_time_y==7 && key==4) T_time_y = 1;
  if(T_time_y==0 && key==3) T_time_y = 6;
```

```
  if(key == 2) scr = 1;
}

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

//*****************************************************************************
void screen_staBatt(void)      // scr=5
{
   lcd_gotoxy(1,1);
  printf(lcd_putc,"Voltage Batería\n");
   printf(lcd_putc,"\n  Batt: %2.1fV ", v_bat);
  //**********************************
  if(key == 2)    scr=2; // hacia screen menu
}

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

//*****************************************************************************
void screen_status_prueba(void)     // scr= 6
{
  lcd_gotoxy(1,1);
   printf(lcd_putc,"=====SYSADQ=====\n");
   printf(lcd_putc,"T act: %2.1f%cC \n", temp_RTD,0xdf);
   printf(lcd_putc,"Strain: %3.2fmm \n", longitud);
   printf(lcd_putc,"Índice: %lu    ", index);

   //*************************
   if( key==1 ) scr= 2;
  if( key==3 )  scr= 7;
  if( key==4 )  scr= 1;
}
//*****************************************************************************

//*****************************************************************************
void screen_parametros(void)     // scr= 7
{
  lcd_gotoxy(1,1);
   printf(lcd_putc,"=====SYSADQ=====\n");
   printf(lcd_putc,"T fij: %u%cC \n", setTemp,0xdf);
   printf(lcd_putc,"Ctrl Temp: ");
   if( CtrlTemp=='V' )   printf(lcd_putc,"ON\n"); else
   printf(lcd_putc,"OFF\n");
   printf(lcd_putc,"Muestras: %lu    ", N_muestras);
```

```
    //*************************
    if( key==1 )  scr= 2;
    if( key==3 )  scr= 1;
   if( key==4 )  scr= 6;
}
//*****************************************************************************

//*****************************************************************************
void screen_muestreo(void)    // scr= 8
{
    lcd_gotoxy(1,1);
    printf(lcd_putc,"-T. de Muestreo-\n");
    printf(lcd_putc," Muestras: %lu", muestras);
    printf(lcd_putc," Período: %3.1fs",muestreo);
    printf(lcd_putc," Salir");
    lcd_gotoxy (1, lcd_y);
   printf (lcd_putc, "%c", 0x7E);

    if(lcd_y==5 && key==1);


}
//*****************************************************************************

//*****************************************************************************
void screen_datos(void)    // scr= 9
{
    lcd_gotoxy(1,1);
    printf(lcd_putc,"Muestras: %lu\n", N_muestras);
    printf(lcd_putc,"Muestreo: %3.1w\n", N_muestreo);
    printf(lcd_putc,"Set Temp: %u\n", setTemp);
    printf(lcd_putc,"Ctrl Temp: %c ", Ctrltemp);

    if( key==2 )   scr=1;
}
//*****************************************************************************

//*****************************************************************************
void screen_error(void)    // scr= 10
{

}
//*****************************************************************************
```

Código fuente del LCD.

```
// As defined in the following structure the pin connection is as follows:
//     D0  enable
//     D1  rs
//     D2  rw
//     D4  D4
//     D5  D5
//     D6  D6
//     D7  D7
//
//   LCD pins D0-D3 are not used and PIC D3 is not used.

// todas estas tres señales son salidas
#bit  enable = 0xf84.0     // Pin E0 is defined
#bit  rs     = 0xf84.1     // Pin E1 is defined
#bit  rw     = 0xf84.2     // Pin E2 is defined

struct lcd_pin_map {                // This structure is overlayed
          //int     data : 4;
          BOOLEAN enable;           // on to an I/O port to gain
          BOOLEAN rs;               // access to the LCD pins.
          BOOLEAN rw;               // The bits are allocated from
          BOOLEAN unused;           // low order up.  ENABLE will
          int     data : 4;         // be pin B0.
       } lcd;


  //define lcd data port
  #byte lcd = 0xF83       // the port D is defined

  //define tris lcd port
  #define set_tris_lcd(x) set_tris_d(x)



#define lcd_type 2          // 0=5x7, 1=5x10, 2=2 lines
#define lcd_line_two 0x40    // LCD RAM address for the second line


BYTE const LCD_INIT_STRING[4] = {0x20 | (lcd_type << 2), 0xc, 1, 6};
                     // These bytes need to be sent to the LCD
                     // to start it up.
```

```
                    // The following are used for setting
                    // the I/O port direction register.


//*****************************select nibble**********************************
struct lcd_pin_map const LCD_WRITE = {0,0,0,0,0}; // For write mode all
pins are out
struct lcd_pin_map const LCD_READ = {0,0,0,0,15}; // For read mode
data pins are in

//struct lcd_pin_map const LCD_WRITE = {0,0,0,0,1}; // For write mode
all pins are out
//struct lcd_pin_map const LCD_READ = {15,0,0,0,1}; // For read mode
data pins are in

BYTE lcdline;

BYTE lcd_read_byte() {
     BYTE low,high;
     set_tris_lcd(LCD_READ);
     rw=1;
     delay_cycles(1);
     enable = 1;
     delay_cycles(1);
     high = lcd.data;
     enable = 0;
     delay_cycles(1);
     enable = 1;
     delay_us(1);
     low = lcd.data;
     enable = 0;
     set_tris_lcd(LCD_WRITE);
     return( (high<<4) | low);
}


void lcd_send_nibble( BYTE n ) {
     lcd.data = n;
     delay_cycles(1);
     enable = 1;
     delay_us(2);
     enable = 0;
}
```

```
void lcd_send_byte( BYTE address, BYTE n ) {

      rs = 0;
      while ( bit_test(lcd_read_byte(),7) ) ;
      rs = address;
      delay_cycles(1);
      rw = 0;
      delay_cycles(1);
      enable = 0;
      lcd_send_nibble(n >> 4);
      lcd_send_nibble(n & 0xf);
}


void lcd_init() {
    BYTE i;
    set_tris_lcd(LCD_WRITE);
    rs = 0;
    rw = 0;
    enable = 0;
    delay_ms(15);
    for(i=1;i<=3;++i) {
       lcd_send_nibble(3);
       delay_ms(5);
    }
    lcd_send_nibble(2);
    for(i=0;i<=3;++i)
       lcd_send_byte(0,LCD_INIT_STRING[i]);
}


void lcd_gotoxy( BYTE x, BYTE y) {
   BYTE address;

   //if(y!=1)
   //  address=lcd_line_two;
   //else
   //  address=0;
   switch(y) {
     case 1 : address=0x00; lcdline=1; break;
     case 2 : address=0x40; lcdline=2; break;
     case 3 : address=0x10; lcdline=3;break;
```

```
    case 4 : address=0x50;break;
   }
   address+=x-1;
   lcd_send_byte(0,0x80|address);
}

void lcd_putc( char c) {
   switch (c) {
     case '\f'   : lcd_send_byte(0,1);
                   lcdline=1;
                   delay_ms(2);
                   break;
     //case '\n'   : lcd_gotoxy(1,2);
        //           break;
     case '\n'   : lcd_gotoxy(1,++lcdline);        break;
     case '\b'   : lcd_send_byte(0,0x10);
                   break;
     default     : lcd_send_byte(1,c);
                   break;
   }
}

char lcd_getc( BYTE x, BYTE y) {
   char value;

    lcd_gotoxy(x,y);
    while ( bit_test(lcd_read_byte(),7) ); // wait until busy flag is low
    rs=1;
    value = lcd_read_byte();
    rs=0;
    return(value);
                                          }
```

Código fuente para el control del extensor de puertos PCA9535.

