DISEÑO DE UN SISTEMA DE ADQUISICIÓN DE DATOS POR BUS SERIAL UNIVERSAL (USB) Y MEMORIA SRAM EXTERNA

FREDY ALEXANDER ASCENCIO CAMACHO

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE CIENCIAS FISICOMECÁNICAS

ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

BUCARAMANGA

2005

DISEÑO DE UN SISTEMA DE ADQUISICIÓN DE DATOS POR BUS SERIAL UNIVERSAL (USB) Y MEMORIA SRAM EXTERNA

TESIS DE GRADO

AUTOR:

FREDY ALEXANDER ASCENCIO CAMACHO

DIRECTOR:
MSC. JAIME BARRERO PÉREZ

CODIRECTOR:
ING. JORGE E. HIGUERA PORTILLA

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE CIENCIAS FISICOMECÁNICAS
ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES
BUCARAMANGA

2005

CONTENIDO

pág.

LISTA DE FIGURAS

Pág.

LISTA DE TABLAS

Pág.

TITULO: DISEÑO DE UN SISTEMA DE ADQUISICIÓN DE DATOS POR BUS SERIAL UNIVERSAL (USB) Y MEMORIA SRAM EXTERNA[1]

AUTORES: FREDY ALEXANDER ASCENCIO CAMACHO [2]

PALABRAS CLAVE:

Bus Serial Universal, USB
Tarjeta de adquisición de datos
Instrumento virtual
Conectar y listo
LabView 7.0.

DESCRIPCIÓN:

En la actualidad, las interfaces de comunicación con el PC son cada vez más rápidas, flexibles y permiten un manejo fácil por parte del usuario. Un ejemplo de esto es el Bus Serial Universal (USB), una interfaz que permite la configuración *plug and play* y de la cual se han construido una gran variedad de periféricos para el PC. Este trabajo presenta una tarjeta de adquisición de datos, que utiliza el bus USB 2.0, cuyo diseño cuenta con cuatro entradas analógicas de tensión entre ±10 V no diferenciales, ancho de banda de 200KHz y resolución de 16 bits. Además, desarrolla la comunicación serial SPI para la comunicación entre el microcontrolador y el conversor A/D. La

---

[1] * Trabajo de grado

[2] Facultad de Ingenierías Físico mecánicas. Escuela de Ingeniería Eléctrica, Electrónica y Telecomunicaciones. Director Jaime Barrero Pérez. Codirector: Jorge E Higuera P

temporización del conversor A/D es realizada por medio de un oscilador interno del conversor A/D AD974. Los datos digitalizados son almacenados en una memoria no volátil SRAM BQ4017. El microcontrolador MC68HC908GP32 se encarga de dar o de recibir el dato que va a ser escrito o leído de la memoria SRAM, así mismo, se encarga de generar la lógica de control de escritura o de lectura para acceder correctamente a la memoria. La transmisión de datos hacia el PC se realiza a través de un controlador USB externo FT232BM. Esta etapa utiliza como canal para la comunicación el bus USB 2.0, con las ventajas que este bus ofrece, tales como facilidad de conexión, *plug and play,* y alta velocidad. La tarjeta ADQ USB se conecta a un computador por el bus USB y envía las muestras de los cuatro canales a una tasa efectiva de 124Kbytes por segundo. Para configurar la frecuencia de bus del microcontrolador, se trabajó con un cristal oscilador de baja frecuencia (32Kz) y el modulo PLL del microcontrolador habilitado, lo que permite que la frecuencia del oscilador sea multiplicada por un valor deseado para obtener el reloj de referencia del bus del microcontrolador. Esto permite que con un cristal externo de frecuencia de 32.768KHz se configuren frecuencias de bus de hasta 8.2 MHz, que es la máxima frecuencia recomendada por el fabricante; la frecuencia de trabajo que se configuró finalmente para la tarjeta es de 7.9872MHz con esta versión del bus. Las señales digitalizadas se visualizan en el dominio del tiempo y la frecuencia mediante una interfaz desarrollada en LabView 7.0, de esta forma, se tiene un sistema que permite “conectar y listo”, es decir la inserción en caliente, reduciendo la intervención del usuario para su configuración, además se explora y aprovecha al máximo las prestaciones del bus USB 2.0, lo cual es un avance del grupo de investigación CEMOS en la creación de dispositivos USB y permitirá llevar a cabo futuros proyectos que utilicen la descarga de datos a alta velocidad para desarrollar aplicaciones mas flexibles que interactúen con el PC.

TITLE: DESIGN AND IMPLEMENTATION OF DATA ACQUISITION SYSTEM USING USB BUS AND MEMORY SRAM[3]

AUTHORS: FREDY ALEXANDER ASCENCIO CAMACHO[4]

KEYWORDS:

Bus Serial Universal, USB

Data acquisition system

Virtual instrument

Plug & play

LabView 7.0.

DESCRIPTION:

Actually, the communications interfaces with a PC are more quickly and flexibility and allow easy handling on the part of user. An example is the Universal Serial Bus (USB), an asynchronous interface that has allowed to design and to build a great variety devices to use this communication port in the last years. This work presents the data acquisition that uses the USB version 2.0 full speed whose design has four analog voltage inputs, non differential, ±10V input range, 200KHz bandwidth and 16-bit sampling. Also, development the serial communication SPI with the microcontroller and data converter A/D. The timing of convertion is a development for internal oscilator built in. Digital data are save in a memory SRAM non volatile BQ4017. The microcontroller MC68HC908GP32 receive and give the data for read and write in the SRAM memory, also request the logic of control for write or read

---

[3] Degree work.

[4] Physics-mecanical Faculty. Electrical, Electronic and Telecommunications Engineering School. Jaime Barrero Pérez, director. Jorge Eduardo Higuera P codirector

in a memory SRAM. The data transmission at PC is development for a external USB controller FT232BM. This stage use a chanel USB 2.0 with a advantage such easy connect , *plug and play* and high velocity. This USB card are connect at PC with USB bus and send the samples of for chanel with a effective rate of 124Kbps. The Frecuency configuration of bus microcontroller is development with a cristal oscillator of 32KHz and the PLL module enable allow multiply clock rate for microcontroller bus. This multiply clock rate with external cristal of 32Khz allow configurate the maximun bus frecuency of 8.2Mhz. The work frecuency is configurate finally in this card in 7.987Mhz with this USB version. The digital signal are visualize in time domain and frecuency domain through a Labview interface, this way I have a system plug and play decrease the user intervention in the configuration, also are explorating the maximun characteristics of USB bus 2.0 this development is an advance of the group research CEMOS in the creation of devices USB and allow the maximum characteristics of USB bus 2.0 and will allow at new investigation in this area for data adquisition system more fast and flexibility and let more interaction with the PC

INTRODUCCIÓN

En la actualidad, las interfaces de comunicación con el PC son cada vez más rápidas, flexibles y permiten un manejo fácil por parte del usuario. Un ejemplo de esto es el Bus Serial Universal (USB), una interfaz que permite la configuración plug and play y de la cual se han construido una gran variedad de periféricos para el PC. El bus serial universal (USB) comenzó a desarrollarse en 1994 a partir del Forum USB compuesto por empresas como Compaq, Intel, Microsoft, NEC entre otras, partiendo de tres elementos fundamentales:

La conexión del PC a servicios de valor agregado, permitiendo la expansión de los buses de conexión a redes LAN y dispositivos compatibles con el PC, que tradicionalmente trabajaban con puertos RS232 y paralelos.

La facilidad de uso, respecto a las demás interfaces que implican el uso de hardware adicional, configuraciones menos flexibles y baja velocidad de transferencia de datos.

La expansión de puertos, hasta entonces limitada a la inserción de tarjetas de circuitos en el PC y, en consecuencia, una muy limitada flexibilidad de elementos, dispositivos y programas compatibles.

La primera especificación comercial de USB (conocida como versión USB 1.1) fue liberada el 23 de septiembre de 1998. Un año después, USB era una interfaz común en la mayoría de los equipos de cómputo personal. El objetivo se cumplió y permitió que dispositivos de diversos fabricantes pudieran comunicarse entre sí en una arquitectura compatible y flexible. En

abril del año 2000 se presenta el bus USB de alta velocidad o USB 2.0 que mejora el desempeño de las versiones anteriores introduciendo transferencias de datos para aplicaciones de video y audio multimedia de alta velocidad.

En el capitulo uno se presenta una breve descripción de los sistemas de adquisición de datos. En el capitulo dos se tratan aspectos fundamentales del bus USB. En el capitulo tres se enuncian los dispositivos de hardware que componen el diseño de la tarjeta de adquisición de datos USB. En el capitulo cuatro se presenta los bloques funcionales del sistemas de adquisición de datos detallando el diseño del hardware, la comunicación con el PC y su comunicación por bus USB. En el capitulo cinco se tratan los temas concernientes al diseño de drivers y la implementación del firmware del microcontrolador que controla el sistema de adquisición de datos USB. En el capitulo seis se describe la aplicación desarrollada en Labview para el manejo de la tarjeta de adquisición de datos USB así como su configuración mediante el bus USB.

1. TARJETAS DE ADQUISICIÓN DE DATOS

Las tarjetas de adquisición de datos son los dispositivos de hardware que permiten leer señales de naturaleza analógica o digital para ser visualizadas en el PC. A través de ellas los datos analógicos obtenidos de sus canales conectados a transductores, son convertidos a datos digitales y visualizados en el software de la aplicación.

Las tarjetas de adquisición de datos se conectan directamente a los buses del PC y permiten adquirir y procesar datos por lo general en tiempo real. Cada tarjeta presenta funcionalidades diferentes, lo que permite aplicaciones muy variadas, como podría ser el conteo de eventos, la generación de señales digitales de salida, o la adquisición de señales analógicas de entrada.

Normalmente una tarjeta de adquisición de datos aporta los bloques de encaminamiento de la señal, el acondicionamiento de la señal, un bloque de memoria y la visualización se realiza a través del software de aplicación.

Una ventaja importante en las tarjetas de adquisición de datos es la versatilidad, la resolución de la señal adquirida, y la transmisión al PC de los datos digitales para realizar el procesamiento y la visualización de los datos. También es importante la facilidad de instalación, de puesta en marcha y su flexibilidad de uso en variadas aplicaciones industriales, de laboratorio y automotrices.

1.1. TIPOS DE TARJETAS EN FUNCIÓN DE LA FORMA DE CONEXIÓN.

Las tarjetas de adquisición de datos se pueden clasificar según la forma de conexión de la tarjeta al PC.

1.1.1. TARJETA DE ADQUISICIÓN DE DATOS POR BUS ISA (INTERNA)

Su arquitectura de comunicación de datos interna de 16 bits permitía la conexión al slot ISA del PC. Era necesario abrir la carcaza de la PC para su instalación. Se debía contar con ranura ISA disponible. Este tipo de tarjeta era habitual en los PC antiguos pero ya no se incluye en la tarjeta madre de los computadores actuales.

1.1.2. TARJETA DE ADQUISICIÓN DE DATOS POR BUS PCI (INTERNA).

Es una tarjeta de adquisición de datos que se inserta en un slot PCI (32 bits) de la placa madre del PC. Por lo tanto también es interna. La ventaja respecto a la anterior es que la velocidad de transmisión de datos entre la tarjeta y el procesador del PC es mayor por el bus PCI. Es también muy habitual en computadores de sobremesa.

1.1.3. TARJETA DE ADQUISICIÓN DE DATOS PCMCIA.

Es una tarjeta interna que se puede conectar a la mayoría de los computadores portátiles actuales. Posee un tamaño reducido y tasas de descarga de datos mayores al puerto serial y paralelo (de hecho es similar al bus ISA o PCI). Algunos PC industriales también disponen de conector PCMCIA. También se consiguen adaptadores PCMCIA para conexión con el PC.

1.1.4. TARJETA DE ADQUISICIÓN DE DATOS POR PUERTO PARALELO.

En este caso la tarjeta es externa al PC. Requiere de alimentación externa y se conecta al puerto paralelo. La velocidad de transmisión de datos entre la tarjeta de adquisición de datos por puerto paralelo es mucho menor a la que se tiene con el bus ISA , PCI, PCMCIA y BUS USB 2.0. Sirve tanto para computadores de sobremesa como para portátiles. Este tipo de tarjeta de adquisición de datos esta en decadencia debido al posicionamiento de otros buses de comunicación.

1.1.5. TARJETA DE ADQUISICIÓN DE DATOS POR BUS USB

Es un periférico externo usado en PCS, *PDAS* y *notebooks* actuales. Se debe contar con puerto USB 2.0. Esta opción no es compatible bajo los sistemas operativos Windows 95 o NT y requiere de un driver para su configuración inicial. Debido a su flexibilidad, bajo costo y velocidad tiene gran aceptación en aplicaciones de instrumentación, comunicación y control. Puede alimentarse directamente del bus USB, sin necesidad de fuente externa.

1.1.6. TARJETA DE ADQUISICIÓN DE DATOS POR PUERTO SERIE RS-232

Es una tarjeta de adquisición de datos externa usada en PCS, *notebooks*, PLCs y otros dispositivos que soportan la norma RS-232. Debido a que ha entrado en desuso en los PCS actuales no es recomendable para nuevos diseños que requieran conexión al PC. La velocidad de descarga de datos es limitada.

1.2. TIPOS DE TARJETAS DE ADQUISICIÓN DE DATOS EN FUNCIÓN DE LAS ENTRADAS Y SALIDAS.

Hay infinidad de modelos de tarjetas de adquisición de datos en el mercado, cada una con unas características diferentes. Las entradas y salidas de todas esas tarjetas pueden ser analógicas o digitales. Normalmente una tarjeta de adquisición de datos combina entradas y salidas de los dos tipos. Son habituales, por ejemplo, las tarjetas con 16 entradas digitales, 8 o16 salidas digitales. En función de la aplicación se eligen sus características. También hay tarjetas de adquisición de datos con entradas especiales (como entradas para encoder incremental) o salidas especiales (como salida PWM).

1.2.1. TARJETA DE ADQUISICIÓN DE DATOS CON ENTRADAS ANALÓGICAS SIMPLES ("SINGLE ENDED").

Es una tarjeta de adquisición de datos que posee entradas analógicas de tensión referenciadas a tierra. Son habituales las tarjetas con 8 ó 16 entradas analógicas "single ended". Todas las entradas tienen un punto común (la masa analógica), de forma que las señales de todos los sensores deben tener la misma masa. Los rangos de entrada pueden variar, pero los más habituales son [0,5], [-5,5], [0,10], [-10,10]. La resolución puede variar entre 10 bits y 16 bits, aunque lo más habitual es 16 bits. Una característica importante de las entradas analógicas es el tiempo de conversión. Este es el tiempo que tarda desde que se da la orden de conversión hasta que se tiene el valor digital convertido. Este tiempo de conversión limita la frecuencia máxima a la que se pueden tomar muestras. Hay tarjetas de adquisición de datos con tiempos de conversión desde los nanosegundos hasta segundos, dependiendo del conversor A/D y el tipo de tarjeta.

1.2.2. TARJETA DE ADQUISICIÓN DE DATOS CON ENTRADAS ANALÓGICAS DIFERENCIALES.

Son tarjetas de adquisición de datos con entradas en tensión. Cada entrada analógica tiene dos bornes aislados, de forma que se mide la diferencia de tensión entre los dos. De esta forma cada sensor puede tener una alimentación diferente aislada de los otros sensores. Son habituales las tarjetas con 4 u 8 entradas diferenciales. Es habitual también que estas tarjetas permitan configurar las entradas como diferenciales o como "*single ended*", de forma que se tienen 8 entradas diferenciales ó 16 "*single ended*".

1.2.3. TARJETA DE ADQUISICIÓN DE DATOS CON ENTRADAS ANALÓGICAS DE CORRIENTE.

Este tipo de tarjetas de adquisición de datos admiten señales de corriente de 0 a 20mA ó de 4 a 20 mA. Una entrada de tensión puede utilizarse para leer una señal de corriente sin más que colocar una resistencia en paralelo. Si se coloca una resistencia de 250 Ω la corriente de 4 a 20mA se convierte en una tensión de 1 a 5 V.

1.2.4. TARJETA DE ADQUISICIÓN DE DATOS CON ENTRADAS ANALÓGICAS DE TEMPERATURA.

Estas entradas permiten conectar directamente un termopar o una termocupla. Contienen la circuitería necesaria de amplificación y acondicionamiento de la señal.

1.2.5. TARJETA DE ADQUISICIÓN DE DATOS CON ENTRADAS DIGITALES.

Son entradas digitales de un nivel de tensión de 5 voltios, compatibles TTL ó CMOS. Se pueden conectar a ellas directamente señales de circuitos digitales TTL ó CMOS con alimentación de 5 V. Son habituales las tarjetas

con 16 entradas de este tipo. Normalmente todas las entradas tienen un borne común (masa digital), por lo que todas las señales deben tener la misma tierra digital.

1.2.6. TARJETA DE ADQUISICIÓN DE DATOS CON ENTRADAS DIGITALES A 24V.

Son entradas digitales opto acopladas como las que tiene un autómata Programable (PLC). Admiten un nivel de tensión de 24V.

1.2.7. TARJETA DE ADQUISICIÓN DE DATOS CON ENTRADAS DE CONTADOR RÁPIDO.

Es una entrada digital especial que va conectada internamente a un *timer* o contador. Se utiliza para contar pulsos de manera sincrona(*timer*) o asíncrona (contador externo).

1.2.8. TARJETA DE ADQUISICIÓN DE DATOS CON ENTRADAS DE CAPTURA.

Es una entrada digital que va unida a una unidad de captura similar a la que se encuentra en los microcontroladores (un registro captura el valor de un temporizador cuando la salida pasa de cero a uno). Se utilizan para medir el periodo de una señal de entrada.

1.2.9. TARJETA DE ADQUISICIÓN DE DATOS CON ENTRADAS DE CODIFICADOR INCREMENTAL(ENCODER).

Son dos (o tres) entradas digitales especiales donde se conectan las dos (o tres) señales de un encoder incremental de dos fases. La tarjeta lleva incorporada la electrónica necesaria para contar los pulsos (incrementando o decrementando), de forma que se tiene la posición del encoder. Al tratarse

de un *encoder* incremental hay que inicializar el contador en un valor determinado cuando se está en una posición conocida.

1.2.10. TARJETA DE ADQUISICIÓN DE DATOS CON SALIDAS ANALÓGICAS.

Son salidas en tensión o en corriente. El número de estas salidas suele ser menor que el de entradas. Son habituales tarjetas con 2 ó 4 ó 8 salidas analógicas. Pueden ser aisladas o tener una masa común. Los rangos de tensiones habituales son [0,5V], [-5, V], [0,10V],[-10,10V], mientras los de corriente son de 0-20 mA ó de 4-20 mA. La resolución puede variar entre 10 bits y 16 bits, pero lo más habitual es 16 bits. Una característica es el tiempo de conversión. Es el tiempo que pasa desde que se modifica el valor digital hasta que la tensión de salida se ha estabilizado.

1.2.11. TARJETA DE ADQUISICIÓN DE DATOS CON SALIDAS DIGITALES.

Son salidas digitales de un nivel de tensión de 5 voltios, compatibles TTL ó CMOS. Se pueden conectar directamente a circuitos digitales TTL ó CMOS con alimentación de 5 V. Son habituales las tarjetas con 16 salidas de este tipo. Normalmente todas las salidas tienen un borne común (tierra digital), por lo que todas las señales deben tener la misma tierra. Estas salidas sirven para controlar procesos industriales asociados con una lógica de control.

1.2.12. TARJETA DE ADQUISICIÓN DE DATOS CON SALIDA DIGITAL PWM.

Es una salida digital especial que genera una onda cuadrada de ciclo de trabajo programable.

1.2.13. TARJETA DE ADQUISICIÓN DE DATOS CON SALIDAS DIGITALES A RELÉ.

Son salidas digitales con un contacto que se cierra NA o NC. Son iguales a las salidas digitales a relé de los autómatas programables. Permiten conmutar directamente cargas de alterna(CA) o continua(DC) desde tensión baja hasta 110V.

1.2.14. TARJETA DE ADQUISICIÓN DE DATOS SALIDAS DIGITALES A TRANSISTOR A 24V.

Son salidas digitales con un transistor a colector abierto a la salida. Son iguales a las salidas digitales a transistor de los autómatas programables. Permite conmutar cargas de corriente continua (DC) a 24V.

1.2.15. TARJETA DE ADQUISICIÓN DE DATOS CON SALIDAS DIGITALES A TRIAC.

Son salidas digitales con un triac que permite conmutar cargas de corriente alterna (CA). Son iguales a las salidas digitales a *triac* de los autómatas programables.

1.3. TÉCNICAS DE TRANSFERENCIA DE DATOS EN TARJETA DE ADQUISICIÓN DE DATOS POR ENCUESTA, INTERRUPCIONES Y DMA.

Las tarjetas de adquisición de datos pueden ser utilizadas para controlar la adquisición de datos por medio de un proceso en bucle cerrado, o simplemente para medir distintas variables del proceso y almacenar su valor. En función de la aplicación se pueden utilizar diversas formas de temporizar la transferencia de los datos desde el dispositivo al PC.

Se pueden distinguir por una parte 3 modos de inicio ("*trigger*") de la conversión A/D. Éstos pueden ser:

- Por software. Comienza una conversión cuando desde el PC se escribe un *byte* determinado de la tarjeta.
- Por señal externa ("*trigger* externo"). Se inicia una conversión cuando una señal digital externa (de un detector por ejemplo) pasa de 0 a 1.
- Por temporizador, Después de un tiempo predeterminado se inicia una conversión cada vez que el temporizador termina su cuenta. Si el temporizador está configurado en funcionamiento cíclico cuando llega a cero se recarga con el valor inicial, por lo que se produce una conversión cada periodo.

Por otra parte se pueden distinguir 3 modos de transferencia de los datos convertidos desde la tarjeta al PC. Éstos son:

- Por software ("por encuesta"). Cuando termina la conversión la tarjeta pone a 1 un bit determinado. El programa debe comprobar si ese bit se ha puesto a 1 y en caso afirmativo leer el valor convertido desde una posición de memoria de la tarjeta.
- Por interrupción. Cuando se termina la conversión la tarjeta provoca una interrupción del procesador del PC. En la rutina de interrupción correspondiente se lee el valor convertido.
- Por DMA (*Direct Memory Acces*). Es una forma especial de transferencia de datos que no requiere de la intervención del procesador del PC. De forma directa el valor convertido es transferido por el controlador DMA de la placa base a la memoria RAM del PC.

Los tres modos de disparo y los tres modos de transferencia se suelen combinar para dar lugar a 3 modos de funcionamiento: Por encuesta, interrupción y DMA.

1.3.1 TRANSFERENCIA DE DATOS POR ENCUESTA.

Se llama también transferencia por *software*. El programa del PC solicita a la tarjeta que lea el valor de una entrada. Los instantes en que ésto sucede no están predeterminados, sino que dependen de la evolución del programa que se ejecuta en el PC. Ante la petición del PC, la tarjeta realiza una conversión, indicando después al PC la finalización de dicha conversión. Cuando el PC detecta que la conversión ha finalizado, lee ese valor convertido de una posición de memoria de la tarjeta, y lo almacena en la RAM del PC. La forma en que el PC le comunica a la tarjeta que desea leer una entrada consiste en escribir el byte adecuado en una posición de memoria determinada de la tarjeta. La forma en que la tarjeta comunica al PC que ha finalizado la conversión es poniendo a 1 un bit de una posición de memoria determinada de la propia tarjeta. El PC debe leer esa posición continuamente hasta que detecte que el bit se ha puesto a 1. Este procedimiento se puede utilizar cuando únicamente se quieren tomar datos del proceso para visualizarlos o tener una supervisión centralizada de lo que ocurre, pero no se controla el proceso en bucle cerrado desde el PC.

1.3.2. TRANSFERENCIA DE DATOS POR INTERRUPCIÓN.

Cuando está configurado este modo, se provoca una interrupción del procesador de la tarjeta cuando finaliza una conversión A/D. La orden de inicio de la conversión ("*trigger*") puede ser externa (cuando una señal digital externa cambia de 0 a 1), o interna (controlada normalmente por un temporizador de la tarjeta de adquisición de datos).

El "*trigger*" externo se puede utilizar cuando se quiere medir el valor de una variable cuando sucede un evento determinado en el proceso (cuando se activa un detector por ejemplo). El modo de interrupción también puede

utilizarse cuando sólo se quiere medir variables del proceso sin controlarlo en bucle cerrado desde el PC.

1.3.3 TRANSFERENCIA DE DATOS POR DMA (ACCESO DIRECTO A MEMORIA).

Para utilizar este tipo de transferencia se suele configurar el disparo por temporizador. De esta forma cada cierto tiempo se realiza una conversión. Cuando termina la conversión el dato se transfiere automáticamente a la RAM del PC sin intervención del procesador, gracias a la actuación de un chip especial de la placa base del PC, el controlador DMA. Para hacer funcionar la tarjeta de adquisición de datos con este modo de transferencia hay que configurar el modo escribiendo en la posición de memoria adecuada de la tarjeta, y hay que configurar el controlador DMA de la placa base para que realice la transferencia cada vez que se produce una conversión. Este modo de transferencia permite obtener medidas con un periodo muy pequeño, sin incrementar la carga del procesador.

2. EL BUS SERIAL UNIVERSAL (USB)

2.1 FUNCIONAMIENTO DEL BUS USB

El bus serial universal (USB) es un bus asíncrono, y utiliza el algoritmo de codificación NRZI ("No retorno a cero invertido"). En este sistema existen dos voltajes opuestos; una tensión de referencia corresponde a un "1", pero no hay retorno a cero entre bits, de forma que una serie de unos corresponde a un voltaje uniforme; en cambio los ceros se marcan como cambios del nivel de tensión, de modo que una sucesión de ceros produce sucesivos cambios de tensión entre los conductores de señal.

El controlador USB instalado en el PC, denominado controlador de *host,* o concentrador raíz ("*Hub* raíz"), proporcionan un enlace entre el bus de la placa-base y una o más conexiones iniciales con el exterior (generalmente 2 conectores del tipo "A"). A partir de éstas, utilizando concentradores adicionales, pueden conectarse más dispositivos USB.

Actualmente la mayoría de las placas-base incluyen un controlador USB integrado en el *chipset*. Para sistemas antiguos que no dispongan de bus USB pueden instalarse tarjetas PCI (e incluso *PC-CARD* para portátiles) que incluyen un controlador y uno o dos conectores de salida USB.

El protocolo de comunicación utilizado es de testigo, que guarda cierta similitud con el sistema *Token-Ring* de IBM. Puesto que todos los periféricos comparten el bus y pueden funcionar de forma simultanea, la información es enviada en paquetes; cada paquete contiene una cabecera que indica el periférico a que va dirigido. Existen cuatro tipos de paquetes distintos: *Token*; Datos; *Handshake*, y Especial; el máximo de datos por paquete es de 8; 16; 32; 64; 128; 512 y 1024 *Bytes*. Se utiliza un sistema de detección y

corrección de errores bastante robusto del tipo CRC ("chequeo de redundancia cíclica").

El funcionamiento del bus USB está centrado en el *host*, todas las transacciones se originan en él; el controlador *host* es el que decide todas las acciones, incluyendo el número asignado a cada dispositivo (esta asignación es realizada automáticamente por el controlador "*host*" cada vez que se inicia el sistema o se añade, o elimina, un nuevo dispositivo en el bus USB), su ancho de banda, etc. Cuando se detecta un nuevo dispositivo es el *host* el encargado de cargar los *drivers* oportunos sin necesidad de intervención por el usuario.

El bus USB utiliza cuatro tipo de transacciones que resuelven todas las posibles situaciones de comunicación. Cada transacción utiliza un mínimo de tres paquetes, el primero es siempre un *Token* que avisa al dispositivo que puede iniciar la transmisión.

Las comunicaciones asíncronas ponen más énfasis en garantizar el envío de datos, y menos en su temporización ("cuando" llegan); por su parte las comunicaciones isócronas son justamente lo contrario, ponen más énfasis en la oportunidad de la transmisión que en la velocidad. Esta sincronización es importante en situaciones como la reproducción de video, donde no debe existir desfase entre las señales de video y audio.

2.2 COMPONENTES DEL BUS USB

2.2.1. CONTROLADOR USB

El controlador USB del *host* reside dentro del PC y es responsable de las comunicaciones entre los periféricos USB y la CPU del PC. Es también

responsable de la admisión de los periféricos dentro del bus, tanto si se detecta una conexión como una desconexión. Para cada periférico USB añadido, el controlador determina su tipo y le asigna una dirección lógica para utilizarla en las comunicaciones con el *host* PC. Si se producen errores durante la conexión, el controlador lo comunica a la CPU, que, a su vez, lo transmite al usuario. Una vez se ha producido la conexión correctamente, el controlador asigna al periférico USB los recursos del sistema que éste precise para su funcionamiento. El controlador también es responsable del control de flujo de datos entre el periférico USB y la CPU.

2.2.2. PERIFÉRICOS USB

USB soporta periféricos de baja, mediana velocidad y alta velocidad. Se emplean tres velocidades para la transmisión de datos de 1.5 , 12 Mbps y 480Mbps respectivamente. Los periféricos de baja velocidad tales como teclados, ratones, *joysticks*, y otros periféricos para juegos intercambian datos con el controlador USB del *host* a 1.5 Mbps. Se puede dedicar más recursos del sistema a periféricos tales como monitores, impresoras, *módems*, *scanner*, equipos de audio que precisan de velocidades más altas para transmitir mayor volumen de datos cuya dependencia temporal sea más estricta como por ejemplo aplicaciones de video y audio en tiempo real .

2.3 ESTÁNDARES USB

Debido a que el bus USB ha sido promovido principalmente por Intel, aunque le han seguido otros grandes fabricantes de PCs, se ha convertido en un estándar importante. Desde sus comienzos los interesados en esta tecnología se agruparon en el forum, USB (USB-IF), que congrega a más de 460 compañías. El consejo directivo está formado por representantes de las

siguientes compañías: *Compaq Computer Corporation; Hewlett-Packard; Intel Corporation; Lucent Technologies; Microsoft Corporation; NEC Corporation y Philips*, y ha publicado diversas revisiones de la norma entre las que se destacan:

USB 0.9: Primer borrador, publicado en Noviembre de 1995.

USB 1.0: Publicada en 1996 establece dos tipos de conexión: La primera, denominada velocidad baja ("*Low speed*"), ofrece 1.5 Mbps, y está pensada para periféricos que no requieren un gran ancho de banda, como ratones o *joysticks.* La segunda, denominada velocidad completa (*full speed*) , es de 12Mbps, y está destinada a los dispositivos más rápidos.

USB 1.1: Publicada en 1998, añade detalles y precisiones a la norma inicial; es el estándar mínimo que debe cumplir un dispositivo USB actual.

USB 2.0: Su versión final fue publicada en Abril del 2000; es una extensión de la norma compatible con las anteriores. Permite velocidades de hasta 480Mbps, denominada alta velocidad ("*High speed*")

2.4 VELOCIDADES DEL BUS USB

El bus serial universal (USB) es una interface *plug & play* entre la PC y dispositivos externos tales como teclados, *mouses*, *scanner*, impresoras, *módems*, cámaras, memorias, etc.) . y permite instalar estos periféricos sin tener que abrir el equipo para instalar el hardware, ya que es suficiente con conectar dicho periférico detectándolo y configurándolo el sistema operativo. Una característica importante de USB es que permite a los dispositivos trabajar a velocidades mayores, en la versión USB 1.1 se especifican dos modos de transferencia una a baja velocidad(*low-speed*) del bus a 1,5 Mbps y otra a velocidad media (*full-speed*) de 12 Mbps; en la versión USB 2.0 se

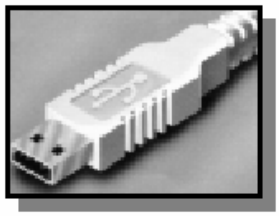
obtiene una transferencia a 480 Mbps, denominado USB de alta velocidad (*high-speed*).

Los dispositivos USB identifican su velocidad eléctricamente, mediante una resistencia de *pull-up* en la línea D+ (*full-speed*) o D- (*low-speed*). En cambio los dispositivos high-speed se identifican en principio como *full-speed* (mediante un *pull-up* en la línea D+), y durante el proceso de *reset* ejecutan un protocolo de bajo nivel a través del cual se determinan si están conectados a un puerto *full-speed* o *high-speed*. Si el puerto es *high-speed*, detectará y responderá al protocolo iniciado por el dispositivo. Sólo si el concentrador y el dispositivo ejecutan el protocolo, se establece una comunicación *high-speed* entre ellos.

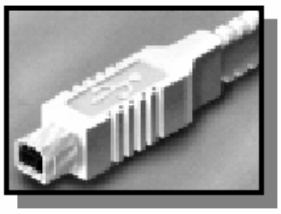
2.5 CABLES Y CONECTORES USB

Desde la perspectiva del usuario, los puertos e interfaces USB son muy sencillos de emplear, comenzando por los cables, cuyos conectores sólo son de dos tipos e imposibles de colocar de manera errónea mostrados en la figura 1. Cuando un dispositivo nuevo USB se asocia a un PC, el sistema operativo detecta su presencia e instala el controlador correspondiente o bien, puede solicitar al usuario el disco de instalación de ese periférico. Después de trabajar con el dispositivo, el usuario puede desconectarlo directamente del puerto USB, sin riesgo de perder la configuración o dañar el equipo.

Figura 1. Tipo de Conectores USB



Tipo A. Macho



Tipo B. Macho

Tipo A. Hembra

Tipo B. Hembra

Fuente: Molex Inc.

Como se observa en la tabla 1, un cable USB está compuesto por cuatro conductores: dos de alimentación y dos de datos, rodeados de una capa de blindaje para evitar interferencias. Por los conductores de alimentación pueden proporcionarse cinco voltios a aquellos dispositivos que así lo requieran sin necesidad de fuente externa (como cámaras de videoconferencia y lectores de tarjetas de memoria), o puede recibir la alimentación externa a dispositivos con mayor consumo de energía (impresoras, discos, quemadores) .

Existen dos tipos de cables: apantallado y sin apantallar. En el primer caso el par de hilos de señal es trenzado; los de tierra y alimentación son rectos, y la cubierta de protección (pantalla) solo puede conectarse a tierra en el PC. En el cable sin apantallar todos los hilos son rectos. Las conexiones a 1.5 Mbps y superiores exigen cable apantallado.

Tabla 1. descripción de los cables USB

Pin	Nombre	Descripción	Color
1	VBUS	+ 5 VCC	rojo
2	D-	Datos -	azul
3	D+	Datos +	amarillo
4	GND	Tierra	verde

Fuente: *Universal Serial Bus Specification rev 1.1*

Cuando se deben conectar más dispositivos USB al *host* controlador USB del PC, es indispensable usar un concentrador USB (*Hub*). El concentrador amplía la cantidad de puertos disponibles para otros dispositivos USB. Un solo PC, combinando cables de no más de cinco metros de longitud cada uno y concentradores, puede tener asociados hasta 127 dispositivos USB.

La justificación para que los cables y conectores tan sencillos puedan transmitir hasta 480 Mbps reside en el controlador USB (*host*), un circuito residente en el PC. Al momento de activarse busca a todos los dispositivos conectados al bus USB y les asigna una dirección lógica variable. Este proceso se llama enumeración, manteniéndose aún después del arranque. De manera adicional, el controlador USB detecta el tipo de transferencia de datos aceptable por el dispositivo externo USB, por ejemplo:

1. Datos por interrupción. Dispositivos como ratones y teclados no requieren enviar grandes volúmenes de información, tan sólo pequeñas tramas de datos.

2. Datos en Bloque. Las impresoras reciben mayores volúmenes de datos, generalmente en bloques de 64,128, 512, y 1024 bytes, siendo cada bloque verificado para tener la garantía de una buena recepción.
3. Datos Síncronos. Aquellos aparatos que operan con información constante (audio y video) bocinas, pantallas y que no se verifica su correcta recepción.

Si el PC tiene un controlador USB 1.1 y se desea migrar a un controlador USB 2.0, será necesario adquirir e instalar una nueva tarjeta controladora. Cuando se utilizan dispositivos *high-speed* y se conectan a un puerto USB del PC que trabaja en modo *full-speed*, éstos dispositivos pueden soportar una mínima funcionalidad permitiendo que el PC pueda detectarlos, identificarlos y configurarlos como dispositivos *full speed*.

El calibre de los conductores destinados a alimentación de los periféricos varía desde 20 a 26 *AWG*, mientras que el de los conductores de señal es de 28 AWG. La longitud máxima de los cables es de 5 metros.

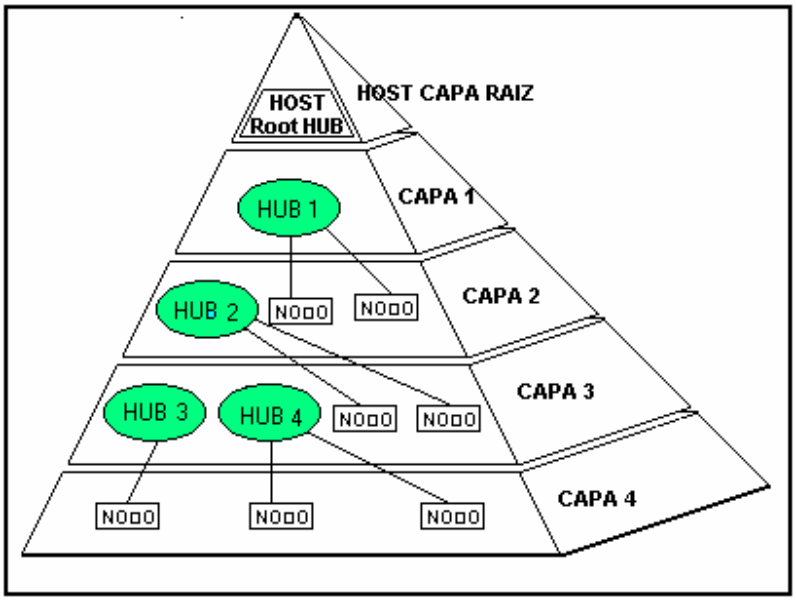
2.6 TOPOLOGÍA DEL BUS USB

El bus USB soporta el intercambio simultaneo de datos entre un PC y un amplio conjunto de periféricos. Todos los periféricos conectados al PC comparten el ancho de banda del bus por medio de un protocolo de arbitraje basado en testigos ("*Token*"). El bus USB permite la conexión y desconexión dinámica, es decir, que los periféricos se conecten, configuren, manipulen y desconecten mientras el sistema operativo y otros periféricos permanecen en funcionamiento.

La topología del bus USB adopta la forma de estrella y se organiza por niveles. En un bus USB existen dos tipos de elementos: Anfitrión ("*host*") y

dispositivos USB externos; a su vez, los dispositivos pueden ser de dos tipos: concentradores *Hub* y periféricos USB convencionales. Algunos dispositivos USB pueden ser de los dos tipos al mismo tiempo. Por ejemplo, una pantalla de video USB puede ser a su vez un concentrador *Hub* con dos o más conexiones auxiliares para conectar otros dispositivos.

Figura 2. Jerarquía de dispositivos USB



Fuente: *Universal Serial Bus Specification rev 1.1*

Un periférico USB es un dispositivo capaz de transmitir o recibir datos o información de control en un bus USB, suele conectarse como un dispositivo independiente enlazado por un cable de menos de 5 metros, a un puerto del hub o directamente al PC.

De esta descripción se desprende que cada segmento del bus representa una conexión punto a punto de alguno de los tipos siguientes: Que un *hub* pueda estar conectado a otro hub, significa que pueden conectarse dispositivos en cascada; el sistema soporta un total de 127 dispositivos.

Una característica importante es que el PC o el concentrador Hub puede proporcionar la energía necesaria al dispositivo por el cable de conexión (que transporta alimentación y datos), lo que evita la necesidad de fuentes de alimentación independientes. Si el dispositivo o los dispositivos adheridos al bus USB consumen menos de 100 mA.

Tabla 2. Esquema de conexión de periféricos USB

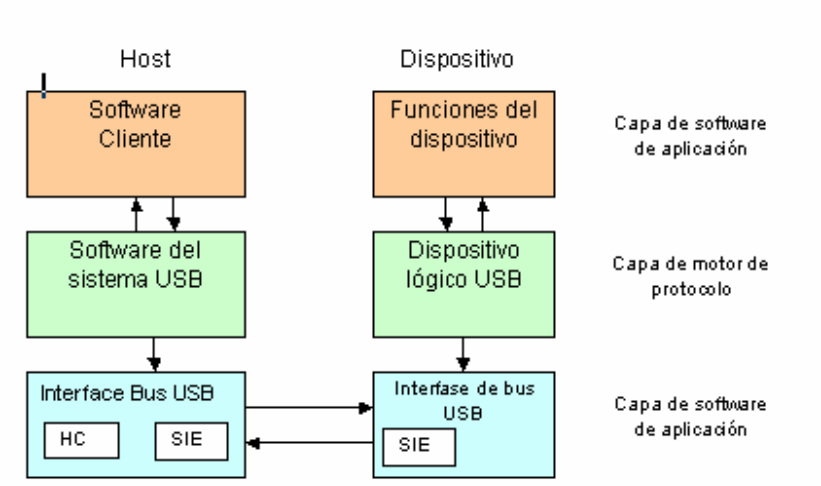
Sistema anfitrión ⇔ periféricos USB
Sistema anfitrión ⇔ Concentrador USB(Host)
Concentrador USB(Host) ⇔ Concentrador USB(Hub)
Concentrador USB(Hub) ⇔ periféricos USB

Fuente: *Universal Serial Bus Specification rev 1.1*

2.7 CAPAS DE LA COMUNICACIÓN USB

El bus USB soporta una serie tres capas del modelo OSI las cuales se muestran en la Figura 3. Cada capa tiene una función especifica y trabaja en conjunto con las otras capas para hacer las comunicaciones USB transparentes al usuario final.