```
void pca9535_init( int8 address )
{

   address = address << 1;
   bit_clear( address, 0 );

   i2c_start();         // Start
   i2c_write(address);  // Device address
   i2c_write(0x02);     // Control register
    i2c_write(0x00);                // data p0
    i2c_write(0x00);                // data p1
   i2c_stop();          // Stop

    delay_us(500);

    i2c_start();         // Start
   i2c_write(address);  // Device address
   i2c_write(0x04);     // Control register
    i2c_write(0x00);                // polarity inv p0
    i2c_write(0x00);                // polarity inv p1
   i2c_stop();          // Stop

    delay_us(500);

    i2c_start();         // Start
   i2c_write(address);  // Device address
   i2c_write(0x06);     // Control register
    i2c_write(0x80);                // config p0
    i2c_write(0xff);                // config p1
   i2c_stop();          // Stop
}


int16 read_pca9535(int8 address )
{
   int8 hi,lo;

   address = address << 1;
   bit_clear( address, 0 );

   i2c_start();               // Start
```

```
   i2c_write(address);     // Device address
   i2c_write(0x01);        // Config register
   i2c_start();
   bit_set( address, 0 );
   i2c_write(address);   // reStart
   hi = i2c_read();
   lo = i2c_read();
   i2c_stop();

   return  make16(hi,lo);

}

void write_pca9535(int8 address, int8 data)
{
    address = address << 1;
   bit_clear( address, 0 );

    i2c_start();              // Start
   i2c_write(address);     // Device address
   i2c_write(0x02);        // Config register
    i2c_write(data);
    i2c_stop();
}
```

Código fuente para controlar los IC de memoria UT61256.

```
#bit  mCE1=  0xf81.4    //B4
#bit  mCE2=  0xf81.5    //B5
#bit  mCE3=  0xf82.6    //C6
#bit  mCE4=  0xf82.7    //C7
#bit  mCE5=  0xf81.6    //B6
#bit  mCE6=  0xf81.7    //B7

#bit  mOE=   0xf84.1    //E1
#bit  mWE=   0xf84.2    //E2

#bit  clk1=  0xf80.4    //A4 para el lsb
#bit  clk2=  0xf80.5    //A5 para el msb

//*****************************************************************************
void  memory_init(void)
{
   mCE1= 1;
    mCE2= 1;
    mCE3= 1;
    mCE4= 1;
    mCE5= 1;
    mCE6= 1;

   clk1= 0;
   clk2= 0;
}
//*****************************************************************************

//*****************************************************************************
void memory_write(char *data, int8 n, int32 address )
{
   int8 i=0;

   clk1= 0;
   clk2= 0;
   mCE1= 1;
   mWE= 1;
   mOE= 1;
   SET_TRIS_D( 0x00 );
   do
   {
```

```
output_d( make8(address,0) );    // Gets LSB of address
clk1= 1;
output_d( make8(address,1) );    // Gets MSB of address
clk2= 1;
output_d( data[i] );
        if(address < 32768)
        {
                mCE1= 0;
        mWE= 0;
        mWE= 1;
        mCE1= 1;
        }
        if(address>=32768 && address<65536)
        {
                mCE2= 0;
        mWE= 0;
        mWE= 1;
        mCE2= 1;
        }
        if(address>=65536 && address<98304)
        {
                mCE3= 0;
        mWE= 0;
        mWE= 1;
        mCE3= 1;
        }
        if(address>=98304 && address<131072)
        {
                mCE4= 0;
        mWE= 0;
        mWE= 1;
        mCE4= 1;
        }
        if(address>=131072 && address<163840)
        {
                mCE5= 0;
        mWE= 0;
        mWE= 1;
        mCE5= 1;
        }
        if(address>=163840 && address<196608)
        {
                mCE6= 0;
        mWE= 0;
```

```
            mWE= 1;
            mCE6= 1;
            }
      //***************************
      address++;
      clk1= 0;
      clk2= 0;
   }while(++i<n);
}
//*****************************************************************************

//*****************************************************************************
void memory_read(char *data, int8 n, int32 address)
{
   int8 i=0;

   clk1= 0;
   clk2= 0;
   mCE1= 1;
   mWE= 1;
   mOE= 1;
   do
   {
      SET_TRIS_D( 0x00 );
      output_d( make8(address,0) );    // Gets LSB of address
      clk1= 1;
      output_d( make8(address,1) );    // Gets MSB of address
      clk2= 1;
      if(address < 32768)
            {
                  mCE1= 0;
                  mOE= 0;
            SET_TRIS_D( 0xff );
            data[i]= input_d();
            mOE= 1;
            mCE1= 1;
            }
            if(address>=32768 && address<65536)
            {
                  mCE2= 0;
                  mOE= 0;
            SET_TRIS_D( 0xff );
            data[i]= input_d();
            mOE= 1;
```

```
            mCE2= 1;
            }
            if(address>=65536 && address<98304)
            {
                    mCE3= 0;
                    mOE= 0;
            SET_TRIS_D( 0xff );
            data[i]= input_d();
            mOE= 1;
            mCE3= 1;
            }
            if(address>=98304 && address<131072)
            {
                    mCE4= 0;
                    mOE= 0;
            SET_TRIS_D( 0xff );
            data[i]= input_d();
            mOE= 1;
            mCE4= 1;
            }
            if(address>=131072 && address<163840)
            {
                    mCE5= 0;
                    mOE= 0;
            SET_TRIS_D( 0xff );
            data[i]= input_d();
            mOE= 1;
            mCE5= 1;
            }
            if(address>=163840 && address<196608)
            {
                    mCE6= 0;
                    mOE= 0;
            SET_TRIS_D( 0xff );
            data[i]= input_d();
            mOE= 1;
            mCE6= 1;
            }
        //***************************
        address++;
        clk1= 0;
        clk2= 0;
      }while(++i<n);
}
```

Código fuente del RTC BQ328x

```
#bit  CS   = 0xf80.1   // A1
#bit  AS   = 0xf84.1   // E1
#bit  DS   = 0xf84.2   // E2
#bit  rRW  = 0xf80.2   // A2


#byte port = 0xf83      //potr D
//#byte tris = 0xf95      //tris D

char  rtc_date[6];

//***************************************************************************
int8 read_rtc( int8 address )
{
  int8  data;


  set_tris_d( 0x00 );
  rRW = 1;
  DS = 1;
  port = address;
  AS = 1;
  //delay_us(1);
  CS = 0;
  //delay_us(1);
  AS = 0;
  //delay_us(1);
  set_tris_d( 0xff );
  DS = 0;
  //delay_us(1);
  port = 0;
  data = port;
  DS = 1;
  CS = 1;
  AS = 1;
  return data;
}


//***************************************************************************
```

```
void write_rtc( int8 address, int8 data )
{
  set_tris_d( 0x00 );
  AS = 0;
  rRW = 1;
  DS = 1;
  port = address;   //address
  AS = 1;
  delay_us(1);
  CS = 0;
  delay_us( 1 );
  AS = 0;
  delay_us(1);
  port = data;      // write
  rRW = 0;
  delay_us(1);
  rRW = 1;
  CS = 1;
  AS = 1;
}


//***************************************************************************
void rtc_read_time(char *s)
{
  //  seconds ********************
  s[0] = read_rtc( 0x00 );      // seconds

  //  minutes ********************
  s[1] = read_rtc( 0x02 );      // minutes

  //  hours **********************
  s[2] = read_rtc( 0x04 );      // hours

  //  daymonth *******************
  s[3] = read_rtc( 0x07 );      // daymonth

  //  month **********************
  s[4] = read_rtc( 0x08 );      // month

  //  year ***********************
  s[5] = read_rtc( 0x09 );      // year

  // set UIE bit
```

```
  write_rtc( 0x0B, 0x16 );
  read_rtc( 0x0C );
}

//***************************************************************************
void rtc_write_time(int8 year, int8 month, int8 daymonth, int8 hours, int8
minutes, int8 seconds)
{
  //  seconds ******************
  write_rtc( 0x00, seconds );

  //  minutes ******************
  write_rtc( 0x02, minutes );

  //  hours *********************X
  write_rtc( 0x04, hours );

  //  daymonth *****************
  write_rtc( 0x07, daymonth );