Figura 3. Capas de la comunicación USB



Fuente: *Universal Serial Bus Specification rev 2.0*

2.8 LA CAPA FÍSICA

La capa física forma la base de la comunicación USB y consiste del hardware que transmite las señales diferenciales en el protocolo (NRZI). En el *host* la capa física se divide en dos partes: el motor de interfaz serial (SIE) y el controlador de *host* (HC). El motor de interfaz serial desarrolla las funciones serialización de las transmisiones, codificación y decodificación de las señales, chequeo de redundancia cíclica (CRC) y la detección del identificador de paquete (PID). El controlador del host inicializa las transacciones y controla el acceso al bus USB, además divide el tiempo en tramas de datos.

2.9 CAPA DE MOTOR DE PROTOCOLO

Esta capa intermedia se refiere al software del sistema USB en el *host* y a la lógica del dispositivo USB en el dispositivo USB externo. El *software* del

sistema USB incluye el ancho de banda y maneja la alimentación del bus. Provee la interfaz para el controlador del *host*, además, identifica, enumera y sirve a requerimientos de datos de dispositivos en el bus USB. El dispositivo lógico USB esta compuesto de uno o más *endpoints*[5] unidireccionales que transfieren datos. La tarjeta de adquisición de datos USB tiene un *endpoint* para salida de datos y uno para entrada de datos así como un *endpoint* de configuración. Cada *endpoint* tiene un cierto tipo de transferencia de datos.

2.10 CAPA DE APLICACIÓN

En el *host* la capa de aplicación es el *software* cliente a través del cual el usuario interactúa con el dispositivo USB. Desde el punto del dispositivo USB la capa de aplicación es simplemente las funciones que el dispositivo desempeña. Para la tarjeta de adquisición de datos diseñada esta capa es el driver de *FTDI* en el *host* PC, y las funciones de lectura y escritura en el controlador USB FT232.

2.11 TRAMAS Y MICROTRAMAS USB

En la versión USB 1.1 se dividía el tiempo en tramas de 1 milisegundo. Por su parte, la versión USB 2.0 define un tiempo de microtrama de 125 µs. En USB 2.0 también se reservan ciertos porcentajes del tiempo de microtrama, para dar servicio a los distintos tipos de transacciones de alta velocidad *high-speed*.

[5] ENDPOINT: Es un buffer que almacena múltiples bytes. Típicamente es un bloque de datos de memoria o un registro en el chip controlador USB. Transportar datos en una sola dirección, sin embargo los endpoint de control son bidireccionales.

2.12 TIPOS DE TRANSFERENCIAS DE DATOS USB

2.12.1 TRANSFERENCIAS DE CONTROL

Estos tipos de transferencias proporcionan control de flujo y una entrega de datos garantizada y libre de errores. Todos los dispositivos *full, high y low-speed* pueden incorporar *endpoints* de Control, y por lo tanto pueden hacer uso de las transferencias de Control. Todos implementan, al menos, un *endpoint* de salida y uno de entrada en la dirección 0, para poder establecer la transacción de Control por Defecto.

Las transferencias de Control se componen de 3 transacciones denominadas Configuración*(Setup)*-Datos-Estado. Los tamaños máximos del paquete de datos durante la transacción de datos son:

· Full-speed: 8, 16, 32, ó 64 bytes
· High-speed: 64 bytes
· Low-speed: 8 bytes.

El bus USB hace una gestiona el mejor esfuerzo para ir dando curso a las distintas transferencias de Control pendientes en cada momento en todas los *pipes*[6] de Control establecidos con todos los dispositivos. Para ello se hace la siguiente reserva del tiempo de trama o microtrama:

- En un bus USB de baja o mediana velocidad (*full/low-speed)*, la reserva es del 10% del tiempo de trama.
- En un bus USB de alta velocidad (*high-speed)*, la reserva es del 20% del tiempo de microtrama.

[6] PIPES: Un Pipe USB no es un objeto físico; sino una asociación entre el endpoint del dispositivo USB y el *software* controlador del PC. El Pipe es un puente lógico por donde viajan los datos en el bus USB.

Las reglas definidas por USB para el envío de las transferencias pendientes son:

- Si el tiempo de trama o microtrama utilizado por las transferencias de Control pendientes es inferior al reservado, el tiempo restante puede utilizarse para transferencias en bloque (*Bulk*).
- Si hay más transferencias de Control pendientes que tiempo reservado, pero hay tiempo adicional en la trama o microtrama no consumido por transferencias de Interrupción o Isócronas, entonces el *host* puede utilizar dicho tiempo adicional para enviar nuevas transferencias de Control.
- Si hay más transferencias de Control pendientes que tiempo disponible en una trama o microtrama, el *host* selecciona cuáles se procesan, quedando el resto pendientes para una próxima trama o microtrama.

Los *endpoints* de Control de alta velocidad (*high-speed)* soportan el protocolo de control de flujo *PING* en las transacciones de Dato y Estado de salida.

2.12.2 TRANSFERENCIAS ISÓCRONAS

Las transferencias de datos Isócronas están diseñadas para soportar aquellos dispositivos que precisan una entrega de datos a velocidad constante, y en las que no importa la pérdida eventual de información (audio y video). Esto es necesario para aplicaciones en las que la información de tiempo va implícita en la propia velocidad de transmisión / recepción de datos (isocronismo).Para ello, las transferencias Isócronas proporcionan el ancho de banda garantizado, latencia limitada, velocidad de transferencia de datos constante garantizada a través de el *pipe* y en caso de error en la entrega, no se reintenta la transmisión. No tienen control de flujo. Sólo los dispositivos de alta y mediana velocidad (*high y full-speed*) pueden incorporar endpoints

Isócronos. Las transferencias Isócronas se componen sólo de transacciones de datos. Las frecuencias y los tamaños de los paquetes de datos son:

- Mediana Velocidad *(Full-speed)*: 1 transacción por trama de hasta 1,023 bytes.
- Alta velocidad *(High-speed)*: 1 transacción por microtrama de hasta 1,024 bytes.
- Alta velocidad (*High-speed)* con alto ancho de banda *(high-bandwidth)*: 2 ó 3 transacciones por microtrama de hasta 1,024 *bytes* cada una.

La gestión que hace el bus USB para garantizar las transferencias es la de establecer o no, el *pipe* en función de que haya suficiente tiempo libre de trama o microtrama para realizarlas. Para ello, los *endpoints Isócronos* indican qué cantidad de información como máximo debe transferir la *pipe* en cada trama o microtrama, de forma que el sistema USB puede calcular si hay suficiente tiempo o no para acomodar el *pipe*, y en función de eso la establece o no. La reserva de tiempo de trama o microtrama para acomodar transferencias Isócronas y de Interrupción es como máximo el tiempo no reservado para transferencias de Control. El sistema USB puede ir estableciendo *pipes* Isócronas y de Interrupción con distintos dispositivos hasta agotar dicha reserva:

· Velocidad Mediana *(Full-speed):* Hasta un 90% del tiempo de trama.
· Alta Velocidad (*High-Speed*): Hasta un 80 % del tiempo de microtrama.

2.12.3 TRANSFERENCIAS DE INTERRUPCIÓN

Las transferencias de Interrupción están diseñadas para soportar aquellos dispositivos que precisan enviar o recibir datos de manera no frecuente, pero

con ciertos límites de latencia. Para ello, las transferencias de Interrupción proporcionan: Tiempo máximo de servicio (latencia) garantizado, reintento de transferencia en el siguiente periodo, en caso de un eventual fallo en la entrega de datos. Todos los dispositivos USB de alta velocidad, mediana y baja velocidad pueden incorporar *endpoints* de Interrupción. Las transferencias de Interrupción se componen sólo de transacciones de datos. Los tamaños de los paquetes de datos son:

· Baja velocidad *(Low-speed*): hasta 8 bytes.
· Mediana Velocidad (*Full-speed*): hasta 64 bytes.
· Alta velocidad (*High-speed*): hasta 1,024 bytes.
· Alta velocidad (*High-speed*)con gran ancho de banda (*high-bandwidth*): 2 ó 3 transacciones por microtrama de hasta 1,024 bytes cada una.

La gestión que hace el USB para garantizar las transferencias es la de establecer o no el *pipe* en función de que haya suficiente tiempo libre de trama o microtrama para realizarlas. Para ello, los *endpoints* de Interrupción indican qué cantidad de información como máximo debe transferir el *pipe* en cada transacción, así como el tiempo máximo entre transacciones, de forma que el sistema USB puede calcular si hay suficiente tiempo o no para acomodar la *pipe*, y en función de eso la establece o no.

El tiempo máximo entre transacciones (tiempo de latencia máximo) especificado por cada dispositivo puede ser:

· Baja velocidad (*Low-speed*): de 10 a 255 ms.
· Mediana Velocidad (*Full-speed*): de 1 a 255 ms.
· Alta velocidad (*High-speed*): de 125 us a 1 ms.

La reserva de tiempo de trama o microtrama para acomodar transferencias Isócronas y de Interrupción es como máximo el tiempo no reservado para transferencias de Control. El sistema USB puede ir estableciendo pipes Isócronas y de Interrupción con distintos dispositivos hasta agotar dicha reserva:

· Mediana Velocidad (*Full y Low-speed*): Hasta un 90% del tiempo de trama.
· Alta velocidad (*High-Speed*): Hasta un 80 % del tiempo de microtrama.

2.12.4 TRANSFERENCIAS EN BLOQUE (*BULK*)

Las transferencias en bloque (*Bulk)* están diseñadas para soportar aquellos dispositivos que precisan enviar o recibir grandes cantidades de datos, con latencias que pueden tener amplias variaciones, y en que las transacciones pueden utilizar cualquier ancho de banda disponible. Para ello, las transferencias en bloque *(Bulk)* proporcionan: Acceso al bus en función del ancho de banda disponible, reintento de transferencias en caso de errores de entrega, entrega garantizada de datos, pero sin garantía de latencia máxima ni de ancho de banda.

Las transferencias en bloque *(Bulk)* se realizan relativamente de manera rápida si el bus dispone de mucho ancho de banda libre, pero en un bus USB con mucho ancho de banda reservado, pueden alargarse durante periodos de tiempo relativamente grandes. Sólo los dispositivos de alta y mediana velocidad *(high y full-speed)* pueden incorporar *endpoints* tipo *Bulk*. Las transferencias en bloque *(Bulk)* se componen sólo de transacciones de datos. Los tamaños de los paquetes de datos son:

- Mediana Velocidad *(Full-speed)*: 8,16, 32 y 64 bytes.
- Alta Velocidad *(High-speed)*: 512 bytes.

El bus USB hace una gestión “del mejor esfuerzo” para ir dando curso a las distintas transferencias pendientes en cada momento en todas los *pipes* tipo *Bulk* establecidas con todos los dispositivos. Las transferencias de Control tienen preferencia sobre las *Bulk*, por lo que las transferencias *Bulk* se realizan siempre que no haya otro tipo de transferencias que hacer en una trama o microtrama. Los endpoints de salida *(Bulk-OUT)* para dispositivos de alta velocidad (*high-speed)* soportan el protocolo de control de flujo *PING*

2.13 INICIACIÓN DE UNA TRANSFERENCIA USB

Cuando un *driver*[7] de dispositivo USB en el h*ost* PC busca comunicarse con el dispositivo USB éste inicia una transferencia. Las especificaciones definen a una transferencia como el proceso de hacer, transformar y llevar hacia fuera una comunicación requerida. Una transferencia puede ser muy corta, enviando unos pocos bytes de datos, o muy larga, enviando el contenido de un archivo muy grande.

Típicamente las aplicaciones de *Windows* inician una comunicación con un dispositivo USB usando las funciones *API* estándar del sistema operativo. Para comenzar una transferencia, una aplicación puede llamar a una función *API* para requerir la transferencia desde el driver del dispositivo USB. Las aplicaciones pueden requerir datos desde el dispositivo o enviar datos hacia el dispositivo. Una forma de pedir datos desde la aplicación podría ser enviar el contenido de un archivo de datos (.txt) en el *host* o enviar el contenido de datos al dispositivo. Cuando una aplicación requiere una transferencia USB, el sistema operativo pasa este requerimiento al driver del dispositivo apropiado el cual pasa este requerimiento a otro nivel del driver del sistema

[7] DRIVER: Es un componente de software que habilita a las aplicaciones de alto nivel a acceder a un dispositivo de hardware.

en el *Host* controlador USB. El *host* controlador USB en este momento inicia la transferencia de datos sobre el bus USB.

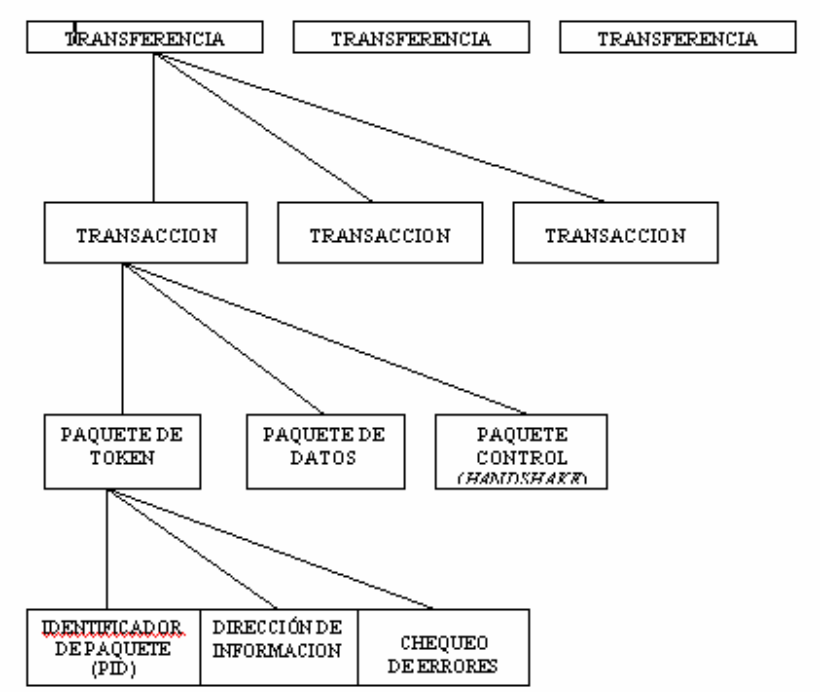
En algunos casos, el *driver* esta configurado para permitir transferencias periódicas, y las aplicaciones leen los datos de entrada o colocan datos en estas transferencias. Otras transferencias tales como las que se hacen en el momento de la enumeración, son iniciadas por el sistema operativo al momento de detectar el dispositivo USB.

2.13.1. BLOQUES QUE CONSTITUYEN UNA TRANSFERENCIA

La figura 4. especifica los elementos de una transferencia típica, y la tabla 3. lista los elementos que permiten cada uno de los tipos de transferencia de datos. La figura 4 muestra las transferencias y las transacciones, los estados y las fases de cada transacción y los paquetes de datos, estado y la fase *handshake.* La fase de datos tiene paquetes de handshake y la fase de estado (*estatus)* tiene paquetes de datos.

Cada transferencia consiste de una o más transacciones y cada transacción consiste de uno, dos o tres paquetes. La tabla 3. muestra los cuatro tipos de transferencias y cada estado esta compuesto de dos o tres fases. Los tres tipos de transacciones están definidos según el propósito y la dirección del flujo de datos: configuración (*Setup*) para enviar una transferencia de control al dispositivo USB, entrada (*In*) para recibir datos del dispositivo USB y salida (O*ut*) para enviar otros datos al dispositivo. Las especificaciones USB definen una transacción como una distribución del servicio para un *endpoint* . El servicio en este caso puede ser la mitad del ancho de banda de cualquiera de los *host* que estén enviando una trama de información hacia el dispositivo USB o los *host* que requieran y reciban una trama de información desde el dispositivo.

Figura 4. Elementos de una Transferencia USB



Fuente: *Universal Serial Bus Specification rev 1.1*

Cada transacción incluye identificadores como el chequeo de errores, estado, e información de control y también cualquier tipo de datos puede ser intercambiado. Una transferencia completa puede tomar lugar sobre múltiples paquetes *(frames),* pero una transacción es una comunicación única y debe ser completada sin interrupciones. Ninguna comunicación sobre el bus puede cortarse en medio de una transacción.

Tabla 3. Tipos De Transferencias Para El Bus USB

TIPO DE TRANSFERENCIA	ESTADOS TRANSACCIONES	FASES (PAQUETES)
CONTROL	SETUP	TOKEN DATA HANDSHAKE
	DATO(ENTRADA O SALIDA) (OPCIONAL)	TOKEN DATA HANDSHAKE
	ESTADOS(ENTRADA O SALIDA)	TOKEN DATA HANDSHAKE
BULK	DATO(ENTRADA O SALIDA)	TOKEN DATA HANDSHAKE
INTERRUPCION	DATO(ENTRADA O SALIDA)	TOKEN DATA HANDSHAKE
ISOCRONA	DATO(ENTRADA O SALIDA)	TOKEN DATA

Fuente: *Universal Serial Bus Specification rev 2.0*

Los dispositivos USB deben estar disponibles para responder rápidamente a cualquier requerimiento de datos o información de estado en una transacción. El código de programa en el dispositivo USB puede preparar un *endpoint* para responder a una transacción que sea requerido, pero el hardware es quien toma esa respuesta cuando el requerimiento llega. Una

transferencia con una pequeña cantidad de datos puede tener una única transacción. Si la cantidad de datos es grande, una transferencia puede usar múltiples transacciones con una porción de datos en cada una de ellas

2.13.2. FASES EN UNA TRANSACCIÓN

Cada transacción USB tiene hasta tres fases que ocurren en la siguiente secuencia: *Token,* Datos, *Handshake*. Cada fase consiste de uno o dos paquetes de transmisión. Cada paquete es un bloque de información con un formato definido. Todos los paquetes inician con un identificador de paquete(PID) que contiene información de identificación como se muestra en la tabla 4 . Dependiendo de la transacción, el identificador de paquete*(PID)* puede estar seguido, por la dirección a un *endpoint,* datos, información de estado, o un número de paquete (*frame)* alojando unos bit de chequeos de errores(CRC).

En la fase de *Token* de una transacción, el h*ost* envía un requerimiento de comunicación en un paquete de *Token*. El identificador de paquete indica el tipo de transacción, tal como un paquete de inicio de configuración, entrada o salida.

En la fase de datos, el h*ost* o el dispositivo pueden transferir cualquier tipo de información en un paquete de datos. El identificador de paquete indica la posición de los datos cuando hay múltiples paquetes de datos. En la fase de *Handshake* el *Host* o el dispositivo envían información de estado (o *handshake)* en un paquete de *handshake.* El identificador de paquete (PID) mantiene los códigos de estado (ACK, NAK, STALL, NYET). Algunas veces, las especificaciones usan los términos fase de estado y paquetes de estado para referirse a la fase y los paquetes de Handshake .

Tabla 4. Identificadores De Paquete

TIPO DE PAQUETE	NOMBRE PID	VALOR	TRANSFERENCIA USADA EN	FUENTE	VELOCIDAD BUS	DESCRIPCION
TOKEN	OUT	0001	TODOS	HOST	TODOS	Dirección del endpoint para una transacción de salida (Host al dispositivo USB)
	IN	1001	TODOS	HOST	TODOS	Dirección del endpoint para una transacción de entrada (dispositivo al host)
	SOF	0101	INICIO DE FRAME	HOST	TODOS	Marcador de inicio de frame y número de frame
	SETUP	1101	CONTROL	HOST	TODOS	Dirección del endpoint para una transacciones de configuración (SETUP)
DATOS	DATO 0	0011	TODOS	HOST DISPOSITIVO	TODOS	Secuencia de datos (DATATOGGLE)
	DATO 1	1011	TODOS	HOST DISPOSITIVO	TODOS	Secuencia de datos (DATATOGGLE)
	DATO 2	0111	ISOCRONAS	HOST DISPOSITIVO	ALTA	Secuencia de datos
	µTRAMA DE DATOS	1111	ISOCRONAS INTERRUPCION	HOST DISPOSITIVO	ALTA	Secuencia de datos
Handshake	ACK	0010	TODOS	HOST DISPOSITIVO	TODOS	Receptor acepta paquete de datos libres de errores
	NAK	1010	CONTROL (BULK) INTERRUPCION	DISPOSITIVO	TODOS	Receptor no puede aceptar datos no puede enviar datos o no tiene datos para transmitir
	STALL	1110	CONTROL BULK INTERRUPCION	DISPOSITIVO	TODOS	El requerimiento de control no es soportado o el endpoint esta interrumpido
	NYET	0110	CONTROL DE ESCRITURA BULK OUT TRANSACCIONES SPLIT	DISPOSITIVO	ALTA	El dispositivo acepta paquetes de datos libre de errores pero no puede estar listo todavía para otro o el hub todavía no ha completado los datos de SPLIT

Fuente : *Universal Serial Bus Specification rev 2.0*

La fase de Token tiene un uso adicional. Un paquete de *Token* puede transportar un paquete *(frame)* de inicio (SOF), el cual es un tiempo de referencia que el host envía a intervalos de 1ms para dispositivos USB de mediana velocidad y para dispositivos de alta velocidad es enviado cada 125 µs. Este paquete además contiene un número de paquetes que incrementa hasta alcanzar el máximo. El número indica el contador de *frame* mientras que 8 microtramas dentro de un paquete usan el mismo número de paquete *(frame)*. Un *endpoint* puede estar sincronizado con un paquete de inicio de *frame*(SOF) y usar el contador de frame como tiempo de referencia.

El marcador de inicio de paquete (SOF) además ayuda a los dispositivos USB para entrar al modo de bajo consumo de alimentación en el estado suspendido cuando no hay tráfico USB. Los dispositivos de baja velocidad no ven el paquete SOF en cambio los dispositivos *hub* usan un simple final de paquete (EOP) llamado la señal de actividad para dispositivos de baja velocidad y es enviado una vez por frame. Como el paquete SOF se hizo para dispositivos de mediana velocidad, para dispositivo de baja velocidad permite entrar a estado suspendido.

De los cuatro indicadores de paquetes especiales (PID),mostrados en la tabla 5 uno de ellos es usado únicamente con dispositivos de baja velocidad, uno es usado únicamente con dispositivos de alta velocidad y los otros dos son para *hub USB2.0* para dispositivos de mediana y alta velocidad que se comunican a alta velocidad con el *host*. El identificador de paquete especial para dispositivos de baja velocidad es (PRE), el cual contiene un código de preámbulo que le dice al *hub* que el siguiente paquete es de baja velocidad el *hub* debe habilitar la comunicación con un dispositivo conectado de baja velocidad.

Tabla 5. Identificador De Paquete Especial

TIPO DE PAQUETE	NOMBRE PID	VALOR	TRANSFERENCIA USADA EN	FUENTE	VELOCIDAD BUS	DESCRIPCION
	PRE	1100	CONTROL INTERRUPCION	HOST	MEDIA	Preámbulo emitido por el Host para indicar que el siguiente paquete es de baja velocidad identificador de paquete
	ERR	1100	TODAS	DISPOSITIVO HUB	ALTA	Retornado por un Hub para reportar un error de un dispositivo de baja o mediana velocidad en una transacción por partes(SPLIT)
	SPLIT	1000	TODAS	HOST	ALTA	Precede a un paquete Token para indicar una transacción por partes(SPLIT)
	PING	0100	CONTROL DE ESCRITURA, BULK DE SALIDA	HOST	ALTA	identificador de paquete que indica que esta ocupado se usa con transferencias BULK de salida, control de datos de escritura después del indicador del paquete NYET
ESPECIAL	RESERVADO	0000				reservado para uso futuro

Fuente: *Universal Serial Bus Specification rev 2.0*

En un bus de baja o mediana velocidad el identificador de paquete PRE precede todos los paquetes de *Token*, datos y *Handshake* dirigidos para dispositivos de baja velocidad. Los buses de alta velocidad USB codifican el identificador de paquete PRE en un paquete *SPLIT* de esta manera ellos no son enviados separadamente. Los paquetes enviados por un dispositivo de baja velocidad no requieren un identificador de paquete PRE.

El identificador de paquete usado únicamente con dispositivos de alta velocidad es *PING*. El *Host* envía un paquete *PING* para encontrar si un *endpoint* de un dispositivo de alta velocidad esta ocupado antes de enviar el siguiente paquete de datos en una transferencia de control o de tipo *Bulk* con múltiples paquetes de datos. El dispositivo responde con un código de estado*(estatus*).

El *PID SPLIT* identifica un paquete *Token* como parte de una transacción separada. Para usar mejor el tiempo en un bus USB los *Host* USB y los *Hub* envían trafico de baja y mediana velocidad a alta velocidad. Cuando el Host inicia una transacción destinada para dispositivos de baja o mediana velocidad, el *hub* USB 2.0 más cercano al dispositivo es el responsable de completar la transacción con el dispositivo USB, almacenando y retornando datos o información de estado, y reportando esto en una o más transacciones posteriormente. De esta forma el bus no tiene que esperar que una transacción se halla completado para un dispositivo de baja velocidad. Estas transacciones especiales entre el *Hub* y el *Host* son llamadas transacciones Separadas (*SPLIT).*

El identificador de paquete *ERR* es usado únicamente en transacciones de tipo *SPLIT*. Un *hub* USB 2.0 usa este identificador de paquete para reportar un error al *host* en una transacción de baja o mediana velocidad. El identificador de paquete *ERR* y *PRE* tienen el mismo valor, pero no deben ser confundidos debido a que un Hub nunca envía un identificador de paquete *PRE* al *host* o un identificador de paquete *ERR* al dispositivo.

3. DISPOSITIVOS DE HARDWARE PARA DISEÑO DE SISTEMAS DE ADQUISICION DE DATOS USB.

3.1. MICROCONTROLADOR

Un microcontrolador es un circuito integrado programable que contiene todos los componentes lógicos de un computador, se emplea para realizar una tarea determinada para la cual ha sido programado. Dispone de procesador, memoria para el programa y los datos, líneas de entrada y salida de datos (puertos) y suele estar asociado a múltiples recursos auxiliares. *Freescale Semiconductor*, tiene en el mercado mundial una serie de microcontroladores de 8 bits con tecnología *FLASH*. La familia HC08, basa su funcionamiento en la poderosa arquitectura del procesador central CPU08 (común a toda la familia. MC68HC08). Este procesador es del tipo arquitectura *Von Neuman* con mapa lineal de memoria sin diferenciación entre memoria de datos (RAM) y programa (*flash*), con lo cuál no existe limitaciones en el uso de instrucciones tanto para la memoria de datos, como para la de programa.

Para el presente trabajo se seleccionó el microcontrolador MC68HC908GP32 de la familia MCUs (*MicroController Units*) M68HC08 de Freescale, el cual es un miembro de bajo costo, alto desempeño y posee las características necesarias para el desarrollo del sistema de adquisición de datos por bus USB.

3.1.1 ADMINISTRACIÓN DE LA CPU

3.1.1.1 CARACTERÍSTICAS PRINCIPALES:

- Código compatible con la familia HC05 de *Freescale* (Motorola).
- Velocidad de bus de 8Mhz. Ciclo de instrucción mínimo de 125 ns
- Puntero de pila de 16 bits, con instrucciones de manipulación del *stack*.

- Registro índice de 16 bits, con instrucciones para manipular dicho índice.
- 78 instrucciones nuevas (comparado con el HC05)

Tabla 6. Mapa de memoria del microcontrolador MC68HC908GP32

$0000	REGISTRO I/O
$003F	64 BYTES
$0040	RAM
$023F	512 BYTES
$0240	NO IMPLEMENTADO
$7FFF	32192 BYTES
$8000	MEMORIA FLASH
$FDFF	32256 BYTES
$FE00	SIM REGISTRO DE ESTADO DE ALIMENTACION(SBSR)
$FE01	SIM REGISTRO DE ESTADO DE RESET (SRSR)
$FE02	RESERVADO(SUBAR)
$FE03	BANDERA DE CORTE DE ALIMENTACION(SBFCR)
$FE04	REGISTRO DE ESTADO DE INTERRUPCION 1(INT1)
$FE05	REGISTRO DE ESTADO DE INTERRUPCION 2 (INT2)
$FE06	REGISTRO DE ESTADO DE INTERRUPCION 3(INT3)
$FE07	RESERVADO
$FE08	REGISTRO DE CONTROL DE MEMORIA FLASH(FLCR)
$FE09	REGISTRO DE DIRECCIONES ALTAS(BRKH)
$FE0A	REGISTRO DE DIRECCIONES BAJAS(BRKL)
$FE0B	REGISTRO DE CONTRO Y ESTADO DE CORTE
$FE0C	REGISTRO DE ESTADO LVI (LVISR)
$FE0D	NO IMPLEMENTADO
$FE0F	3 BYTES
$FE10	NO IMPLEMENTADO 16 BYTES
$FE1F	RESERVADO
$FE20	MONITOR DE LA ROM
$FF52	307 BYTES
$FF53	NO IMPLEMENTADO
$FF7D	43 BYTES
$FF7E	BLOQUE PROTEGIDO DE LA MEMORIA FLASH(FLBPR)
$FF7F	NO IMPLEMENTADO
$FFDB	93 BYTES
$FFDC	VECTORES MEMORIA FLASH
$FFFF	36 BYTES

Fuente: Manual de usuario MC68HC908GP32 Freescale

- Instrucciones para mover datos entre posiciones de memoria, sin uso del acumulador.
- Multiplicación en 8 bits y división entera/fraccional.
- Control de bucles Instrucciones mejoradas para manipular números BCD.
- 16 modos de direccionamiento, incluyendo uno relativo al *stack*.
- Memoria Flash de 32 KB Diseño estático, de bajo voltaje y baja potencia.
- Memoria RAM de 512 bytes. Diseño estático, de bajo voltaje y baja potencia

3.1.1.2. MÓDULO DE INTEGRACIÓN DEL SISTEMA (SIM08)

Este módulo tiene varias funciones, todas estrechamente ligadas a la CPU08, incluyendo la generación del reloj del bus y su control (modos *stop y wait*); control del reset (incluyendo el *reset* de encendido y el del *watchdog*) y control de interrupciones.

3.1.1.3. GENERADOR DE RELOJ (CGM08)

El módulo generador de reloj genera dos señales de reloj importantes para los dispositivos HC08: una señal de reloj de cristal (CGMXCLK) para el COP (perro guardián) y el generador de baudios del módulo de comunicaciones serie (SCI) y una señal dividida por dos (CGMOUT) para el reloj de todos los otros sistemas internos. El módulo CGM08 incluye un circuito oscilador a cristal, un PLL con frecuencia de salida en múltiplos enteros de la referencia de cristal externa, y un circuito selector para la señal de reloj (CGMOUT).

3.1.1.4. MÓDULO DE PROTECCIÓN DEL SISTEMA

En este modulo se detectan las instrucciones ilegales: Cada instrucción es comparada con el mapa de códigos de operación. Si se determina que la instrucción no existe, se la considera ilegal y se produce un reset.

Detección de direcciones ilegales: Si se detecta que se está buscando una instrucción desde una posición de memoria no implementada o perteneciente a un registro interno, se produce un *reset*.
Detección de bajo voltaje: El módulo LVI (*Low Voltage Inhibit*) monitorea el valor de la tensión de alimentación. Cuando ésta cae por debajo de un valor establecido, se genera una interrupción.

COP (*Computer Operating Properly*)**:** Es un *watchdog*, diseñado para detectar condiciones de error de software. Cuando está habilitado, el software del usuario es responsable de resetearlo regularmente. Si esto no sucede, el COP produce el reset de la CPU, llevándola a un estado conocido.

3.1.2. MEMORIA DE PROGRAMA

3.1.2.1. MEMORIA FLASH INTERNA

Todos los microcontroladores de la familiar HC08 cuentan con memoria Flash para almacenar el programa. Esta memoria se puede borrar y volver a grabar. No requiere de dispositivos ni valores de tensión especiales. Existe un procedimiento, denominado ISP (*In-system programming*) que permite grabar la memoria flash con el microcontrolador soldado en la placa, en su lugar definitivo.

3.1.2.2. MEMORIA RAM INTERNA

La memoria RAM está presente en todos los microcontroladores y sirve para almacenar datos de manera volátil. El tamaño varía entre 128 y 4096 bytes.

3.1.3. PINES DE ENTRADA Y SALIDA DE PROPÓSITO GENERAL

La cantidad de pines disponibles para entrada y salida es de 40 pines. En general estos pines están compartidos con las entradas o salidas de otros módulos (Timer, A/D, etc), por lo que la cantidad final disponible depende del uso que se haga de ellos. Algunos tienen la capacidad de manejar corrientes elevadas, para controlar de manera directa *leds* o relés.

3.1.4. MODULO DE CONVERSION A/D INTERNO

Todos los microcontroladores de la familia HC08 incluyen un conversor A/D. Existen dos módulos diferentes, uno de 8 bits con una cantidad variable de canales. Algunas características comunes son:

- Utilizan el método de Aproximaciones sucesivas lineales.
- Resolución de 8 o 10 bits.
- Conversión simple (una sola conversión) o continua.
- La finalización de la conversión se señala con un bandera o una interrupción.
- Reloj seleccionable.
- Tiempo de conversión de 17 microsegundos con un reloj del A/D de 1 MHz
- Módulo extensible a cualquier cantidad de entradas analógicas, utilizando un multiplexor.

3.1.5. MODULO DE TEMPORIZACIÓN

3.1.5.1. MÓDULO DE TIMER (TIM08)

Este módulo, que puede tener 2 canales independientes, es muy flexible y muy útil para resolver una variedad de aplicaciones. Entre sus características principales se destacan:

- Cada canal puede programarse de manera independiente para funcionar en el modo *Output Compare*, *Input Capture* o PWM sin *búfer* .
- Un par de canales puede combinarse para generar un PWM con *buffer.*
- Un preescalador con divisor programable, incluyendo reloj externo.
- Contador de 16 bits, *free-running* o con módulo programable.
- Un *overflow* puede ser usado para cambiar el estado de cualquier pin del *timer.*
- La captura de entrada se puede disparar con flancos de subida o bajada.
- El comparador de salida puede cambiar el pin de salida.
- Cada interrupción tiene su propio vector.

3.1.5.2. MÓDULO DE BASE DE TIEMPO

El TBM (*Time Base Module*) sirve para generar interrupciones periódicas. La frecuencia de estas interrupciones está determinada por el valor del registro del TBM y el oscilador. Usando un cristal de 32.768 KHz. En conjunto con el PLL se puede trabajar a 32 Mhz.

3.1.5.3. PIT (TIMER DE INTERRUPCIONES PROGRAMABLE)

Es un timer de 16 bits que puede utilizarse para la generación de interrupciones periódicas.

3.1.5.4 PWMMC (MODULADOR DE ANCHO DE PULSO)

Este módulo provee tres pares de PWM complementarios o seis señales de PWM independientes.

3.1.6 MODULOS DE COMUNICACIONES

Existen distintos módulos para distintos tipos de comunicaciones

3.1.6.1 SPI (INTERFACE DE PERIFÉRICOS SERIALES)

El módulo SPI es utilizado para comunicaciones síncronas a lo largo de pequeñas distancias (generalmente dentro de la misma placa) a altas velocidades. El SPI permite al microcontrolador comunicarse con dispositivos periféricos tales como memorias *EEPROM*, conversores A/D, expansores de I/O, etc.

Algunas de las características mas importantes de este módulo son:

- Comunicación *full-dúplex*.
- Transferencias síncronas usando 3 señales.
- Operación en modo maestro o esclavo
- La velocidad máxima en modo maestro es la frecuencia del bus dividida 2.
- La velocidad máxima en modo esclavo es la frecuencia del bus.
- Cuatro velocidades de maestro seleccionables.
- Polaridad y fase de reloj seleccionables.

3.1.6.2 MODULO SCI (SERIAL COMMUNICATIONS INTERFACE)

El módulo SCI permite realizar comunicaciones asíncronas entre el microcontrolador y un puerto RS232, dispositivos con salida serie u otros microcontroladores en una red. Un generador de baudios interno es capaz de derivar las velocidades de comunicación estándar del oscilador del microcontrolador.

Algunas de las especificaciones del modulo SCI son:

- *UART* incorporada.
- Formato NRZ estándar.
- Operación *full-dúplex*.
- Doble *buffer,* tanto en el transmisor como en el receptor.
- Longitud de caracteres programable: 8 o 9 bits.
- Generador de baudios.
- Posibilidad de funcionar en base a interrupciones o por encuesta.
- Generación y comprobación de paridad.
- Soporte para operación en bajo consumo de corriente.

3.1.7 MÓDULO DE EXPLORACIÓN DE TECLADO

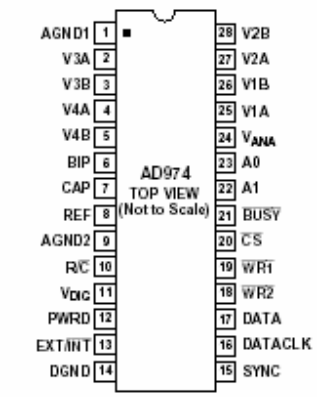
El módulo KBI (*Keyboard Interrupt Module*) está específicamente diseñado para ayudar al programador a detectar la pulsación de botones o teclas. Esto es muy útil sobre todo en aplicaciones de bajo consumo, alimentadas con una batería, donde es deseable mantener al microcontrolador en un estado de bajo consumo mientras espera la pulsación de una tecla. Cuando la tecla es pulsada, se genera una interrupción que saca al microcontrolador de esa condición y ejecuta el código de lectura de teclas. El KBI además añade resistencias de *pull-up* en las entradas de teclas, lo que disminuye el espacio necesario en la placa y el costo del sistema.

3.2 CONVERSOR A/D AD974

3.2.1 DESCRIPCIÓN GENERAL

El conversor analógico digital AD974 es un sistema de adquisición de datos de 4 canales, con una interfaz serial SPI. Este dispositivo contiene un multiplexor de entrada, y puede muestrear señales analógicas de [+10 V,-10 V] con una resolución de 16 bit y una frecuencia máxima de 200 Khz. Todo el dispositivo opera con una fuente de 5 V DC y posee un modo de bajo consumo de corriente cuando no esta operando. El conversor se puede configurar para trabajar en un rango de señales analógicas de entrada de rango [0 a 4 V], [0 a 5 V], o [+10,-10 V].

Figura 5. Configuración de Pines y Encapsulado Convesor AD974

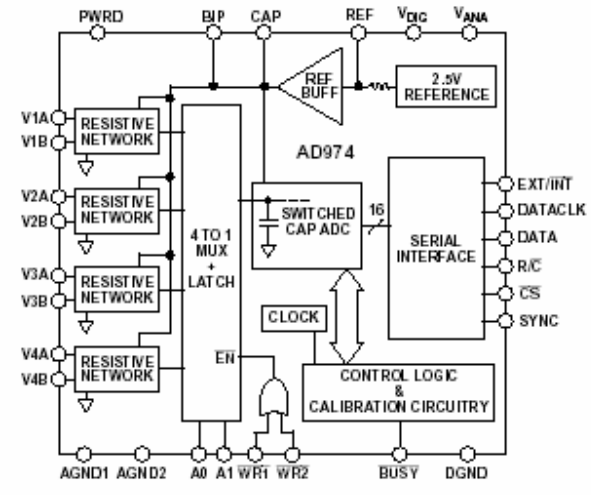


Fuente*: Datasheet AD974 Analog Devices*

Este conversor esta diseñado para una transferencia eficiente de datos, requiriendo un número bajo de interconexiones externas. El AD974 es fabricado por *ANALOG DEVICES* mediante un proceso BICMOS, el cual posee un alto desempeño utilizando dispositivos bipolares y transistores

CMOS. El AD974 esta disponible en encapsulado de 28 pines DIP, SOIC y encapsulado SSOP.

Figura 6. Diagrama de Bloques Funcional AD974



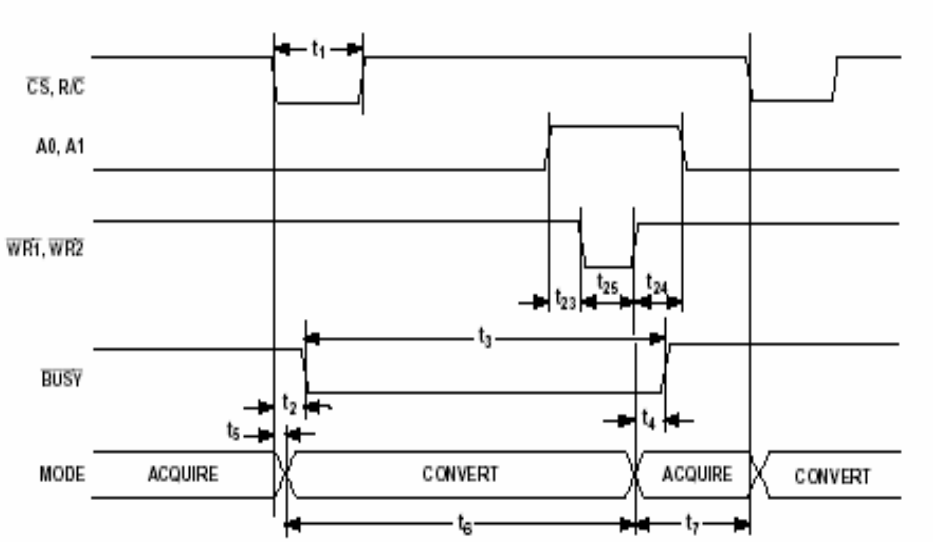
Fuente: *Analog Devices*

3.2.2 CONTROL DE LA CONVERSIÓN EN EL CONVERSOR AD974

El AD974 esta controlado por dos señales: R/C y CS. Cuando la señal R/C cae a un nivel bajo, con CS en nivel bajo, por un mínimo de 50 ns, la señal de entrada podría permanecer en el arreglo de capacitores internos y una conversión (n) será iniciada. Una vez el proceso de conversión es iniciado, la señal *Busy*- estará en nivel bajo hasta que la conversión este completa. Internamente las señales R/C y CS requieren un retardo de 10 ns cuando son llevadas a nivel bajo. Después de que la conversión esta completa, la señal *Busy* retornará al nivel alto y el AD974 estará listo para capturar una señal de entrada. Bajo ciertas condiciones el pin CS puede ser llevado a nivel bajo y la señal R/C podría ser usada para determinar si se está iniciando una conversión o se están leyendo datos. En la primera conversión, después que el AD974 ha sido inicializado los datos de salida pueden ser indeterminados.