  //  month ********************
  write_rtc( 0x08, month );

  //  year *********************
  write_rtc( 0x09, year );
}


//***************************************************************************
void rtc_set_swf(int8 f)
{
  int8  a;
  a = read_rtc( 0x0A );
  a = a & 0xF0;
  f = f & 0x0F;
  a = a | f;
  write_rtc( 0x0A, a );
}


//***************************************************************************
void rtc_init(void)
{
```

```
    CS = 1;
    DS = 0;
    rRW = 0;
    AS = 0;
}

//*****************************************************************************
```

Código fuente del conversor ADC111x

```
void ADS111x_init( int8 address, int8 config )
{
  // configurado con Shutdown mode off y resolucion de 16 bit
  address = address << 1;
  bit_clear( address, 0 );
  i2c_start();         // Start
  i2c_write(address);     // Device address
  i2c_write(config);     //   Configuration
  i2c_stop();          // Stop
}



int16 ADS111x_data(int8 address )
{
  int8 hi,lo,reg;

  address = address << 1;
  bit_set( address, 0 );
  i2c_start();         // Start
  i2c_write(address);     // Device address
  hi = i2c_read();
  lo = i2c_read();
  reg = i2c_read();
  i2c_stop();

  return  make16(hi,lo);

}
```

Código fuente de las interrupciones del hardware

```
#int_TIMER2
void TIMER2_isr()
{
  func_task();
}

#int_EXT1
void EXT1_isr()
{
  Task_f2 = 1;
}
/*
#int_RDA
RDA_isr()
{

}

#int_TBE
TBE_isr()
{

}

#int_LOWVOLT
LOWVOLT_isr()
{

}
*/
```

Código fuente de las funciones de proposito específico

```
//*****************************************************************************

int8  key_code( void )
{

  var1= read_pca9535(pca);
   a = make8( var1, 1);
  a = ~a;
  a = a & 0x1f;
  switch (a)
  {
    case 1:
          a= 1;    // Menu
          break;

    case 2:
          a= 2;    // return
          break;

    case 4:
          a= 3;    //Down
          break;

    case 8:
          a= 4;    //Up
          break;

    case 16:
          a= 5;    //Clear
          break;

    default:
          a= 0;
          break;
  }

  if( b==0 && a>0 )
  {
    b=a;
    return a;
  }
```

```
    else
    {
      b=a;
      return 0;
    }
  }

  //*****************************************************************************
****
  void write_muestra(void)
  {
    char  sample[10], test_sample[10];
    int16 tmp;
    int32 temp;
    int8  j,k;

    //memset(sample, 0, sizeof(sample));     //Limpiando array
    //memset(test_sample, 0, sizeof(test_sample)); //Limpiando array

    index++;

    write_rtc(0x20, make8(index, 0));    //guardando el index en el rtc
    write_rtc(0x21, make8(index, 1));

    sample[0]= make8(index, 0);    //escribiendo index
    sample[1]= make8(index, 1);

    temp= (int32)index*N_muestreo;   //escribiendo el tiempo
    sample[2]= make8(temp, 0);
    sample[3]= make8(temp, 1);
    sample[4]= make8(temp, 2);
    sample[5]= make8(temp, 3);

    tmp= (int16)100*longitud;
    sample[6]= make8(tmp, 0);      //escribiendo Longitud
    sample[7]= make8(tmp, 1);

    tmp = (int16)10*temp_rtd;
    sample[8]= make8(tmp, 0);      //escribiendo tempetatura
    sample[9]= make8(tmp, 1);
    tmp=0;
    memory_write(sample, 10, (index-1)*10 );
  }
```

```
        //*************************************************************************
****
        int8 ci, cj, ck;



        //*************************************************************************
****
        void export_datos( void )
        {
           char  sample[10];
            int8    len, ret= 13;
            int16 ee_i=0;

            for( i=0; i<64; i++)
                        out_data[i]= 0;
            sprintf(out_data "%cl,%lu%c%c",2, index,26,0x0d);
            len= strlen(out_data);
            //returns TRUE if this endpoint's IN buffer (PIC-PC) is empty and
ready
                    while( !usb_tbe(USB_CDC_DATA_IN_ENDPOINT) )
restart_wdt();
                    usb_put_packet(USB_CDC_DATA_IN_ENDPOINT,
out_data, len, USB_DTS_TOGGLE);

            ////////////////////////////////
           do
            {
            restart_wdt();
                    for( i=0; i<64; i++)
                            out_data[i]= 0;

                    memory_read(sample, 10, (ee_i++)*10);
                    sprintf(out_data
"%cR,%04lu,%07lu,%02.2w,%03.1w%c%c",2,
                   make16(sample[1], sample[0]),
                   make32(sample[5], sample[4], sample[3], sample[2]),
                   make16(sample[7], sample[6]),
                   make16(sample[9], sample[8]),26, 0x0d );
                    len= strlen(out_data);

                    //returns TRUE if this endpoint's IN buffer (PIC-PC) is
empty and ready
```

```
                    while( !usb_tbe(USB_CDC_DATA_IN_ENDPOINT) )
restart_wdt();
                    usb_put_packet(USB_CDC_DATA_IN_ENDPOINT,
out_data, len, USB_DTS_TOGGLE);


           }while(ee_i < index);
            ee_i=0;
            ////////////////////////////////
            for( i=0; i<64; i++)
                        out_data[i]= 0;
            sprintf(out_data "%cQ,0000%c%c",2,26,0x0d);
            len= strlen(out_data);
            //returns TRUE if this endpoint's IN buffer (PIC-PC) is empty and
ready
                    while( !usb_tbe(USB_CDC_DATA_IN_ENDPOINT) )
restart_wdt();
                    usb_put_packet(USB_CDC_DATA_IN_ENDPOINT,
out_data, len, USB_DTS_TOGGLE);
        }
        //*************************************************************************
****

        //*************************************************************************
****
        void config_test(void)
        {
            char    cmdtmp[4][15], x, stctmp[15];
            int8    i, j, k, len;

            for(i=0; i<4; i++)
                    for(j=0; j<15; j++)
                            cmdtmp[i][j]=0;

            len= strlen(in_data);
            i=0;
            for(k=0; k<len; k++)
            {
                    x=in_data[k];
                    if( x==',' )
                    {
                            i++;
                            j=0;
                    }
```

```
                if(i==1 && isdigit(x))
                        cmdtmp[0][j++]= x;
                if(i==2 &&(isdigit(x)||x=='.'))
                        cmdtmp[1][j++]= x;
                if(i==3 &&(isdigit(x)||x=='.'))
                        cmdtmp[2][j++]= x;
                if(i==4 && isalpha(x))
                        cmdtmp[3][j++]= x;
            }
            for(i=0; i<15; i++)
                    stctmp[i]= cmdtmp[0][i];
            N_muestras= atol( stctmp );

            for(i=0; i<15; i++)
                    stctmp[i]= cmdtmp[1][i];
            N_muestreo= (int16)10*atof( stctmp );

            for(i=0; i<15; i++)
                    stctmp[i]= cmdtmp[2][i];
            setTemp= atoi( stctmp );

            CtrlTemp= cmdtmp[3][0];

            writEE_int16( ee_muestras, N_muestras);
            writEE_int16( ee_muestreo, N_muestreo);
            write_eeprom( ee_setTemp, setTemp);
            write_eeprom( ee_CtrlTemp, CtrlTemp);



        }
        //*************************************************************************
****


        //*************************************************************************
****
        void usb_func(void)
        {
          int8  len, len_IN;
           char    dat;
          if(usb_cdc_connected())
          {
            for(i=0; i<64; i++)
```

```
            {
              in_data[i]= 0;
              out_data[i]= 0;
            }
                  len_IN= 0;
                  ci=0;
            if(usb_cdc_kbhit())
            {
                  do
                        {
                              dat= usb_cdc_getc();
                              in_data[ci++]= dat;
                        }while( dat != '&' );
            }