La conversión resultante puede ser llevada a cabo serialmente usando un generador de reloj interno realizado por el AD974 o por medio de un reloj externo.

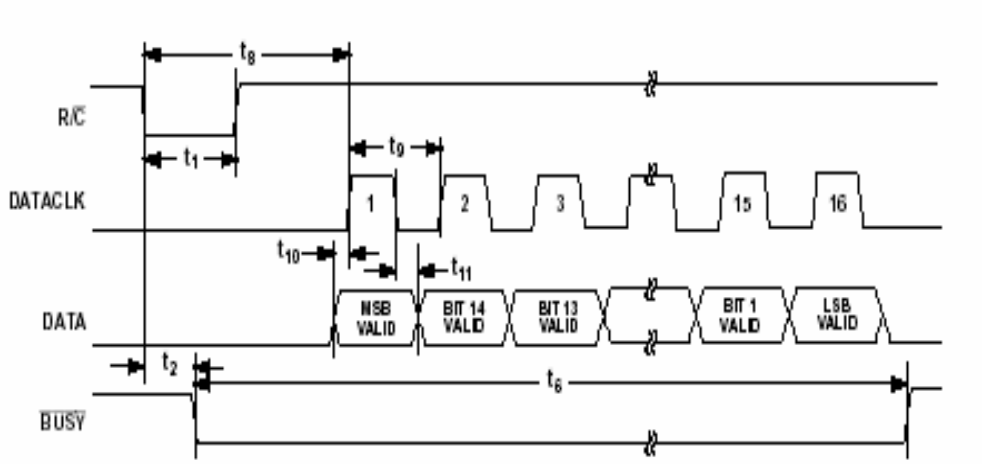
Figura 7. Tiempo de Conversión Básico AD974



Fuente: *Analog Devices*

Para configurar el reloj de datos interno se debe colocar el pin EXT/INT a nivel bajo. Para configurar el modo de reloj externo se debe colocar EXT/INT a nivel alto.

Figura 8. Temporización de datos seriales del conversor AD974



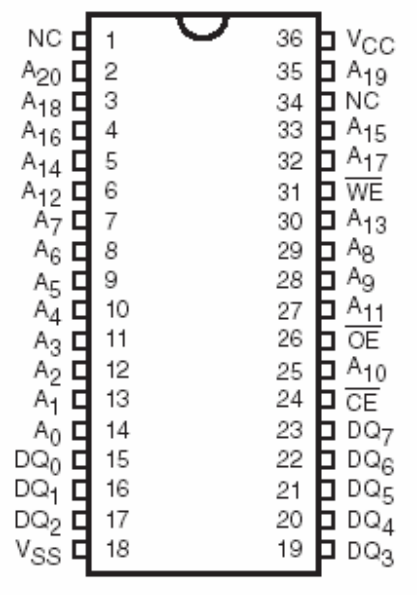
Fuente: *Analog Devices*

3.3 MEMORIA S-RAM BQ4017

La memoria CMOS BQ4017, es una memoria RAM estática de 16.777.216 bits organizada en 2.097.152 palabras por 8 bits. El circuito de control integral y la batería de litio interna permite retener los datos como una memoria *flash* no-volátil con ciclos de escritura ilimitados de una memoria *SRAM* estándar.

El circuito de control monitorea constantemente una fuente de 5V DC y cuando el voltaje Vcc cae por debajo de un nivel de tolerancia la memoria SRAM es condicionalmente protegida contra escritura previniendo la pérdida de datos. La fuente integral de energía es conmutada a la batería de litio hasta que el nivel de la fuente retorna al de Vcc. La memoria BQ4017 usa una corriente de *stand-by* extremadamente baja acoplada con una pequeña celda de litio permitiendo la no-volatilidad durante un largo ciclo de escritura y con las limitaciones asociadas a la *EEPROM*

Figura 9. Pines de conexión memoria BQ4017



Fuente: *Texas Instruments*

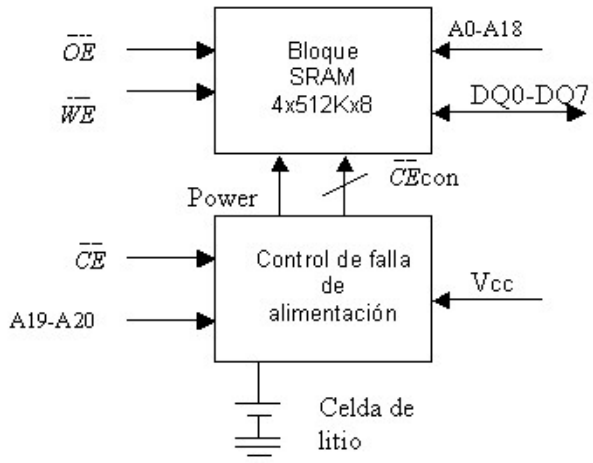
3.3.1 DESCRIPCIÓN FUNCIONAL

Cuando la fuente de poder es válida la memoria BQ4017 opera como una memoria CMOS S-RAM estándar. Durante una caída de tensión la memoria actúa como una memoria no-volátil protegiendo y preservando automáticamente el contenido de los datos.

El circuito de control monitorea constantemente el voltaje de alimentación detectando una caída de tensión de 5 V DC a 4.62 V DC con una tolerancia del 5% de la fuente de alimentación. Cuando el voltaje de alimentación cae de este nivel la memoria SRAM protege los datos escritos automáticamente. Todas las salidas se van a alta impedancia y todas las entradas son descartadas. Si un acceso válido está en proceso al tiempo que se detecta

la caída de tensión de la fuente, el ciclo de memoria continua hasta completarse.

Figura 10. Diagrama de Bloques memoria BQ4017



Fuente: *Texas Instruments*

La caída de tensión de la fuente de alimentación permite al circuito de control de la memoria conmutarse a los 3 V de la batería de litio de respaldo permitiendo la retención de los datos hasta que la fuente de alimentación sea restablecida.

La batería de litio puede retener los datos hasta 5 años en ausencia de la fuente de alimentación.

Tabla 7. Tabla de Verdad memoria BQ4017

Modo	$\overline{CE}$	$\overline{WE}$	$\overline{OE}$	**Operación I/O**	**Estado**
No seleccionado	H	X	X	Alta impedancia	Standby
Salida deshabilitada	L	H	H	Alta impedancia	Activo
Lectura	L	H	L	Salida datos	Activo
Escritura	L	L	X	Entrada datos	Activo

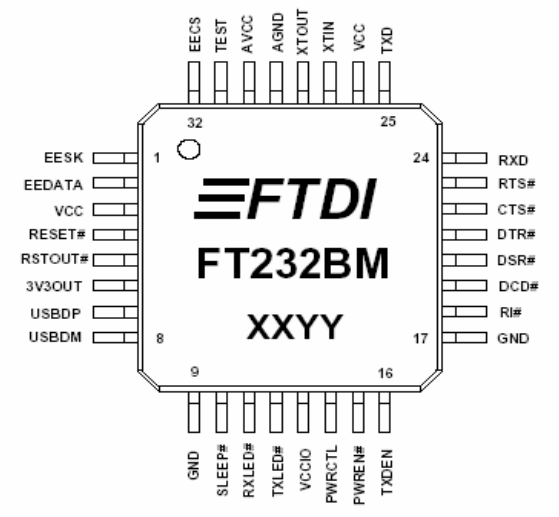
Fuente: *Texas Instruments*

Cuando el voltaje de alimentación retorna a 5 V el circuito de protección de escritura continua por 120ms máximo para permitir a la memoria la estabilización, después de este tiempo la memoria entra en operación normal.

3.4 CONTROLADOR USB 2.0 FT232BM

El controlador USB 2.0 FT232BM es la segunda generación del controlador USB UART de *Future Technology devices*

Figura 11. Diagrama de pines FT232BM



Fuente: *Future Technology devices FTDI*

Este dispositivo adiciona funcionalidades extra a su predecesor el controlador FT8U232AM reduciendo la cantidad de componentes externos y manteniendo un alto grado de compatibilidad con su predecesor permitiendo una fácil actualización del controlador USB y reduciendo los costos de diseño e incrementado el potencial de uso de éste dispositivo en nuevas áreas de aplicación.

3.4.1 CARACTERÍSTICAS DEL CONTROLADOR USB 2.0 FT232BM.

- Transferencia de datos seriales asíncronos mediante el bus USB 2.0 en un solo circuito integrado.
- Posee señales para realizar una interfaz de *modem* estándar con señales de control y reconocimiento.

El dispositivo soporta envio de datos de 7 y 8 *bit*, 1 o 2 bit de parada, paridad par e impar o no paridad.

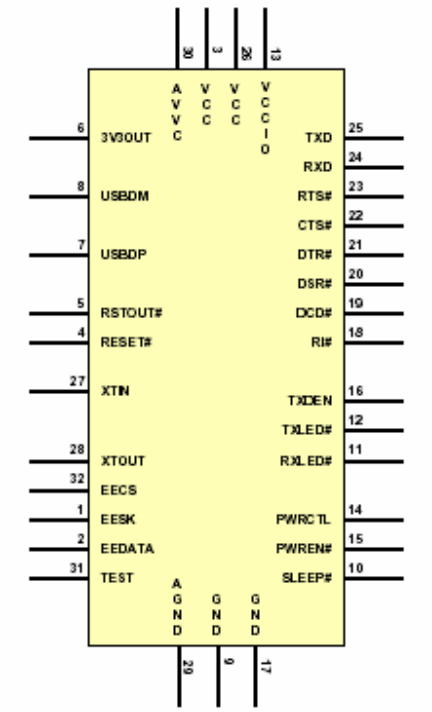
Figura 12. Encapsulado controlador USB 2.0 FT232BM

Fuente: *Future Technology devices*

- Descarga de datos a 3 Mb mediante interfaz USB-TTL.
- Descarga de datos a 1 Mb mediante interfaz USB-RS232
- Descarga de datos a 3 Mb mediante interfaz USB-RS422/RS485
- Posee un buffer de recepción de 384 *bytes* y un *buffer* de transmisión de 128 bytes para alta transferencia de datos.
- El FT232 tiene un *buffer* ajustable de recepción dependiente del tiempo.
- Posee asistencia de control de hardware en modo *x-on / x-off*.
- Posee *buffer* automático de transmisión para control por RS-485.
- Conversor de nivel integrado para control de señales de 5 V y lógica de 3.3 V.

- Posee un regulador integrado de 3.3 V para dispositivos USB.
- Tiene un circuito de *power on* reset integrado.
- Posee un PLL integrado que multiplica la señal del cristal externo de 6 MHz a 48 MHz.
- Posee modos de transferencia de datos isócrono y *por volumen*.
- Opera con una fuente de voltaje de 4.35 V a 5.25 V.

Figura 13. Esquema FT232BM



Fuente: *Future Technology devices FTDI*

- Es compatible con el estándar USB 1.1. y USB 2.0.
- Posee un identificador de vendedor e identificador de producto, y puede ser actualizado mediante una memoria serial externas EEPROM.

- La memoria EEPROM externa puede ser programable por medio del BUS USB.
- Encapsulado LQFP de 32 pines
- Posee un *driver* de fabricante llamado driver de puerto com virtual (PCV) compatible con los sistemas operativos *Windows* 98, 2000,ME, XP, CE, MAC OS8 y MAC OS9, MAC OSX, Linux 2.4 o superior.
- Posee driver USB D2XX para programar con el kit de desarrollo de *driver* de *Microsoft* compatible con los sistemas operativos *Windows* 98, 2000,ME, XP.
- Las áreas de aplicación de este controlador USB son:

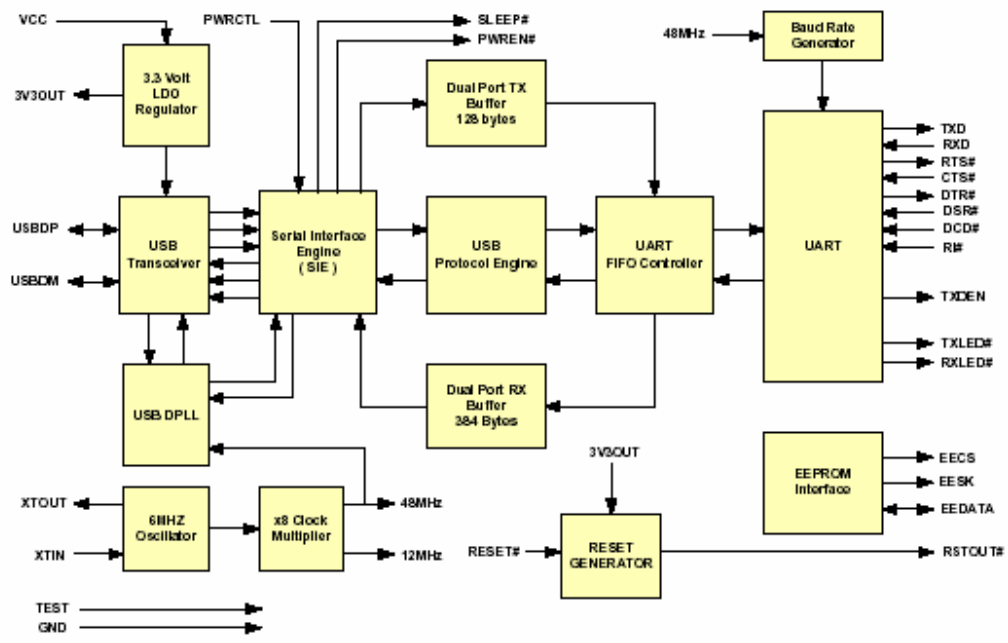
    - Conversores USB RS-232
    - Conversores USB RS422/RS485
    - Actualización de puertos serie a periféricos USB
    - Telefonía celular y telefonía inalámbrica para transferencia de datos USB
    - Interfaz con microcontroladores para diseños basados en BUS USB
    - Transferencia de datos, video y audio de mediano ancho de banda
    - Transferencia de datos PDA a USB
    - Diseño de lectores de tarjetas inteligentes USB
    - Diseño de modems USB
    - Diseños de *modem* tradicionales y *modem* inalámbricos USB
    - Diseños de sistemas de adquisición de datos para instrumentación y control
    - Diseño de lectores de códigos de barras USB.

3.4.2 DESCRIPCION DE LOS BLOQUES FUNCIONALES DEL CONTROLADOR USB 2.0 FT232BM

3.4.2.1 REGULADOR DE 3.3. V LDO

El regulador LDO de 3.3 V genera un voltaje de 3.3 V para alimentar el transmisor- receptor (*transceiver)* USB y requiere de un capacitor externo de desacople en el pin de salida también se puede alimentar a cualquier dispositivo externo que se alimente a 3.3 V y consuma menos de 5 mA.

Figura 14. Diagrama de bloques simplificado controlador FT232BM



Fuente: *Future Technology devices FTDI*

3.4.2.2 TRANSMISOR / RECEPTOR USB

La celda de transmisión - recepción USB provee la interfaz física USB 1.1/ USB 2.0 a 12 Mb por segundo. La salida es alimentada con 3.3 V, controla la señalización con el receptor de datos diferencial permitiendo la entrada y salida de datos mediante los pines USBDM y USBDP.

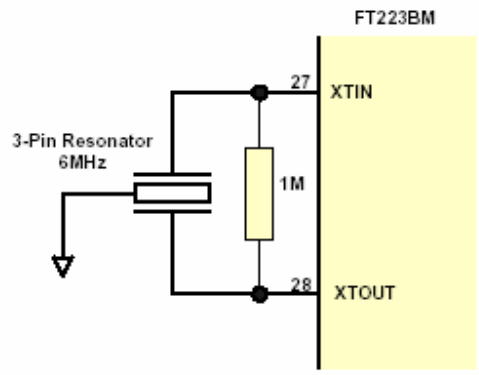
3.4.2.3 USB DPLL

La celda USB DPLL asegura los datos de entrada USB en el formato *NRZI* y permite separar el bloque de datos, de la señalización al bloque *SIE.*

3.4.2.4 OSCILADOR DE 6 Mhz

La celda del oscilador de 6 Mhz permite una entrada de reloj que es multiplicada por ocho (8) del cristal de cuarzo externo colocado en los pines XTOUT y XTIN.

Figura 15. Configuración Del Cristal para el controlador USB FT232BM



Fuente: *Future Technology devices FTDI*

3.4.2.5 MULTIPLICADOR DE RELOJ

El multiplicador de reloj toma una señal de 6Mhz del oscilador de entrada y genera un reloj de referencia de 12 Mhz para el motor de protocolo USB y el bloque controlador de la *UART* FIFO. Este módulo además genera un reloj de referencia de 48 Mhz para el bloque generador de frecuencia de baudios y el USB DPPL .

3.4.2.6 MOTOR DE INTERFAZ SERIAL (SIE).

El bloque de motor de interfaz serial SIE desarrolla la conversión paralela a serie y serie a paralela de los datos USB. De acuerdo a la especificaciones USB 2.0 y genera un chequeo de redundancia cíclica sobre la trama de datos USB.

3.4.2.7 MOTOR DE PROTOCOLO USB

El motor de protocolo USB maneja la trama de datos del dispositivo USB mediante el *endpoint* de control. Los requerimientos generados por *host* controlador USB y los comandos para controlar los parámetros funcionales de la *UART*.

3.4.2.8 BUFFER DE TRANSMISIÓN DUAL(128 BYTES)

Los datos del endpoint de salida USB son almacenados en un buffer dual de puerto TX y removidos del buffer al registro de transmisión de la *UART*. Bajo el control del controlador *FIFO* de la *UART*.

3.4.2.9 BUFFER DE RECEPCIÓN DUAL RX (384 BYTES)

Los datos del registro de recepción de la UART son almacenados en el *buffer* dual RX con prioridad a ser removidos por el *SIE* en el requerimiento de datos USB del *endpoint IN* de datos del dispositivo.

3.4.2.10 CONTROLADOR FIFO UART

El controlador *FIFO UART* permite la transferencia de los datos entre los buffer duales de puerto RX y TX y los registros de transmisión y recepción de la *UART*.

3.4.2.11 LA UART

La *UART* desarrolla la conversión asíncrona 7/8 bit paralela a serie y serie a paralela de los datos en la interfaz RS232 (RS422 y RS485). Las señales de control soportadas por la UART incluyen las señales de RTS, CTS, DSR, DTR, DCD, y RI. La *UART* permite habilitar la transmisión de las señales de control (TXDEN) para interfasar los transceiver RS485. La *UART* soporta las señales de reconocimiento RTS/CTS, DSR/DTR y X-ON/X-OFF.

3.4.2.12 GENERADOR DE FRECUENCIA DE BAUDIOS

El generador de frecuencia de baudios permite la multiplicación de la señal de reloj por 16 a la entrada de la *UART*, permitiendo una señal de reloj de referencia de 48 Mhz y consiste de un pre-escalador de 14 bit y tres registros los cuales proveen la sintonía fina de la frecuencia de baudios. La frecuencia de baudios de la *UART* puede ser programada desde 183 baudios a 3 millones de baudios.

3.4.2.13 GENERADOR DE RESET

La celda de generación de reset permite realizar un *reset* al iniciar la alimentación del dispositivo. Una entrada adicional de *reset* y una salida de *reset* son implementadas para permitir resetear otros dispositivos FT232BM. Durante el reset el pin *RSTOUT* es llevado a cero. En operación normal este saca 3.3 V por medio del regulador interno y el pin *RSTOUT* puede ser controlado por medio de una resistencia de *pull-up* sobre el pin USBDP directamente permitiendo un tiempo para la enumeración del dispositivo USB.