                        switch( in_data[1] )
                  {
                        case 'A':   //entrega datos al PC-HOST para
                        actualizar pantallas

                        sprintf( out_data,
"%c%3.1f,%2.2f,%u,%u,%u,%u%c%c", 2, temp_rtd, longitud, low_batt,
test_on, ac_off, cont_Temp, 26, 0x0d);
                        len = strlen(out_data);

                                          //returns TRUE if this endpoint's IN
buffer (PIC-PC) is empty and ready
                                                  while(
!usb_tbe(USB_CDC_DATA_IN_ENDPOINT) ) restart_wdt();

   usb_put_packet(USB_CDC_DATA_IN_ENDPOINT, out_data, len,
USB_DTS_TOGGLE);
                        break;

                        case 'H':   //actualizar hora y fecha en el dispositivo
                        write_rtc (0x0B, 0x06); //Desabilitando actualizacion de
RTC
                        day= 10*(in_data[2]-0x30) + in_data[3]-0x30;
                        month= 10*(in_data[5]-0x30) + in_data[6]-0x30;
                        year= 10*(in_data[8]-0x30) + in_data[9]-0x30;
                        hour= 10*(in_data[11]-0x30) + in_data[12]-0x30;
                        min= 10*(in_data[14]-0x30) + in_data[15]-0x30;
                        sec= 10*(in_data[17]-0x30) + in_data[18]-0x30;
                        // escribiendo datos en el RTC
```

```
                rtc_write_time (year, month, day, hour, min, sec);
                write_rtc (0x0B, 0x16); //Habilitando actualizacion de RTC
                read_rtc (0x0C);
                break;

                        case 'E':
                export_datos( );
                        break;

                        case 'C':
                config_test( );
                        break;

                        case 'T':
                status= in_data[2];
                                if( status== 'R' )
                                        index=0;

                                write_eeprom( ee_status, status);
                        break;

                default:
                break;
                }
          }
        }
        //*************************************************************************
****

        //*************************************************************************
****
        void        battery_monitor( void )
        {
           int16 adBatt;
           fil_B[3]= fil_B[2];
           fil_B[2]= fil_B[1];
           fil_B[1]= fil_B[0];
           fil_B[0]= read_adc();
           adBatt= fil_B[3]+fil_B[2]+fil_B[1]+fil_B[0];
           adBatt=  adBatt/4;
           v_bat= (float) 10*adBatt/1023;

           if( V_bat > 7 )
                   low_batt= 0;
```

```
            if( V_bat < 6.5)
                    low_batt= 1;
        }
        //*************************************************************************
****
```

C. Código fuente de la aplicación (software).

Código fuente de la pantalla principal

```
Const APPNAME = "CreepSoft"
Dim ImpRun As Boolean   'Variable que induca si la importacion de datos
esta en proceso
Dim maxIndex As Long
Dim senDat As Variant
Dim allDat As String
Dim Time_ms As Long
Dim CreepON As Boolean


Private Sub ActHF_Click()   'Comando de actulizacion de Hora y fecha en
DataCreep
   Dim Fecha As Variant

   Fecha = Format$(Now, "dd/mm/yy,hh:mm:ss")
   Fecha = "*H" & Fecha & "&"
   MSComm1.Output = Fecha
End Sub


Private Sub cfg_test_Click()
   configTest.Show vbModal, Me
End Sub


Private Sub selComm_Click()
   MSComm1.CommPort = selComm.ListIndex + 1
End Sub


Private Sub conect_Click()  'Conectar a circuito DataCreep

   On Error Resume Next

   MSComm1.PortOpen = Not MSComm1.PortOpen
   If Err Then MsgBox Error$, 48
   If MSComm1.PortOpen = True Then
      commON.Active = True
      selComm.Enabled = False
```

```
        cfg_test.Enabled = True
        TestON.Enabled = True
        conect.Caption = "Desconectar"
    Else
        commON.Active = False
        selComm.Enabled = True
        cfg_test.Enabled = False
        TestON.Enabled = False
        conect.Caption = "Conectar"
    End If

End Sub


Private Sub ExitCreep_Click()   'Salir de programa
    Unload Me
End Sub



Private Sub Form_Load()

    conect.Checked = False
    ImpRun = False
    'DatOnline.Enabled = False
    ActHF.Enabled = False
    ImportDat.Enabled = False
    'MSComm1.RThreshold = 1
    BarraEstado.Top = 3960
    CreepForm.Height = 5115
    sendCfg = False
    cfg_test.Enabled = False
    TestON.Enabled = False
    Call GetSettings
    With selComm
        .AddItem "Comm 1", 1
        .AddItem "Comm 2", 2
        .AddItem "Comm 3", 3
        .AddItem "Comm 4", 4
        .AddItem "Comm 5", 5
        .AddItem "Comm 6", 6
        .AddItem "Comm 7", 7
        .AddItem "Comm 8", 8
        .AddItem "Comm 9", 9
```

```
        .AddItem "Comm 10", 10
        .AddItem "Comm 11", 11
        .AddItem "Comm 12", 12
        .AddItem "Comm 13", 13
        .AddItem "Comm 14", 14
        .AddItem "Comm 15", 15
        .AddItem "Comm 16", 16
        .ListIndex = 3
    End With
    MSComm1.CommPort = selComm.ListIndex + 1
End Sub

Private Sub Form_Unload(Cancel As Integer)
    If MSComm1.PortOpen = True Then
        MSComm1.PortOpen = False
    End If
    Call SaveSettings
End Sub


Private Sub ImportDat_Click()   'Importando datos de datacreep

    Timer1.Enabled = False
    Timer2.Enabled = True
    ImpRun = True
    BarraEstado.Top = 4560
    CreepForm.Height = 5685
    WaitImp.pStart
    WaitImp.Visible = True
    LabelImp.Visible = True
    LabelImp.Caption = "Espere mientras carga los datos"
    allDat = ""
    ' envíando comando de impotacion de datos
    MSComm1.Output = "*E&"

    'DíalogImp.Show vbModal
End Sub


Private Sub TestON_OnClick()    'activado la prueba en datacreep
    Dim res As Variant
    Dim comd As String
    Timer1.Enabled = False
    res = MsgBox("¿Seguro de realizar esta acción?", 4 + 64, "Confirmacion")
```

```
    If res = vbYes Then
        CreepON = Not TestON.Active
        TestON.Active = CreepON
        If CreepON = True Then
            comd = "*TR&"
        Else
            comd = "*TS&"
        End If
        MSComm1.Output = comd
    End If
    Timer1.Enabled = True
End Sub

Private Sub Timer1_Timer()  'Timer para actualizacion de datos online
    On Error Resume Next
    Dim OpenFlag

    mnuOpen.Checked = OpenFlag
    mnuSendText.Enabled = OpenFlag
    'tbrToolBar.Buttons("TransmitTextFile").Enabled = OpenFlag


    If MSComm1.PortOpen = True And ImpRun = False Then

        If sendCfg = True Then
            Comandos = "*C," + Str(n_Muestras) + "," + Str(n_Muestreo) + "," +
Str(tempSet) + "," + Str(CtempON) + ",&"
            MSComm1.Output = Comandos
            sendCfg = False
        Else
            MSComm1.Output = "*A&"
        End If

    End If

    If MSComm1.PortOpen Then
        commON.Active = True
        conect.Checked = True
        DatOnline.Enabled = True
        BarraEstado.Caption(1) = "Dispositivo Conectado..."
        ActHF.Enabled = True
        ImportDat.Enabled = True
    Else
        commON.Active = False
```

```
        conect.Checked = False
        DatOnline.Enabled = False
        ActHF.Enabled = False
        ImportDat.Enabled = False
        BarraEstado.Caption(1) = "Dispositivo Desconectado..."
    End If