3.4.2.14 INTERFAZ EEPROM

Aunque el controlador FT232BM podría trabajar sin la memoria serial EEPROM SPI , una memoria externa como la EEPROM 93C46(93C53 o 93C66) puede ser usada para personalizar el identificador de vendedor (VID) PID, número serial, descriptores del producto, descriptor de alimentación para aplicaciones OEM. Otros parámetros controlados por la EEPROM incluyen manejo remoto, modo de transferencia isócrona y descriptores para modos USB 2.0. La EEPROM debe configurar tramas de 16 bits tal como la memoria EEPROM 93LC46B o equivalente con una capacidad de 1 MB a 5 V. La EEPROM se puede programar en la *board* directamente usando un software disponible directamente en la página web de FTDI[4]. Si la memoria EEPROM no es utilizada en la aplicación el controlador USB FT232BM usará los descriptores del producto por defecto VIP, y el descriptor del valor de alimentación. En este caso el dispositivo no tendrá número serial como parte de sus descriptores USB.

[4] http:// www. Ftdichip.com

4. HARDWARE DE LA TARJETA ADQ USB 2.0

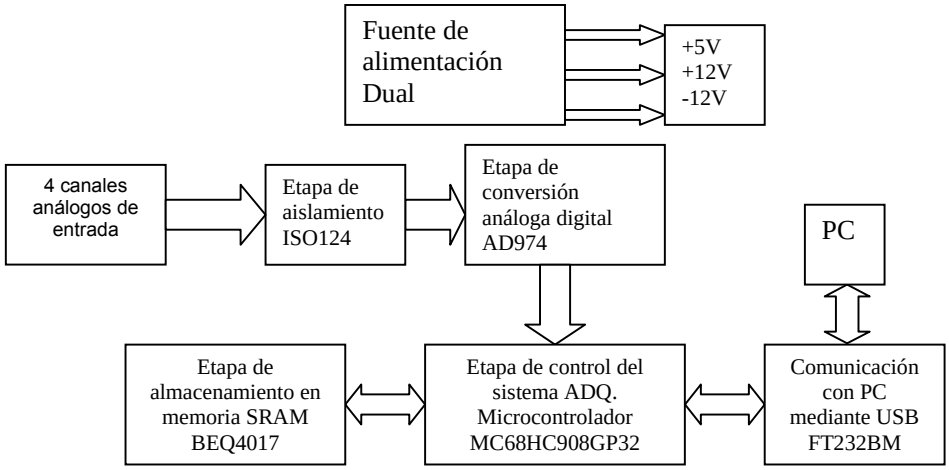
A continuación se describen las diferentes etapas de *hardware* que componen la tarjeta de adquisición de datos y en la cual se utiliza como elemento central de control el microcontrolador MC68HC908GP32, de la familia HC08 de *Freescale* (Motorola). Se describen en este capitulo las etapas de aislamiento, conversión análoga digital, almacenamiento, control y comunicación utilizando el bus USB 2.0 además se presentan los esquemáticos que describen la implementación física de cada una de las diferentes etapas.

Para el diseño de la tarjeta se tuvieron en cuenta los requerimientos de la aplicación, en la cual originalmente se busca utilizarla en un sistema de adquisición de datos de cuatro canales provenientes de sensores montados sobre un motor diesel, que ofrezca una interfaz de comunicación con el PC flexible y de fácil manejo para el usuario, y una aplicación para visualizar las señales adquiridas ya sea en el modo de tarjeta de adquisición de datos o en el modo de *data logger*[8].

En la figura 16 se puede ver un diagrama de bloques del sistema de adquisición de datos.

[8] En este segundo modo se utiliza la memoria como herramienta de almacenamiento para adquirir la señal sin la necesidad de estar conectado al PC, lo que hace que el sistema sea más flexible.

Figura 16. Diagrama de bloques general del sistema de adquisición de datos



Fuente: Los autores.

4.1 ETAPA DE AISLAMIENTO

Esta etapa se diseñó con el fin de aislar y brindar protección a los elementos de la tarjeta de adquisición de datos en el caso de que se presente un sobrevoltaje en cualquiera de los canales, de la misma manera, es deseable mantener los sensores lejos de posibles sobrevoltajes que se pudieran generar en el sistema al que provee información debido al ambiente potencialmente inflamable del motor de combustión interna. Por esta razón es necesario brindar un aislamiento galvánico en ambos sentidos.

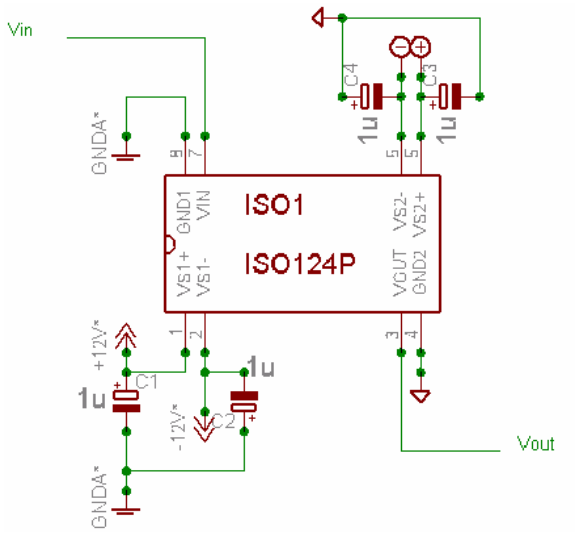
El dispositivo utilizado para realizar el aislamiento es el ISO124, que brinda protección hasta una entrada de 2400Vrms con un ancho de banda de 50KHz.

El aislamiento se logra modulando la señal de entrada con una portadora de 500KHz que luego es transmitida digitalmente a través de la barrera galvánica y finalmente reconstruida en la etapa de salida del aislador.

En la práctica este dispositivo trabaja con ganancia unitaria y con alimentación dual de +12V y – 12V.

A continuación en la Figura 17 se puede ver el diagrama de conexión que se utilizó en el diseño de la tarjeta. La misma configuración se utilizó para cada uno de los cuatro canales de entrada.

Figura 17. Esquemático etapa de aislamiento para un canal.



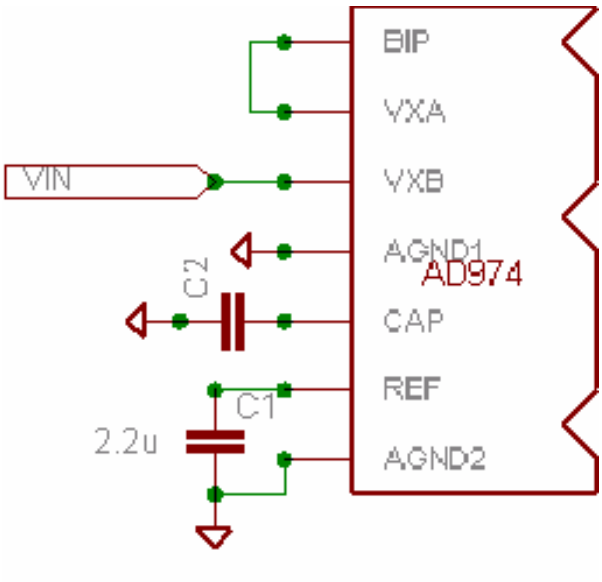
Fuente: *Datasheet* ISO124 *Analog Devices*.

4.2 ETAPA DE CONVERSION ANALÓGICA - DIGITAL

Como se mencionó anteriormente, el conversor A/D utilizado fue el AD974 fabricado por *Analog Devices*, que posee una resolución de 16 bits, conversión por aproximaciones sucesivas, cuatro canales y una frecuencia de muestreo de 200Ks/s (muestras por segundo).

La señal de entrada para los diferentes canales analógicos de entrada que proviene de la etapa de aislamiento, esta en el rango de -10 V a +10 V para permitir la configuración bipolar del conversor A/D. Para implementar esta configuración es necesario conectar las entradas VxA de cada canal del conversor A/D con el pin BIP, y la señal de entrada analógica se proporcionará en el pin VxB de cada canal correspondiente en el conversor A/D. La figura 18 muestra la configuración bipolar del converso AD974.

Figura 18 Conexión bipolar del conversor AD974.



Fuente: *Datasheet* AD974 *Analog Devices.*

Adicionalmente se conecta un capacitor de tantalio de 2.2μF a tierra en el pin CAP que permite acoplar la referencia de salida del buffer del conversor AD974, también se conecta otro capacitor de tantalio de 2.2uF en el pin REF del conversor A/D para obtener la referencia interna de 2.5V que requiere el dispositivo para llevar a cabo la conversión A/D mediante aproximaciones sucesivas de la red resistiva de cada canal analógico de entrada.

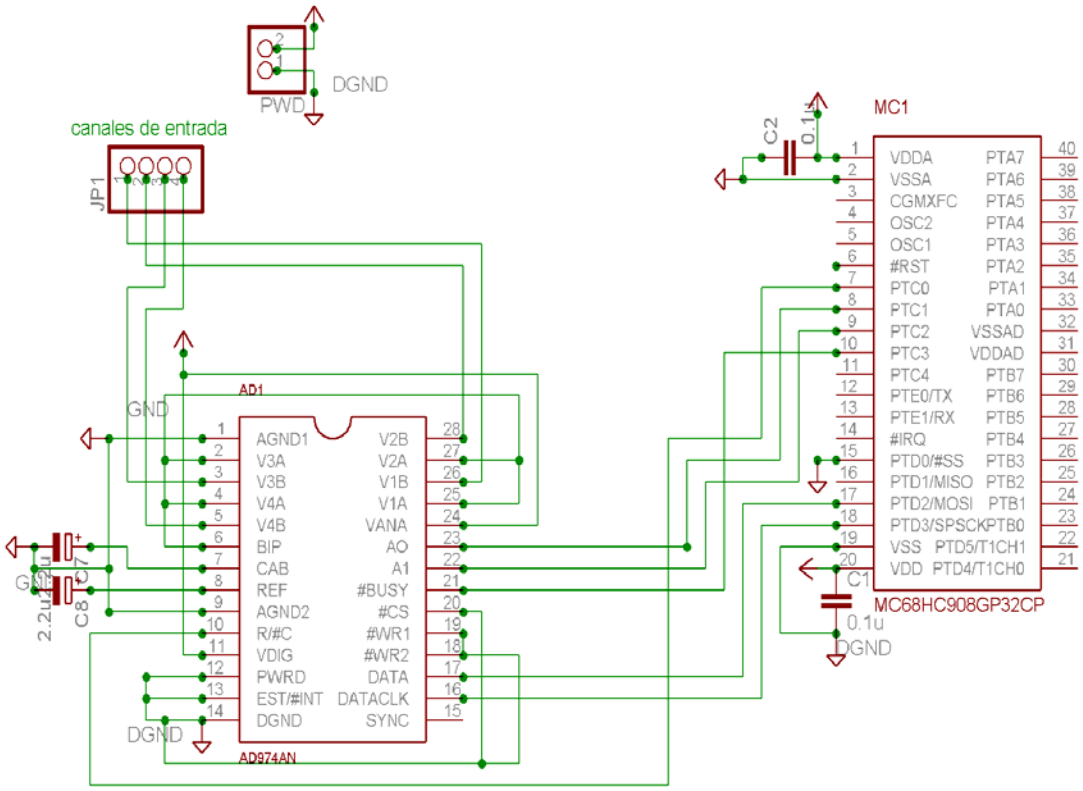
El reloj que controla los datos seriales de salida del conversor se obtiene desde el reloj interno del mismo conversor A/D, por lo tanto el pin (#13), $EXT / \overline{INT}$ se conecta a tierra, configurando de esta manera al AD974 como maestro de la comunicación serial síncrona (SPI).

En la comunicación SPI la conversión de la señal analógica esta controlada por la señal $R/\overline{C}$ [pin 10] . Esta señal permite el inicio de un ciclo de conversión, mientras ocurre la conversión el pin (#21), $\overline{BUSY}$ indica el estado

de la conversión, por lo tanto, ésta señal es la que espera la etapa de control para solicitar un nuevo ciclo de inicio de conversión A/D.

En la Figura 19 se muestra el esquemático de conexión del conversor AD974 y el microcontrolador para la comunicación SPI.

Figura 19. Diagrama de conexión de la etapa de conversión A/D



Fuente: Los autores

En el esquemático de la figura 19 se puede observar el conversor A/D y el microcontrolador encargado de controlar el ciclo de conversión.

La transmisión de datos entre el conversor A/D y el microcontrolador se realiza utilizando una comunicación serial síncrona (SPI), en la cual es necesario contar con un reloj que sincronice la transmisión y recepción de cada bit de los datos que salen serialmente del conversor análogo digital, éste reloj se obtiene del oscilador interno del AD974, el cual posee un

periodo de 220ns. Por cada ciclo de reloj que se obtiene en el pin DATACLK, pin (#16), del conversor análogo digital, en el pin DATA, pin (#17), esta disponible uno de los 16 bits del resultado de la conversión, teniendo en cuenta que el primer bit disponible es el bit más significativo (MSB).

El microcontrolador se encarga de recibir el dato transmitido por el conversor A/D utilizando la interfaz de comunicaciones SPI. Esta interfaz posee tres registros de interés: el primero es el registro de control, en donde se puede configurar el modo de trabajo (maestro o esclavo), que en este caso se configura el modo esclavo, debido a que el reloj de la comunicación proviene del conversor A/D, así mismo, se pueden configurar interrupciones, polaridad, fase y velocidad del reloj de comunicación (en caso de configurarse como maestro). El segundo registro es el de estado, en donde están las banderas que indican situaciones como receptor lleno o transmisor vació y el tercer registro es el registro de datos en donde se pueden leer o escribir los datos a transmitir[9].

Además de estos tres registros el modulo SPI del microcontrolador posee un registro de desplazamiento no direccionable en el que se desplaza bit a bit el dato de entrada mientras simultáneamente se desplaza el dato de salida.

El tamaño estándar de dato que se utiliza en la interfaz SPI del microcontrolador es de ocho bits, sin embargo, si se trabaja coordinadamente con el registro de datos, el registro de desplazamiento y las banderas de estado, se pueden recibir o transmitir datos de hasta dieciséis bits, que es el tamaño del dato que se recibe desde el conversor A/D, pero se tienen que tener en cuenta factores como la frecuencia de bus del microcontrolador, factor que se explicará a continuación: La máxima frecuencia del reloj de entrada para una interfaz SPI configurada en modo esclavo es igual a la frecuencia de bus del microcontrolador, de manera que si el reloj del

[9] Este registro del microcontrolador, aunque posee una sola dirección, es en realidad un registro doble independiente para transmisión y recepción.

conversor tiene un periodo de 220nS, la frecuencia mínima de bus del microcontrolador debería ser según la ecuación 1:

$$F_{\min Bus} = \frac{1}{220nS} = 4.54545MHz \qquad (1)$$

Si embargo, a esta frecuencia, se presenta en la interfaz SPI un error de desborde, (*overflow*), al no poder leerse el primer byte desde el registro de datos del SPI antes de que el segundo byte del dato se encuentre en el registro de desplazamiento, de manera que el segundo byte se pierde. La solución a este problema es incrementar la frecuencia de bus del microcontrolador, que finalmente se eligió como de 7.9872MHz.

Los pines del módulo SPI del microcontrolador están compartidos con los pines del puerto D del 0 al 3. En la tabla 8 se pueden ver los pines del módulo SPI y su función.

Tabla 8. Pines de la interfaz SPI del microcontrolador GP32

Pin del microcontrolador	Dirección en SPI esclavo	Conexión al conversor	Función
PTD_PTD0/#SS	Entrada	N.C	Selecciona el modo de esclavo en la interfaz SPI.
PTD_PTD1/MISO	Salida	N.C.	Envía datos desde el esclavo de la comunicación.
PTD_PTD2/MOSI	Entrada	DATA	Recibe el resultado de la conversión
PTD_PTD3/SPSCLK	Entrada	DATACLK	Recibe el reloj de sincronización de la transmisión.

Fuente: *Datasheet Microcontrolador MC68HC908GP32 Freescale*

Los pines que se encargan del control de la conversión se pueden ver en la tabla 9.

Tabla 9. Pines de microcontrolador que controlan la conversión A/D.

Pin del microcontrolador	Dirección	Conexión al conversor	Función
PTC_PTC0	Salida	$R/\overline{C}$	Inicia conversión-transmisión de datos
PTC_PTC1	Salida	A0	Junto con A1 selecciona el canal que se va a convertir.
PTC_PTC2	Salida	A1	
PTC_PTC3	Entrada	#Busy	Indica cuando la conversión ha terminado.

Fuente: *Los autores.*

En el tipo de comunicación que se utilizó se pueden observar ventajas respecto a la comunicación paralela de los datos. La principal es el menor uso de pines por parte del microcontrolador, ya que para la comunicación del dato se utilizaron solamente dos pines frente a los dieciséis de una comunicación paralela, pero el precio de éste ahorro de pines es el aumento en la complejidad del código y la disminución de la velocidad de comunicación.

4.3 ETAPA DE ALMACENAMIENTO EN MEMORIA SRAM

Esta etapa se implementó con el fin de tener la posibilidad de adquirir una señal sin la necesidad de estar conectado al computador y realizar la descarga posteriormente para analizarla cuando se conecte nuevamente al

PC, esto hace que el sistema de adquisición de datos pueda ser llevado al lugar de la prueba sin tener que estar conectado al PC, lo que lo hace más cómodo y flexible al momento de realizar un ensayo.

La etapa consta de una memoria SRAM como banco de almacenamiento, dos contadores para generar la dirección de la memoria y el microcontrolador como buffer de datos y como proveedor de las señales de control de lectura y escritura de la memoria SRAM.

El banco de memoria esta constituido por la memoria BQ4017, que es una memoria RAM estática no volátil con capacidad de 16Mb y organizada como 2097152 palabras de 8 bits. Posee una batería interna del litio que actúa cuando se detecta una caída de voltaje en la alimentación para proveer los datos.

La memoria BQ4017 es una memoria paralela tanto para la entrada - salida de los datos, como para su direccionamiento. Por lo tanto posee 8 pines para el dato y 21 pines para la dirección, y para el control de escritura, lectura y habilitación de la memoria cuenta con tres pines.

4.3.1 DIRECCIONAMIENTO DE LA MEMORIA SRAM

Con el fin de evitar direccionar la memoria directamente desde el microcontrolador, lo que significaría utilizar 21 pines de entrada – salida, se plantearon varias posibles soluciones:

- Utilizar un dispositivo lógico programable o CPLD, lo que representaría un incremento de costos y complejidad del sistema.
- Utilizar registros de corrimiento o de entrada serie salida paralela, con la desventaja del incremento en la complejidad y reducción de la velocidad con la que se podría acceder a la memoria.
- Utilizar contadores para obtener la dirección en base a una señal de reloj, con la desventaja de no poder acceder rápidamente a una posición específica de memoria.

Entre estas soluciones se optó finalmente por utilizar contadores, ya que la forma en que se espera acceder a la memoria es de manera secuencial, (una palabra es leída o escrita y enseguida la siguiente posición de memoria es leída o escrita y así sucesivamente), de manera que generar la dirección con contadores es una solución que ocupa pocos pines del microcontrolador y que se ajusta a los requerimientos del sistema.

Los contadores que se utilizaron en el diseño fueron los CD4040, contadores CMOS binarios de 12 bits, que se conectan en cascada para generar las 21 líneas de dirección requeridas.

4.3.2 CONEXIÓN DE LA MEMORIA SRAM CON EL MICROCONTROLADOR

El microcontrolador MC68HC908GP32 se encarga de dar o de recibir el dato que va a ser escrito o leído de la memoria SRAM, así mismo, se encarga de generar la lógica de control de escritura o de lectura para acceder correctamente a la memoria y finalmente, se encarga de generar los pulsos que cambian los estados del contador para generar la dirección de memoria SRAM, además de generar el pulso que lleva a reset a los contadores (estado inicial).

Para escribir o leer el dato hacia o desde la memoria SRAM BEQ4017, se utilizó el puerto B del microcontrolador, para la lógica de control de la memoria se utilizaron los pines que se describen a continuación.

Tabla 10. Pines del microcontrolador conectados a la memoria SRAM.

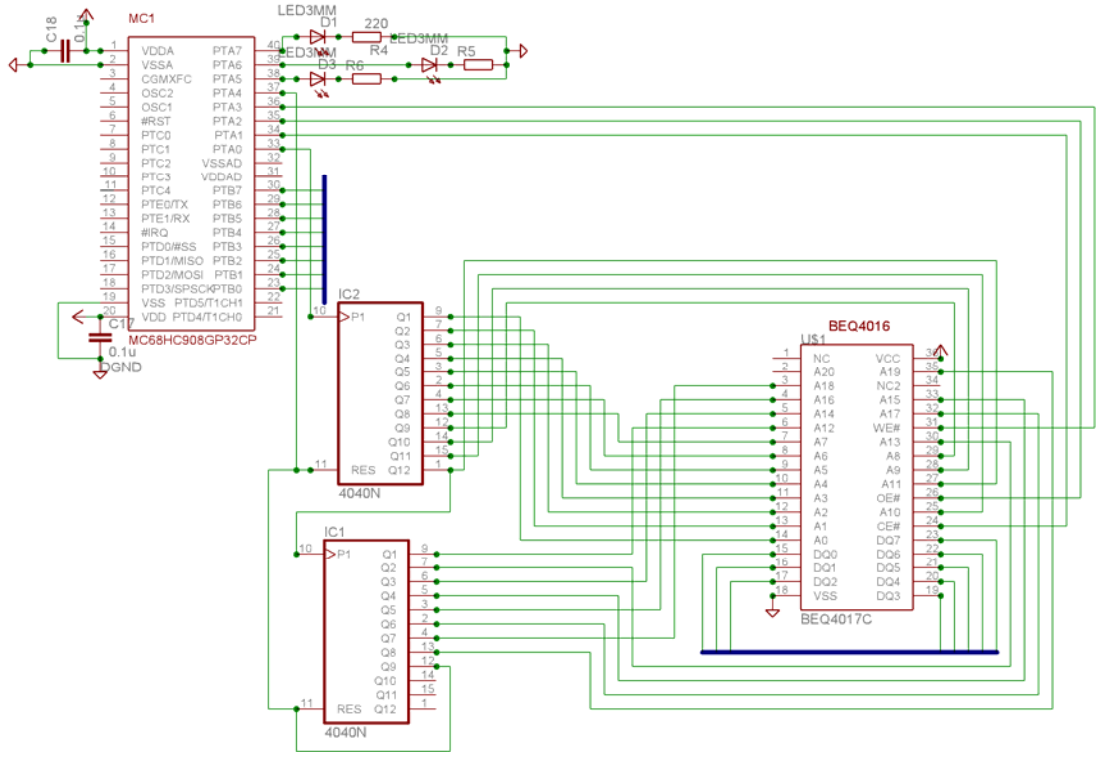
Nombre del pin	Dirección	Función
PTA_PTA7	Salida	Led que indica el estado general del microcontrolador.
PTA_PTA6	Salida	Led que indica la escritura en memoria.
PTA_PTA5	Salida	Led que indica la lectura desde

		memoria.
PTA_PTA4	Salida	Resetea los contadores
PTA_PTA3	Salida	Habilitador de escritura
PTA_PTA2	Salida	Habilitador de salida de memoria(lectura)
PTA_PTA1	Salida	Habilitador de memoria
PTA_PTA0	Salida	Da la señal de reloj que genera la dirección de la memoria

Fuente: Los autores.

A continuación se puede ver el diagrama esquemático de la Figura 20 que muestra la manera en que se conectaron los contadores a la memoria SRAM externa y el microcontrolador, es de destacar que se utilizan únicamente dos pines del microcontrolador para generar la dirección de memoria en lugar de los 21 que se necesitarían sin utilizar los contadores.

Figura 20. Conexión de la memoria SRAM al microcontrolador.



Fuente: Los autores.

4.4 ETAPA DE COMUNICACION USB 2.0

Esta etapa es la que permite la descarga de datos mediante el bus USB 2.0 ya sea desde la memoria SRAM externa o directamente el dato convertido desde el conversor A/D, según las necesidades del usuario, hacia el PC en donde podrán ser visualizados los datos en la aplicación desarrollada en Labview 7.0 Express. Esta etapa utiliza como canal para la comunicación el bus USB, con ventajas tales como facilidad de conexión (*plug and play),* inserción en caliente, y velocidad entre otras.

El controlador USB 2.0 utilizado es el FT232BM de *Future Devices Inc* que de manera general posee un convertidor de datos seriales a USB mediante un motor de control de protocolo USB 2.0 implementado en hardware para trabajar sobre este bus.

En la parte de recepción de datos se puede trabajar con los estándares RS232, RS422 y RS485 ya que posee una interfaz UART para trabajar comunicaciones asíncronas, permitiendo la comunicación serial proveniente desde la interfaz SCI del microcontrolador.
El controlador FT232BM USB 2.0 brinda las siguientes ventajas:

- Modo de transferencia de datos isócrono y por volumen (*bulk).*
- Compatible con el estándar USB 1.1 y 2.0.
- Datos de identificación de producto ID y número serial programables mediante una memoria EEPROM externa.
- La memoria EEPROM puede ser programada desde el USB.
- Disponibles dos tipos de drivers:

1. Drivers VCP: Crean un puerto serial virtual para trabajar sobre aplicaciones que utilicen puerto serie para comunicaciones y programas tales como Labview, Visual Estudio, o C++ Dephi etc.

2. Drivers D2XX: Permiten trabajar con las librerías dinámicas del sistema operativo (DLL) en conjunto con el kit de desarrollo de drivers (DDK) para desarrollar drivers para aplicaciones específicas.

4.4.1 COMUNICACIÓN CON EL MICROCONTROLADOR

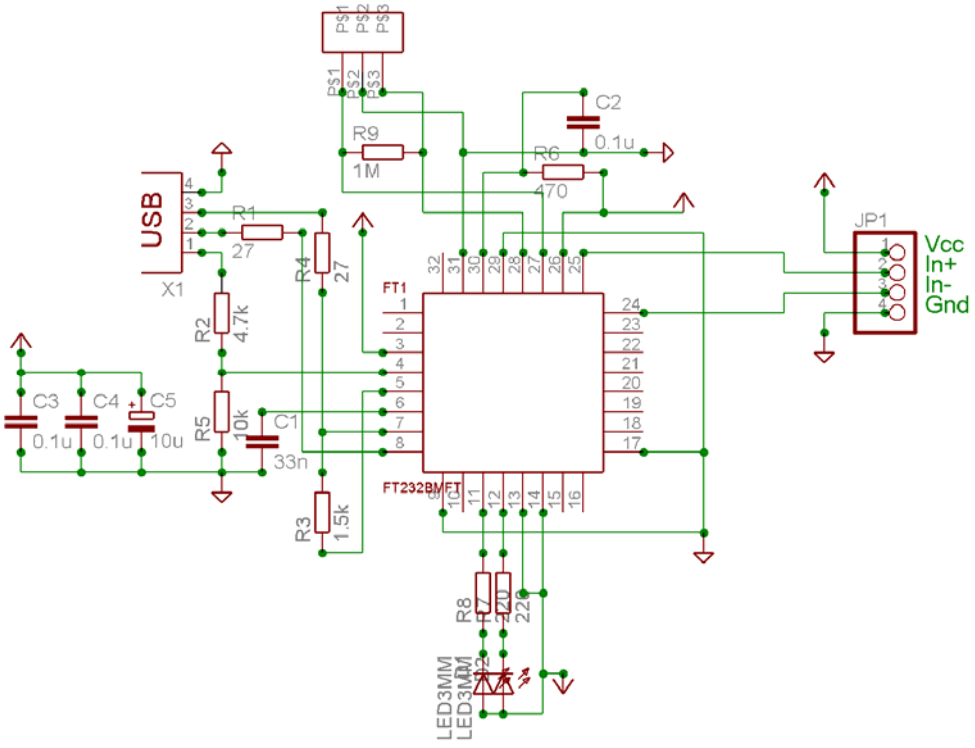
Como se mencionó anteriormente, el controlador de USB 2.0 acepta comunicaciones asíncronas seriales gracias a la interfaz UART que posee, por esta razón, se utiliza interfaz UART del microcontrolador: el módulo de SCI.
La interfaz SCI utiliza dos pines: uno de transmisión y otro de recepción, y la frecuencia de baudios se puede configurar a partir de la frecuencia de bus del microcontrolador, que como se había mencionado en la etapa de conversión A/D, había sido fijada en 7.9872 MHz.

A partir de esta frecuencia se pueden obtener múltiples frecuencias de baudios utilizando preescaladores que se pueden configurar en los registros de configuración del módulo SCI (la máxima frecuencia de baudios posible a esta frecuencia es de 124.800 baudios).

De la misma manera, en los registros de control SCCR se puede configurar el número de bits de datos, los bits de parada, y la paridad de la comunicación.

En la Figura 21 se puede ver la manera en que se conectó el controlador de USB FT232 y los elementos que se requieren para su funcionamiento.

Figura 21. Conexión del controlador USB.



Fuente: *Datasheet* FT232BM *Future Devices Inc*.

En el esquemático anterior se puede ver que no se utilizó la EEPROM para los datos de identificación, debido a que se utilizó la información por defecto

del fabricante del controlador USB 2.0. También se puede apreciar que la alimentación del dispositivo USB se realizó desde la etapa de alimentación externa con una fuente de 5V, pero si se desea, todo el sistema se podría alimentar desde el bus USB (teniendo en cuenta las limitaciones de corriente, 100 mA, que posee), ya que el controlador de USB 2.0 puede soportar el control de dispositivos de alta corriente utilizando el pin #PWREN.

El controlador FT232BM utiliza un cristal de 3 pines de frecuencia 6Mhz, que internamente se multiplica por cuatro para generar la señal de bus interno del dispositivo, también es posible utilizar un cristal de dos pines.

En los pines (#11) y (#12) se pueden conectar dos leds que indican si el controlador esta recibiendo o enviando datos desde el bus USB, un timer digital interno permite que la transmisión o recepción sea indicada en los leds incluso si ésta dura muy poco tiempo.

En el pin (#4), #RESET, se conecta un divisor de voltaje de dos resistencias que van a la línea de alimentación del bus USB, finalmente se utilizan 3 capacitores de desacople conectados entre Vcc y tierra.

4.5 ETAPA DE CONTROL

Esta etapa esta constituida por el microcontrolador de 8 bits MC68HC908GP32 perteneciente a la familia de HC08 de *Freescale*.

El microcontrolador realiza las siguientes funciones en el sistema ADQ:

- Controla el ciclo de conversiones del AD974 utilizando los pines de entrada – salida del puerto C.
- Recibe los caracteres del resultado de la conversión análoga digital utilizando la interfaz SPI del microcontrolador configurada como esclavo.
- Controla el ciclo de lectura escritura de la memoria SRAM externa utilizando los pines del puerto A.

- Lee y escribe los datos desde y hacia la memoria SRAM externa utilizando el puerto B.
- Genera los pulsos de reloj que incrementan el valor de los contadores para generar la dirección de la memoria.
- Provee y recibe datos desde el controlador de USB utilizando la interfaz SCI de comunicaciones asíncrona.

Tabla 11. Descripción de la función de los pines del microcontrolador

Puerto del microcontrolador	***Función***
PTA	Control de lectura escritura de memoria SRAM. Generar reloj para modificar la dirección actual.
PTB	Dato de entrada salida para la memoria SRAM.
PTC	Control del ciclo de conversión del AD974.
PTD	Comunicación síncrona serial utilizando la interfaz SPI.
PTE	Comunicación asíncrona serial utilizando interfaz SCI.

Fuente: Los autores.

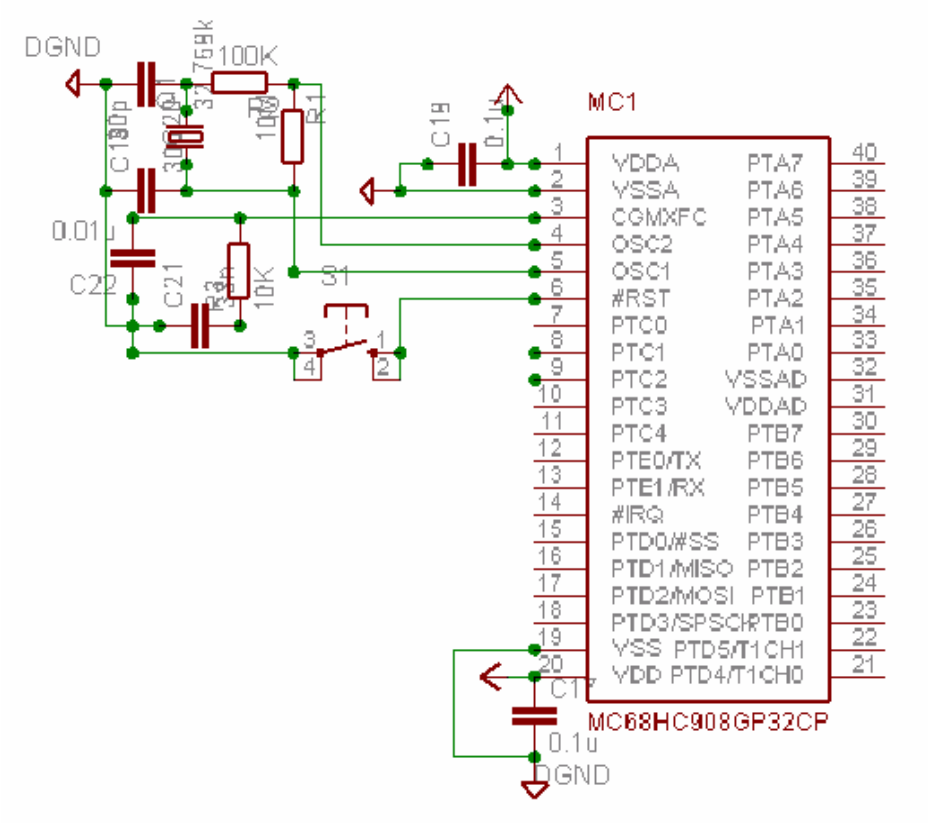
Con el fin de configurar la frecuencia de bus del microcontrolador, se trabajó con un cristal oscilador de baja frecuencia (32,768Kz) y el modulo PLL del microcontrolador habilitado, lo que permite que la frecuencia del oscilador sea multiplicada por un valor deseado para obtener el reloj de referencia del bus del microcontrolador. Esto permite que con un cristal externo de frecuencia 32.768KHz se configuren frecuencias de bus de hasta 8.2 MHz, que es la máxima frecuencia recomendada por el fabricante del microcontrolador; la frecuencia de trabajo que se configuró finalmente para la tarjeta es de 7.9872MHZ.

Si se desea trabajar con el módulo PLL del microcontrolador es necesario habilitarlo en el *firmware* y colocar un filtro externo. Este filtro externo y el circuito del oscilador de cristal se muestran en la figura 22.

Para habilitar el PLL y ajustar la frecuencia de bus necesaria es necesario modificar el *registro* de la CPU del microcontrolador. Esto se explicará con más claridad en el capítulo siguiente.

Los valores de los capacitores y de las resistencias de los circuitos osciladores y del filtro del PLL se ajustaron de acuerdo a las recomendaciones del fabricante del microcontrolador.
Se utilizan capacitores de desacople de 0.1μF entre cada una de las alimentaciones y tierra del microcontrolador.

Figura 22. Circuito de oscilador de cristal y filtro del PLL



Fuente: *Datasheet* MC68HC908GP32 *Freescale*.

El filtro del PLL se encuentra conectado en el pin (#3) del microcontrolador CGMXFC y el circuito de oscilación se conecta a los pines (#4) y (#5), OSC2 y OSC1 respectivamente.

El pin (#6) es la entrada de reset externo, que es activa baja, y debido a que posee una resistencia de *pull up* interna, no es necesario llevarla a VDD para trabajar normalmente.

4.6 ETAPA DE ALIMENTACION

La fuente de alimentación es una parte esencial del sistema de adquisición de datos puesto que es la encargada de suministrar la energía que requieren

todos los dispositivos y circuitos para desempeñar correctamente sus funciones.

Para el diseño de esta fase se tuvieron en cuenta los requerimientos de cada una de las etapas de la tarjeta de adquisición de datos: niveles de tensión, demanda de corriente y bajo nivel de rizado.

La fuente de alimentación permite obtener voltajes a partir de un suministro AC de 110V permitiendo la salida de voltajes DC positivos y negativos de ±12V para la entada y salida de los amplificadores de aislamiento ISO124, y una fuente de 5V para alimentar la etapa digital del sistema de adquisición de datos, además, todas las fuentes deben tener su tierra aislada de las demás etapas .

La fuente de alimentación cuenta con mecanismos de protección contra sobrecarga y cortocircuito. Antes de proceder a la explicación de su funcionamiento y construcción, es conveniente conocer algunos criterios importantes que se tuvieron en cuenta durante su diseño.

4.6.1 CRITERIOS DE DISEÑO DE LA FUENTE DUAL

Para simplificar el diseño de un circuito electrónico generalmente no se incluyen los parámetros de la fuente de alimentación, puesto que se supone que los mismos no afectan el comportamiento esperado del circuito. A continuación se describen algunos de los mas importantes.

4.6.1.1 LA REGULACIÓN DE LA CARGA

Este parámetro indica el cambio de voltaje de la salida cuando la corriente de carga varía entre sus valores mínimo y máximo. Se expresa frecuentemente

mediante una figura llamada porcentaje de regulación dada por la ecuación 2:

$$\%LR = \left(\frac{Vo_{NL} - Vo_{FL}}{Vo_{FL}} \right) * 100\% \qquad (2)$$

Esta expresión, %LR representa el porcentaje de regulación de la carga, Vo_{NL} el voltaje de salida con la mínima corriente de carga y Vo_{FL} el voltaje de salida cuando la corriente de carga es máxima. De la ecuación (2) se concluye que una buena fuente es aquella que ofrece un porcentaje de regulación cercano a cero. Para nuestro caso %LR = [(4.99-4.89)/4.89]*100= 2.044%

4.6.1.2 LA REGULACIÓN DE LA LINEA

Este parámetro define la variación en el voltaje sobre la carga para un cambio de la tensión de la red de alimentación de AC. Generalmente se expresa mediante una figura llamada porcentaje de regulación de red, dada por la ecuación 3.

$$\%SR = \left(\frac{Vo_{HL} - Vo_{LL}}{Vo_{N}} \right) * 100\% \qquad (3)$$

En esta expresión, %SR es el porcentaje de regulación de red, Vo_{HL} y Vo_{LL} los voltajes de salida regulados en la carga para la máxima y la mínima tensión de red, respectivamente, y Vo_{N} el voltaje nominal en la carga.

4.6.1.3 LA RESISTENCIA DE SALIDA

Este parámetro considera la perdida del voltaje regulado entregado a la carga, el cual es directamente proporcional a la corriente entregada. Los reguladores integrados utilizados LM7812, LM7912, LM7805, LM7905 poseen resistencias de salida muy bajas del orden de los miliohmios según la hoja de datos del fabricante *National Semiconductors*.

4.6.1.4 FACTOR DE RECHAZO AL RIZADO

Este parámetro define la atenuación que el regulador efectúa sobre la señal de rizado que se superpone al voltaje de entrada. El mismo está dado por la ecuación 4.

$$RR = \frac{Vr_{out}}{Vr_{in}} \qquad (4)$$

En la ecuación 4, Vr_{in} y Vr_{out} son los voltaje de rizado en la entrada y la salida del regulador, respectivamente. Para nuestro caso RR= [(13e-3)/(50e-3)]=0.26

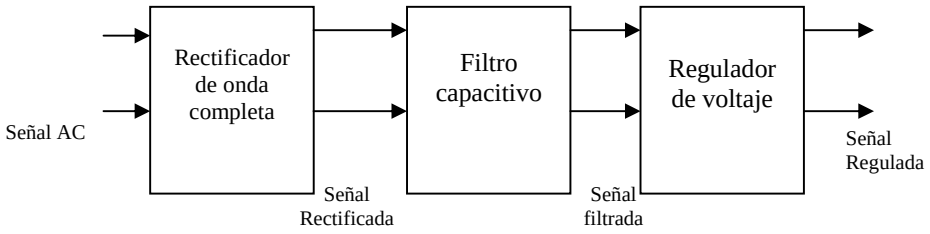
4.6.2 FUNCIONAMIENTO DE LA FUENTE DUAL

La fuente de alimentación bipolar, se compone de dos etapas complementarias, que se encargan de entregar el voltaje regulado positivo y negativo, y de un rectificador de onda completa, implementado con un puente rectificador encapsulado, el cual junto con un trasformador de tres devanados secundarios con salidas de 15V y 8V que es compartido por las etapas reguladoras. El transformador reductor se diseñó teniendo en cuenta la relación de vueltas N2/N1 = 15/115 y N2/N1= 8/115 para los secundarios.

La figura 23 muestra de manera simplificada las tres partes que componen cualquiera de las etapas reguladoras de la fuente. El diagrama esquemático del circuito de la fuente dual se muestra en la figura 24.

El primer bloque de la fuente dual es el rectificador, el cual se encarga de obtener, por medio del puente de diodos y a partir de la onda senoidal proveniente del secundario del transformador , una señal de voltaje monopolar pulsante.