End Sub

' El evento OnComm se usa para interceptar eventos y errores de
comunicaciones.
Private Static Sub MSComm1_OnComm()
    Dim Rcv02 As Integer, EOL As Integer

    ' Bifurca según la propiedad CommEvent.
    Select Case MSComm1.CommEvent
        ' Mensajes de evento.
        Case comEvReceive
            Call DataInComm
        Case comEvSend
        Case comEvCTS
            'EVMsg$ = "Detectado cambio en CTS"
        Case comEvDSR
            'EVMsg$ = "Detectado cambio en DSR"
        Case comEvCD
            'EVMsg$ = "Detectado cambio en CD"
        Case comEvRing
            'EVMsg$ = "El teléfono está sonando"
        Case comEvEOF
            'EVMsg$ = "Detectado el final del archivo"
            'Call DataInComm

        ' Mensajes de error.
        Case comBreak
            'ERMsg$ = "Parada recibida"
        Case comCDTO
            'ERMsg$ = "Sobrepasado el tiempo de espera de detección de
portadora"
        Case comCTSTO
            'ERMsg$ = "Soprepasado el tiempo de espera de CTS"
        Case comDCB
            'ERMsg$ = "Error recibiendo DCB"
        Case comDSRTO
            'ERMsg$ = "Sobrepasado el tiempo de espera de DSR"
```

```
        Case comFrame
            'ERMsg$ = "Error de marco"
        Case comOverrun
            'ERMsg$ = "Error de sobrecarga"
        Case comRxOver
            'ERMsg$ = "Desbordamiento en el búfer de recepción"
        Case comRxParity
            'ERMsg$ = "Error de paridad"
        Case comTxFull
            'ERMsg$ = "Búfer de transmisión lleno"
        Case Else
            'ERMsg$ = "Error o evento desconocido"
    End Select
    End If
End Sub

Public Sub OnlineDato(dato As String)   'Interpretacion de datos online
    Dim datos

    datos = Split(dato, ",", -1, vbTextCompare)
    Temperatura.Value = Val(datos(0))
    iThermometerX1.Position = Val(datos(0))
    Longitud.Value = Val(datos(1))
    iLinearGaugeX1.Position = Val(datos(1))
    If Val(datos(3)) = 1 Then
        iLedRoundX2.Active = True
        CreepON = True
        TestON.Active = True
    Else
        iLedRoundX2.Active = False
        CreepON = False
        TestON.Active = False
    End If
    If Val(datos(5)) = 1 Then
        iLedRoundX3.Active = True
    Else
        iLedRoundX3.Active = False
    End If
    If Val(datos(2)) = 1 Then
        iLedRoundX4.Active = True
    Else
        iLedRoundX4.Active = False
    End If
    If Val(datos(4)) = 1 Then
```

```
        iLedRoundX5.Active = True
    Else
        iLedRoundX5.Active = False
    End If

End Sub

Private Sub ImportDato()    'procesaminto de datos importados
    Dim datos
    Dim dato As String
    Dim v As Long
    Dim Heading(7) As String

    LabelImp.Caption = "Espere mientras convierte a Excel los datos"
    ProgressImp.Visible = True
    Heading(1) = "Index"
    Heading(2) = "Tiempo"
    Heading(3) = "Longitud"
    Heading(4) = "Temperatura"
    Heading(5) = "Fecha Inicio"
    Heading(6) = "Fecha Caduca"
    v = 2
    Call Inicio_Excel
    Call Formato_Excel(4, Heading)

    '**************************************
    Dim pdat As String
    Dim maxI As Long
    Dim sChar As Long
    Dim eChar As Long
    Dim last_eChar As Long
    Dim last_sChar As Long
    Dim i As Long
    Dim lin As Long
    Dim ch As String

    pdat = allDat
    sChar = 0
    lin = 0
    eChar = 0
    last_eChar = 0
    last_sChar = 0
    maxI = Len(allDat) + 1
    Index = 0
```

```
ProgressImp.Value = 0
For i = 1 To maxI Step 1
   ch = Mid$(allDat, i, 1)
   If ch = Chr(2) Then
      sChar = i
   End If
   If ch = Chr(26) Then
      eChar = i
   End If
   If eChar > last_eChar And sChar > last_sChar Then
      'List1.AddItem Mid(pdat, sChar, eChar - sChar)
      dato = Mid(pdat, sChar + 1, eChar - sChar - 1)
      'lin = lin + 1
      'Label2.Caption = lin
      last_eChar = eChar
      last_scahr = sChar

      datos = Split(dato, ",", -1, vbTextCompare)

      Select Case datos(0)
      Case "I"
         maxIndex = Val(datos(1))
      Case "R"
         objExcel.ActiveSheet.Cells(v, 1) = Val(datos(1))
         objExcel.ActiveSheet.Cells(v, 2) = Val(datos(2))
         objExcel.ActiveSheet.Cells(v, 3) = Val(datos(3))
         objExcel.ActiveSheet.Cells(v, 4) = Val(datos(4))
         Index = Val(datos(1))
         ProgressImp.Value = 100 * (Index / maxIndex)
         v = v + 1
       Case "Q"
         MsgBox "Completado con exito!", 64
         objExcel.Visible = True
         'objExcel.ActiveSheet.PrintPreview
         Set objExcel = Nothing
         ImpRun = False
         Timer1.Enabled = True
         LabelImp.Visible = False
         ProgressImp.Visible = False
         BarraEstado.Top = 3960
         CreepForm.Height = 5115
      End Select
   End If
Next i
```

```
End Sub

Public Sub DataInComm()     'recepcion de datos de datacreep
   Dim Buffer As String
   Dim leng As Long

   If ImpRun = True Then
      allDat = allDat + MSComm1.Input
      Time_ms = 0
   Else
      Buffer = Buffer + MSComm1.Input
      leng = Len(Buffer)
      If leng > 30 Then
         leng = 0
      End If
      Rcv02 = InStr(Buffer, Chr(2))
      EOL = InStr(Buffer, Chr(26))
      If Rcv02 And EOL > Rcv02 Then
         Buffer = Mid(Buffer, Rcv02 + 1, EOL - 2)
         Call OnlineDato(Buffer)
      End If
      Buffer = Mid(Buffer, EOL + 1)
   End If

End Sub

Private Sub Timer2_Timer()  'mide tiempo muerto en comunicacion
   Time_ms = Time_ms + 1
   If Time_ms = 5 Then
      Timer2.Enabled = False
      WaitImp.pStop
      WaitImp.Visible = False
      Call ImportDato
   End If
End Sub

Private Sub GetSettings()   'captura datos del registro de windows
   Dim strtemp As String
   ' Get Settings from Registry
   n_Muestras = Val(GetSetting(APPNAME, "Settings", "Muestras", ""))
   n_Muestreo = Val(GetSetting(APPNAME, "Settings", "Muestreo", ""))
   n_Horas = Val(GetSetting(APPNAME, "Settings", "Horas", ""))
   tempSet = Val(GetSetting(APPNAME, "Settings", "TempSet", ""))
   strtemp = GetSetting(APPNAME, "Settings", "CtrlTemp", "")
```

```
    If StrComp(strtemp, " Falso", vbTextCompare) = 0 Then
        CtempON = False
    Else
        CtempON = True
    End If

End Sub

Private Sub SaveSettings()  'guarda datos en el registro de windows
    ' Save Settings to Registry
    SaveSetting APPNAME, "Settings", "Muestras", Str(n_Muestras)
    SaveSetting APPNAME, "Settings", "Muestreo", Str(n_Muestreo)
    SaveSetting APPNAME, "Settings", "Horas", Str(n_Horas)
    SaveSetting APPNAME, "Settings", "TempSet", Str(tempSet)
    SaveSetting APPNAME, "Settings", "CtrlTemp", Str(CtempON)
End Sub
```

Código fuente de la pantalla de configuración

```
Dim last_tempSet As Long
Dim last_muestras As Single
Dim last_muestreo As Single
Dim last_horas As Single
Dim last_CtempON As Boolean

Private Sub cancel_cfgtest_Click()
   tempSet = last_tempSet
   n_Muestras = last_muestras
   n_Muestreo = last_muestreo
   n_Horas = last_horas
   CtempON = last_CtempON
   Unload Me
End Sub

Private Sub Form_Load()
   i_muestras.Value = n_Muestras
   i_muestreo.Value = n_Muestreo
   i_horas.Value = n_Horas
   iKnobX1.Position = tempSet
   iSwitchLedX1.Active = CtempON

   last_tempSet = tempSet
   last_muestras = n_Muestras
   last_muestreo = n_Muestreo
   last_horas = n_Horas
   last_CtempON = CtempON


End Sub

Private Sub i_horas_OnChangeUser()
   n_Muestras = i_muestras.Value
   n_Horas = i_horas.Value
   n_Muestreo = Round(n_Horas * 3600 / n_Muestras, 1)
   i_muestreo.Value = n_Muestreo
End Sub

Private Sub i_muestras_OnChangeUser()

   n_Muestreo = i_muestreo.Value
   n_Muestras = i_muestras.Value
```

```
    n_Horas = n_Muestras * n_Muestreo / 3600
    i_horas.Value = n_Horas

End Sub

Private Sub i_muestreo_OnChangeUser()

    n_Muestreo = i_muestreo.Value
    n_Muestras = i_muestras.Value
    n_Horas = n_Muestras * n_Muestreo / 3600
    i_horas.Value = n_Horas

End Sub

Private Sub iKnobX1_OnPositionChange()
    tempSet = iKnobX1.Position
End Sub

Private Sub iSwitchLedX1_OnChange()
    CtempON = iSwitchLedX1.Active
End Sub

Private Sub ok_cfgtest_Click()
    Dim Comandos As String
    sendCfg = True     'Activacion de bandera
    Unload Me

End Sub
```

D. Manual de usuario.

Instalación de driver USB

Para el funcionamiento del módulo Data creep se es necesario instalar el driver del USB como a continuación se detalla

1. Verificar que el módulo DATACREEP este correctamente energizado.
2. Conectar el módulo al puerto USB del PC.
3. Luego de aparecer la ventana de nuevo hardware encontrado, escoger la opción como se muestra en la figura B.1 y presionar siguiente.

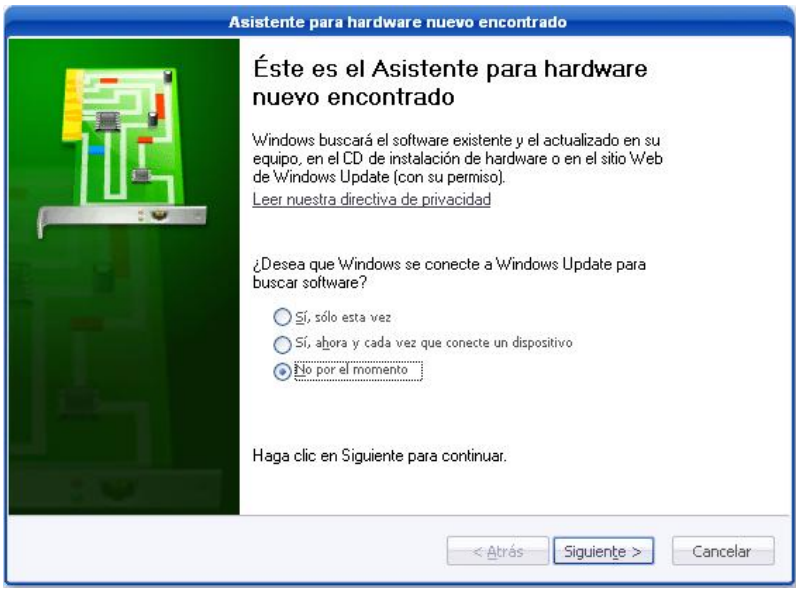


Fig. B.1

4. Se escoge la opción avanzada (Fig. B.2)

5. Buscar la ruta de acceso del driver (Fig. B.3)

6. Escoger la opción continuar como lo muestra la Fig. B.4

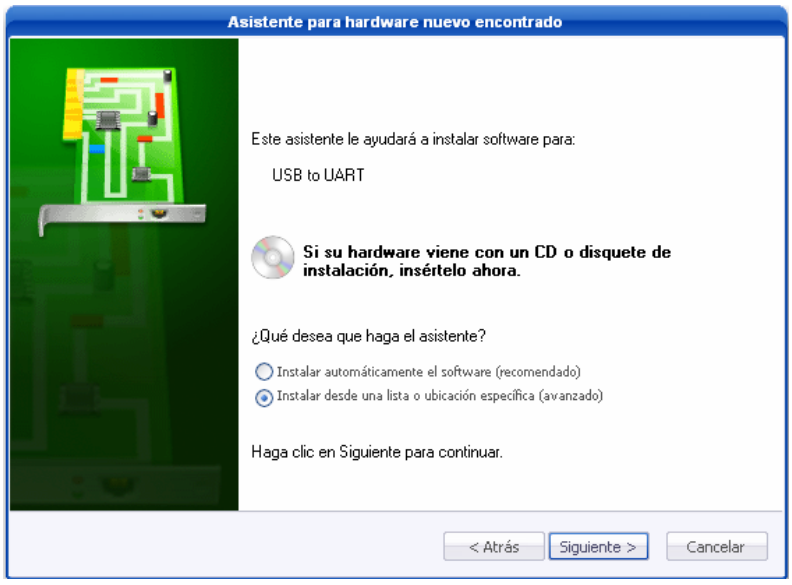


Fig. B.2

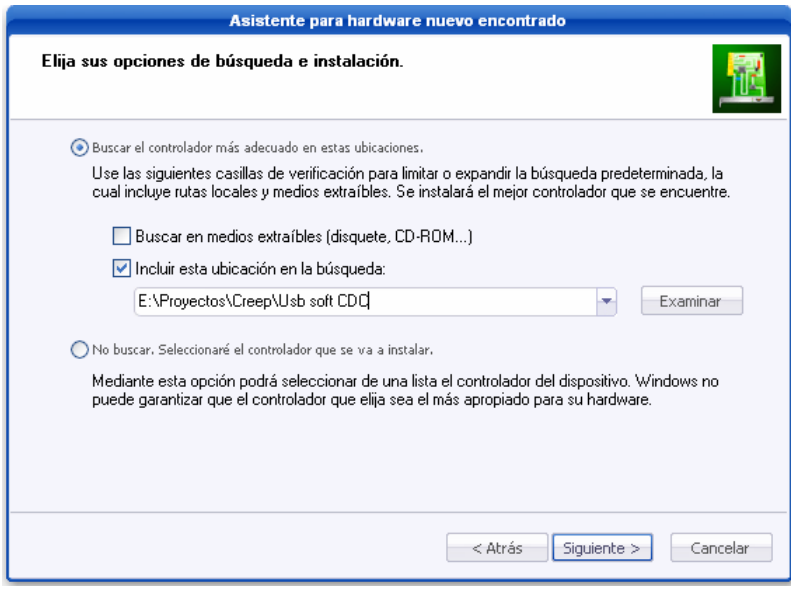


Fig. B.3

7. Presione finalizar (Fig. B.5)

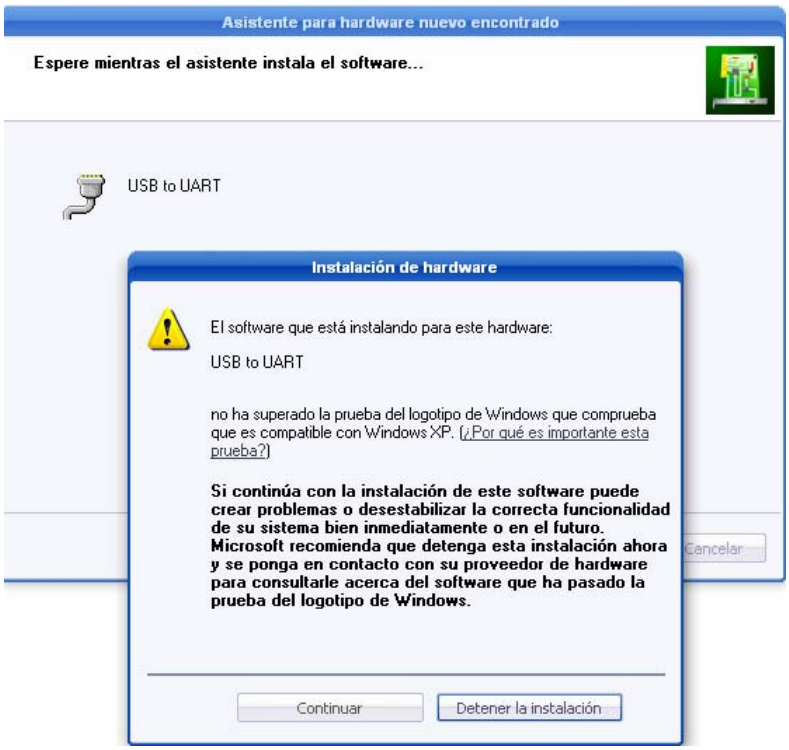

Asistente para hardware nuevo encontrado
Espere mientras el asistente instala el software...
USB to UART
Instalación de hardware
El software que está instalando para este hardware:
USB to UART
no ha superado la prueba del logotipo de Windows que comprueba que es compatible con Windows XP. (¿Por qué es importante esta prueba?)
Si continúa con la instalación de este software puede crear problemas o desestabilizar la correcta funcionalidad de su sistema bien inmediatamente o en el futuro. Microsoft recomienda que detenga esta instalación ahora y se ponga en contacto con su proveedor de hardware para consultarle acerca del software que ha pasado la prueba del logotipo de Windows.
Continuar
Detener la instalación
Cancelar

Fig. B.4

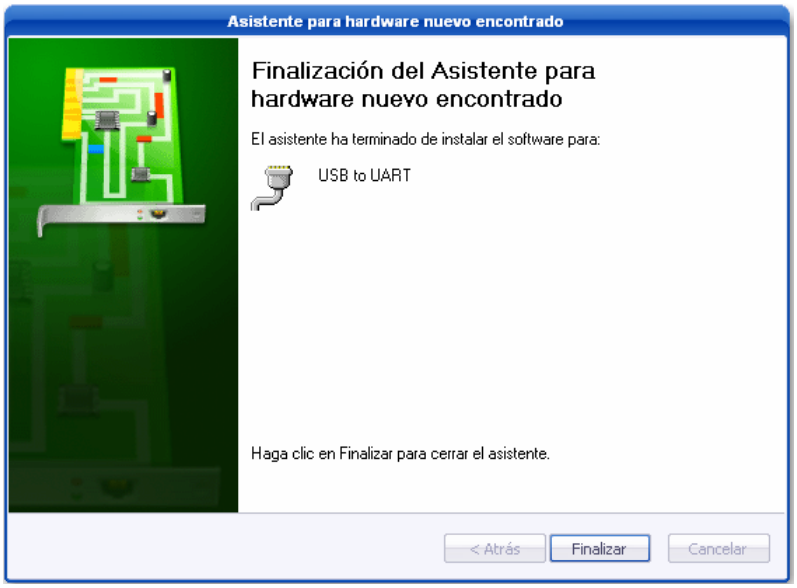

Asistente para hardware nuevo encontrado
Finalización del Asistente para hardware nuevo encontrado
El asistente ha terminado de instalar el software para:
USB to UART
Haga clic en Finalizar para cerrar el asistente.
< Atrás
Finalizar
Cancelar

Fig. B.5

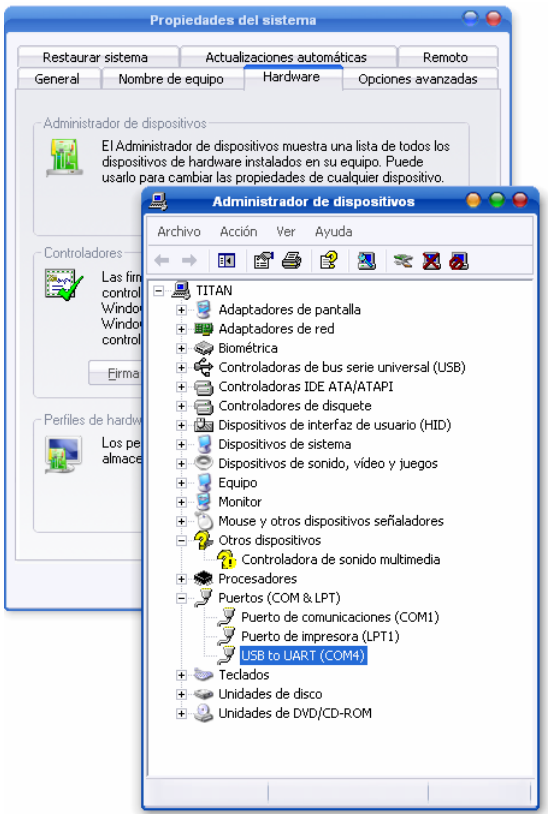


Fig. B.6

8. Por último verifique el COM seleccionado en el administrador de dispositivos (Fig. B.6)

Guía de uso del CreepSoft

1. Ejecutar la aplicación CreepSoft
2. Escoger el COM asignado por Windows en la pestaña (Fig. B.7).
3. El siguiente paso es ir al menú desplegable (Fig. B.8) Ejecutar y hacer clic en Conectar

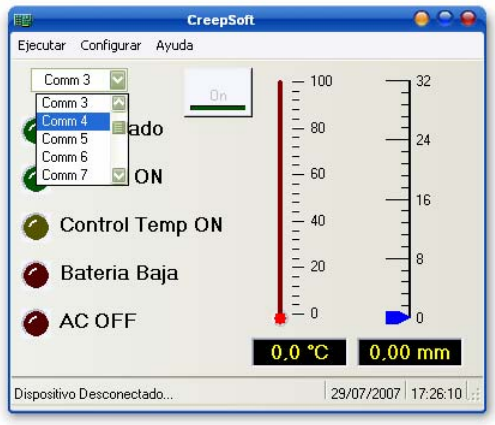


Fig. B.7

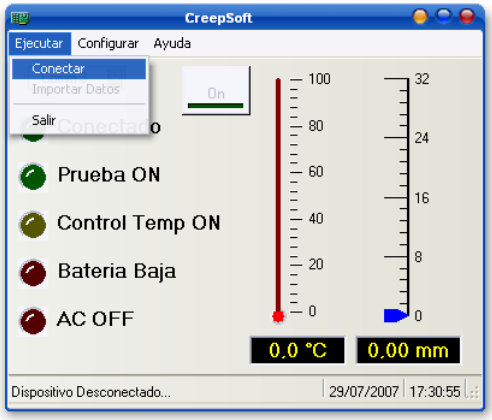


Fig. B.8

4. Luego de esto la luz indicadora de conectado se enciende (Fig.B.9), además de los indicadores de energía eléctrica
5. En la fig. B.9 se observa que en el menú desplegable Configurar tenemos dos opciones. Al escoger Hora y Fecha, el módulo actualiza el RTC. Por otro lado, la configuración del ensayo se la realiza a través de Parámetros de prueba.

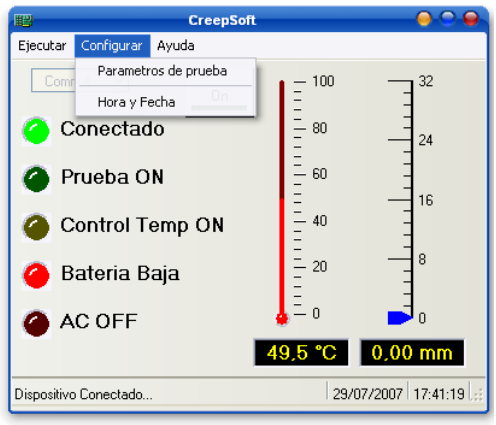


Fig.B.9

Fig.B.10

6. En la pestaña Toma de Muestras (Fig. 10) el usuario debe escoger el número y período de muestras. El tiempo de la prueba es directamente calculada por el programa.

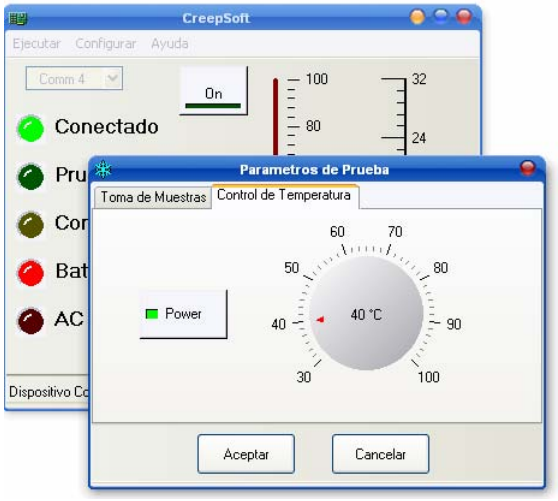
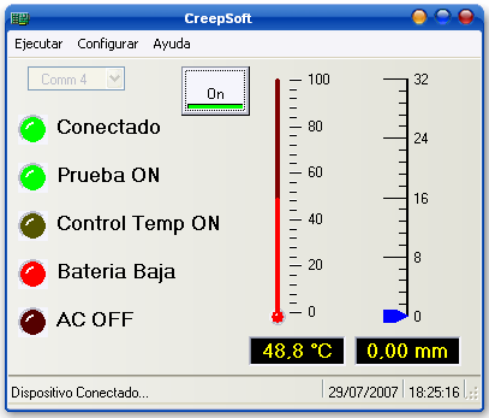


Fig.B.11

7. En la figura B.11 se habilita el control de temperatura mediante el boton Power, y se fija la variable de proceso (Temperatura) con la perilla.

Nota: Una vez aceptado los cambios, automáticamente la aplicación descarga los datos al módulo.

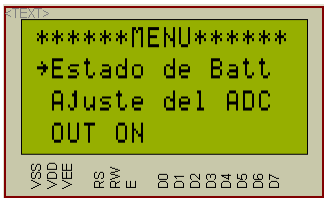
8. Para finalizar se debe poner On el ensayo, esto se logra pulsando el botón ON ubicado en la parte superior de la ventana principal.



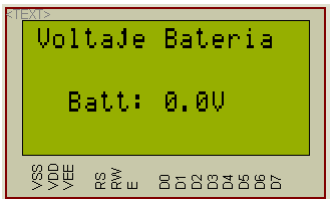
Guía de uso del módulo del DATACREEP

1. La pantalla principal muestra la fecha en formato dd/mm/yy, la hora y el estado de prueba.

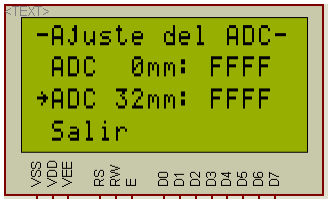
2. Pulsar enter para ingresar al menú principal, con los botones up y down podremos movernos para elegir una de las tres opciones presentes en esta pantalla.



3. Al elegir Estado de Batt obtenemos el valor de voltaje presente en la batería de respaldo del sistema.

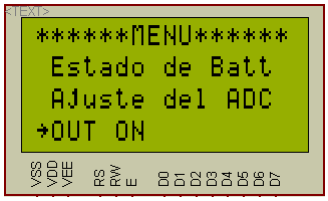


4. Ajuste del ADC es donde se calibra el rango del sensor lineal.



5. La calibración se la realiza de la siguiente manera:
    a. Mover la parte móvil del sensor lineal en posición mínima
    b. Con las teclas up down de la pantalla ubicarse en ADC 0mm y presionar enter.
    c. Luego de esto la parte móvil se tendrá que ubicar en posición máxima.

    d. Nuevamente con las teclas up down ubicarse en ADC 32mm. y pulsar enter.
    e. Finalmente ubicarse en salir y pulsar enter.
    f. Cada vez que se realiza un cambio, un carácter (*) aparecerá junto al valor corregido.
6. La última opción de la pantalla MENU es cuando se requiera encender el calentador por prueba, pero no se tendrá ningun tipo de control de este.



E. Hojas de datos de los componentes electrónicos.

a. Regulador de conmutación integrado

b. Conversor Analógico Digital

c. Transmisor de corriente de 4-20mA

d. Microcontrolador PIC18F4550

a) Regulador de conmutación integrado (PT5100)

El PT5100 es un regulador de voltaje de de alta eficiencia con amplio rango de voltaje de entrada.

Descripción	Min.	Tip.	Max.	Unid.
Corriente de salida	0.1	-----	1.0	A
Rango de voltaje de entrada	9	-----	26	VDC
Frecuencia de conmutación	500	650	800	KHZ
Eficiencia	-----	90	-----	%
Variación del voltaje de salida total	-----	+-1.5	+-3	%V
Capacitancia de salida externa	100	-----	-----	uF
Resistencia térmica	-----	50	-----	ºC/W

b) Conversor Analógico Digital (ADS1110)

Es conversor de analógico-digital de 16 Bits de resolución y entrada diferencial y además es compatible con la interfase I2C.

Descripción	Min.	Tip.	Max.	Unid.
Voltaje de entrada a escala completa	-----	+-2.048	-----	V
Voltaje de alimentación	2.7	-----	5.5	V
Disipación de energía	-----	1.2	1.75	mW
Resolución	12	-----	16	Bits
Impedancia de entrada en modo común	-----	3.5	-----	MΩ
Impedancia de entrada diferencial	-----	3.5	-----	MΩ
Tipo de interfase de comunicación	-----	I2C	-----	

c) Transmisor de corriente de 4-20mA (XTR105)

El XTR105 es un transmisor de dos cables de 4-20mA y está provisto de toda la circuitería necesaria para excitar a un sensor de temperatura tipo RTD por lo que lo hace ideal para aplicaciones en la industria.

Descripción	Min.	Tip.	Max.	Unid.
Corriente de salida	4	-----	20	mA
Limite de sobre-escala	24	27	30	mA
Limite de baja-escala	1.8	2.2	2.6	mA
Voltaje de offset	-----	+-50	+-100	uV
Rango de voltaje de alimentación	7.5	24	36	V
Tipo de sensor de temperatura	-----	PT100	-----	
Rango de temperatura	-40	-----	85	ºC

d) Microcontrolador PIC18F4550

Característica del bus serial universal (USB)

Cumple con la especificación USB V2.0.

Soporta transferencias de control, interrupción, Bulk e isocronica.

Soporta hasta 32 registros de entrada (Endpoint).

Poseé una memoria RAM de 1Kbyte para el uso del USB.

Incluye USB tranceiver en el chip.

Característica de los periféricos del chip.

Tres interrupciones externas

Cuatro módulos de temporización

Dos módulos de captura, comparación y modulación de ancho de pulso

(CPP).

Modulo USART avanzado.

Modulo USB v2.0

Características especiales del controlador.

Arquitectura de CPU optimizada para compilador de C, con juego de instrucciones extendidas.

Memoria Flash con 100.000 ciclos de escritura-lectura.

Memoria Flash con más de 40 años de retención de datos.

Autoprogramable bajo software de control (bootloader).

Interrupciones con 8 niveles de prioridades.

Multiplicación de números enteros con longitud de 1 byte en un solo ciclo.

Protección de código programado.

Perro guardian.

Rango de voltaje de operación de 2-5.5VDC.