Figura 23. Diagrama de bloques de una etapa regulada de la fuente

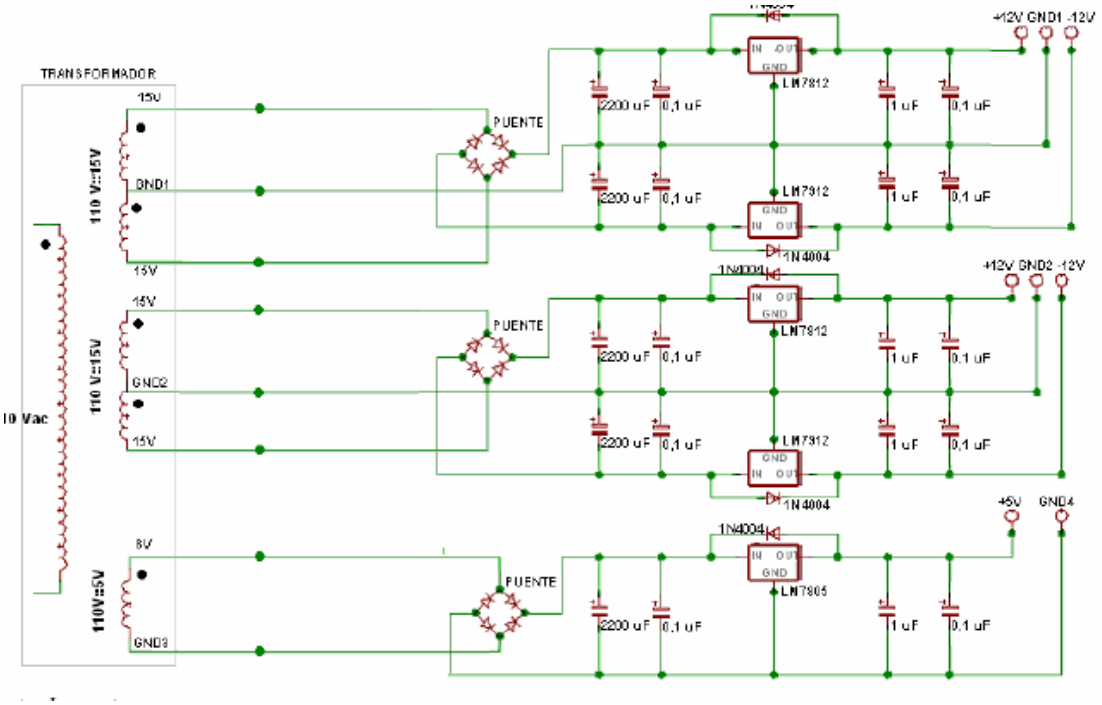


Fuente: Los autores

El segundo bloque es un filtro capacitivo que se encarga de recibir la señal rectificada pulsante, filtrarla y convertirla en una señal de corriente continua con un pequeño rizado. Para ello se utilizan los capacitores de 2200uF.

El tercer bloque es el de reguladores de voltaje, el cual se encarga de mantener estable la tensión que entrega la fuente ante los cambios que pueda presentar la corriente de carga, dentro de los limites establecidos por la capacidad de corriente de los circuitos integrados reguladores usados. Los principales componentes de este bloque son los reguladores de tensión fija de +12V LM7812, -12V LM7912, +5V LM7805. Se eligieron estos reguladores debido a sus buenas características de regulación de la carga , regulación de línea, resistencia de salida y rechazo de rizo, además porque son de fácil consecución en nuestro entorno.

Figura 24. Esquemático de la fuente de alimentación



Fuente: Los autores.

Todo circuito integrado regulador requiere unos pocos componentes externos para su optimo desempeño, los cuales se encargan de estabilizarlo, filtrar el ruido eléctrico y protegerlo contra las corrientes inversas de descarga. Para tal propósito, el diseño del circuito incluye los capacitores de 1uF y 100nF y los diodos 1N4004.

5. IMPLEMENTACION DEL *FIRMWARE*

En este capitulo se explicará como el *firmware* del microcontrolador se encarga de realizar todas las tareas que se requieren para llevar la señal hasta el computador, también se explicarán las funciones que cumple el driver del dispositivo controlador de USB y que hace posible la transmisión y recepción de datos desde el PC.

5.1 *FIRMWARE* DE LA TARJETA ADQ USB 2.0

El *firmware* consiste en el código que será ejecutado por el microcontrolador y que describe que debe hacer este en presencia de señales o eventos externos o internos, en el *firmware* también se configuran e inicializan las interfaces que se van a utilizar: como por ejemplo interfaces de comunicación o de conversión y se configura de que fuente se va a obtener el reloj del bus y a que frecuencia va a trabajar el mismo.

Para escribir el *firmware* de un microcontrolador cualquiera se puede utilizar diferentes herramientas de programación, algunas de estas son:

- Escribir el código en lenguaje ensamblador.
- Escribir el código en un lenguaje de alto nivel como C o C++.

Escribir el código en lenguaje ensamblador consiste en utilizar un lenguaje denominado de bajo nivel, que depende del fabricante del producto y de la arquitectura de la CPU del dispositivo, y que implica trabajar directamente con los registros y la memoria del microcontrolador cuando se escribe una instrucción. Si de desea trabajar con un lenguaje de alto nivel, se tiene la ventaja de poder escribir el programa en un lenguaje que no depende del set de instrucciones del microcontrolador, como por ejemplo C, lo que significa

que la programación será más independiente del dispositivo con el que se este trabajando. Si se desea escribir el código en un lenguaje de alto nivel es necesario tener un compilador que traduzca el programa a lenguaje de máquina comprensible para el microcontrolador específico con el que se esta trabajando.

En la programación del *firmware* de la tarjeta se eligió trabajar en C, utilizando el ambiente integrado de desarrollo que ofrece el programa CODEWARRIOR y específicamente la herramienta denominada Procesador Experto (*PROCESSOR EXPERT*), que permite configurar el microcontrolador agregando *beans,* que consisten en paquetes de programas que manejan la inicialización de las diferentes partes e interfaces del microcontrolador y que brindan subrutinas de operaciones básicas tales como escritura y lectura de registros de las interfaces que se agreguen el programa.

El Procesador Experto (*PROCESSOR EXPERT)* permite modificar los registros de configuración e inicialización de las interfaces agregadas al programa principal utilizando una ventana en donde se pueden escoger los valores deseados sin recurrir a escribir instrucciones en C. Esto permite que se agreguen fácil y rápidamente, para ser usadas, las diferentes herramientas o interfases que posee un microcontrolador y que permitan al usuario tenerlas disponibles sin perder tiempo en rutinas de inicialización, lo que le da la posibilidad de concentrarse en la rutina principal y de utilizar las interfaces simplemente llamando una subrutina.

En la figura 25 se puede ver un ejemplo del uso de los *bean* y de su ventana de configuración, la ventana de la izquierda corresponde a la configuración de la CPU del microcontrolador, se puede ver que en este caso se habilita el reloj desde el PLL y que se configura la frecuencia de bus deseada. La

ventana de la derecha corresponde a la configuración de la interfaz de comunicaciones SCI, en esta ventana se puede configurar los detalles de la comunicación tales como la rata de baudios, el número de bits de datos, la paridad y se pueden además habilitar las interrupciones de la interfaz.

Si no se utilizara el Procesador Experto sería necesario configurar estos valores utilizando los registros de la CPU, para lo cual se deben conocer cuales son estos registros y cual es su función.

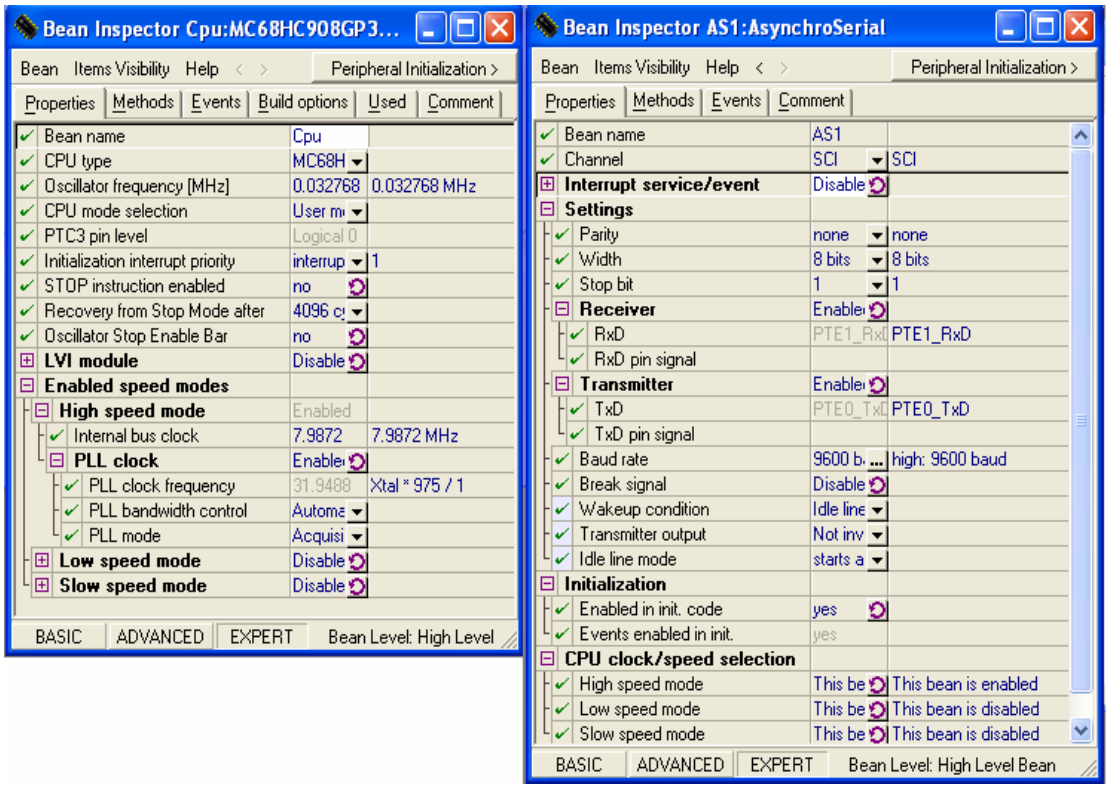
El Procesador Experto ahorra tiempo al crear las subrutinas de inicialización e incluirlas automáticamente en la rutina principal a ejecutar (*main)*, así mismo permite que el usuario no este tan familiarizado con la arquitectura del microcontrolador específico con el que está trabajando.

De la misma forma en que se configura el *bean* de la CPU y de la comunicación serial asíncrona SCI, se pueden configurar los *beans* para los puertos de entrada salida para las señales de control de los dispositivos tales como el AD974, y la memoria SRAM BEQ4017.

La comunicación serial síncrona SPI utiliza únicamente el *bean* de inicialización de la interfaz, por lo que en el código se hace referencia a sus registros. (Al utilizar el *bean* de SCI, por ejemplo, no se hace referencia en el código a registros de la interfaz sino a subrutinas agregadas en el *bean*).

Figura 25. Ejemplo de *bean* de configuración del GP32

a) Configuración de la CPU.
b) Configuración de la interfaz SCI.



a) b)

Fuente: Los autores

5.1.1 BLOQUE DE CONFIGURACIÓN DEL SISTEMA

El bloque de configuración del sistema consiste en los dos primeros bytes de la memoria SRAM, en donde estará almacenada la información sobre el número de canales que se desean adquirir, el tiempo de la prueba, e indicadores como el de prueba terminada o memoria llena. Este bloque de la memoria SRAM se escribirá desde una conexión previa con el PC, la función

de la información almacenada en este bloque de configuración se puede ver en el tabla 12.

Tabla 12. Bloque de configuración del sistema.

Dirección del Byte	Bit	Descripción
0	0	Habilita el canal 0 del conversor
	1	Habilita el canal 1 del conversor
	2	Habilita el canal 2 del conversor
	3	Habilita el canal 3 del conversor
	4 a 7	Divisores de duración de la prueba, si todos están en cero, la prueba durará el tiempo necesario para llenar completamente la memoria.
1	0	Indicador del modo de trabajo
	1	Indicador de prueba terminada, si esta en 1, la memoria se encuentra protegida de escritura.
	2	Configura si se desea guardar una prueba en memoria. (En el modo data logger siempre se graba en memoria).

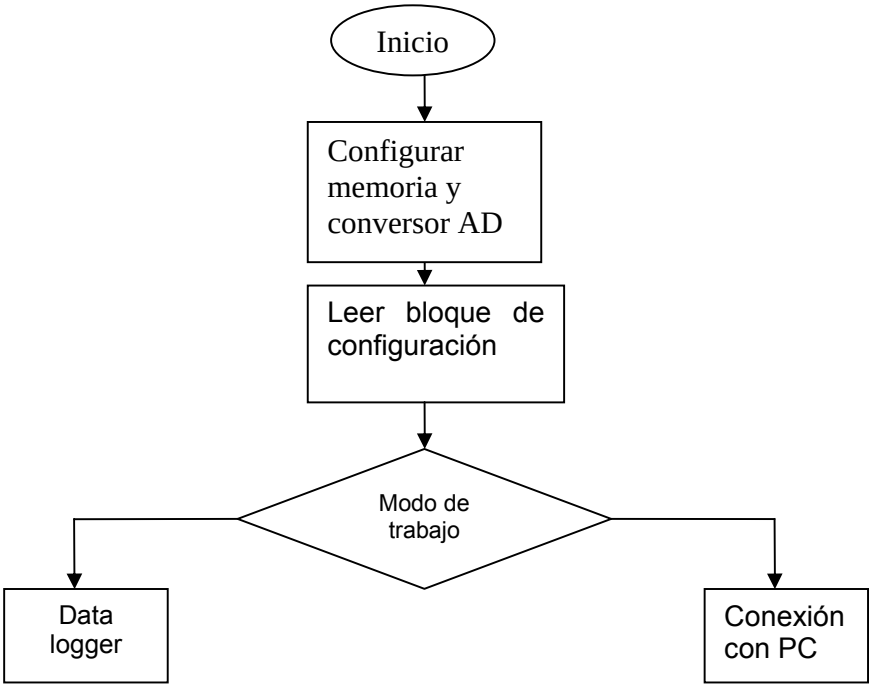
Fuente: Los autores.

5.1.2 DESCRIPCIÓN DE LA RUTINA PRINCIPAL DEL FIRMWARE.

La rutina principal es el código que comienza a ejecutarse una vez la tarjeta se ha encendido o el microcontrolador ha recibido una señal de reset. Al inicio de la rutina se inicializan los puertos que van a controlar al conversor AD y a la memoria SRAM, luego se lee el bloque de configuración del sistema y se verifica el valor del pin (#11) del microcontrolador (PTC4), para verificar el modo en el que va a operar la tarjeta.

En la figura 26 se puede ver el diagrama de flujo de la rutina principal del programa.

Figura 26. Diagrama de flujo del programa principal.



Fuente: Los autores.

Con el fin de mejorar a legibilidad del programa y hacer más sencilla la tarea de programar el *firmware,* se generó un archivo *header* o librería, que permite asignar nombres a los puertos para hacer referencia a ellos dentro de los programas y subrutinas que se escriben. A continuación se puede ver el código fuente de este archivo que se denominó ***pines.h***

```
#ifndef __pines_H
#define __pines_H

/* MODULE pines */
/*Pines para el control de conversión*/
 #define RC PTC_PTC0
 #define A0 PTC_PTC1
 #define A1 PTC_PTC2
 #define BUSY PTC_PTC3

/*pines para el control de la memoria*/
```

```
 #define status PTA_PTA7
 #define led_wmem PTA_PTA6
 #define led_rmem PTA_PTA5
 #define RESET4040 PTA_PTA4
 #define WE PTA_PTA3
 #define OE PTA_PTA2
 #define CE PTA_PTA1
 #define CLK PTA_PTA0

/* END pines */

#endif
```

A continuación se puede ver el código fuente de la rutina principal del *firmware* del microcontrolador.

```
#include "Cpu.h"
#include "Events.h"
#include "AS1.h"
#include "SPI1.h"
#include "TBM1.h"
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"
#include "pines.h"
#include "subrutinas.h"

byte blk1,blk2;
void main(void)
{
  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
  PE_low_level_init();
  /*** End of Processor Expert internal initialization.                    ***/
  /*-------------------------------------------*/
  /*Inicialización de los pines del conversor AD*/
DDRC=0x07;/*'b00000111'*/
PTCPUE_PTCPUE4=1;/*pull up habilitado*/
RC=1;
A0=0;
A1=0;

/*Inicialización de los pines de la memoria*/
DDRA=0xff;
led_wmem=0;
led_rmem=0;
status=1;/*estado general*/
CE=0;
CLK=1;
RESET4040=1;
RESET4040=0;
WE=1;
```

```
OE=1;

blk1=leerbloque();
blk2=leerbloque();

if(MODE==0){/*modo de data logger*/
  conver_cont();
}else{
  wait_char(); /*esperar caracter desde el PC*/
}
    /*-----------------------------------------*/
#ifndef PE_OS_OSEK_SUPPORT
  for(;;){}
#else
  StartOS(Mode);                    /* Jump to OSEKturbo OS startup */
  /*DO NOT WRITE CODE BELOW THIS LINE*/
#endif PE_OS_OSEK_SUPPORT
}
```

En el anterior código es posible ver varias referencias a subrutinas de usuario tales como *conver_cont(), o wait_char()*, todas estas funciones se encuentran en la librería de usuario denominada ***subrutinas.c***, cada una de estas funciones se irá describiendo en durante este capítulo.

La función *PE_low_level_init()* es introducida automáticamente en la rutina principal por el compilador, y se encarga de configurar los registros de las interfaces que se hayan agregado mediante el uso de *beans*.

La subrutina de lectura del bloque de configuración se encuentra a continuación:

```
extern byte blk1,blk2;
byte leerbloque(void){
byte a;
OE=0;
a=PTB;
OE=1;
CLK=0;
CLK=1;
return a;
}
```

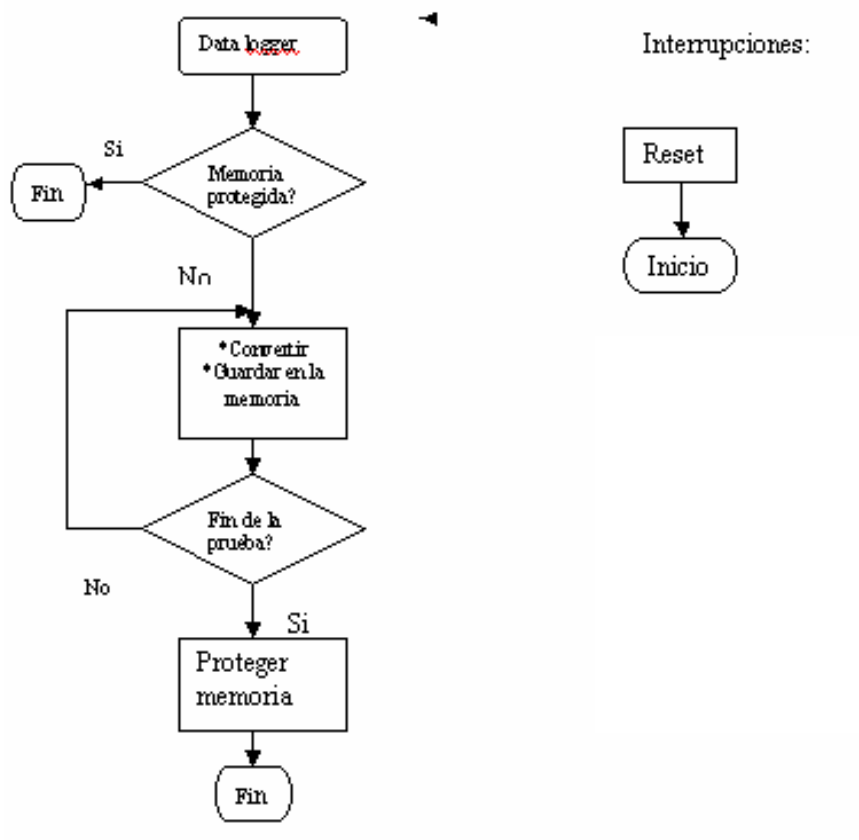
5.1.2.1 MODO DATA LOGGER

Si la tarjeta esta programada para realizar una adquisición en el modo de *data logger* (lo que se hace al colocar el pin (#11), PTC4 a tierra mediante un jumper), y una vez se encienda la tarjeta o se resetee el sistema, el *firmware* del microcontrolador se encargará de dar las señales necesarias para que el conversor análogo digital comience a adquirir las señales desde los canales que hayan sido programados en el bloque de configuración de la tarjeta, así mismo, proveerá de datos y direcciones a la memoria SRAM externa hasta que esta se encuentre llena, evento en el cual se modificará el bloque de configuración protegiendo los datos almacenados en la memoria SRAM.

Si se produce una desconexión o reset del sistema antes de que la memoria esté llena, entonces la memoria comenzará a llenarse con los datos de la conversión nuevamente desde el comienzo.

Si se desea trabajar con conexión al PC, se debe retirar el jumper que configura el modo de *data logger* y reiniciar la tarjeta. En la figura 27 se puede ver el diagrama de flujo general de este modo de trabajo.

Figura 27 Diagrama de flujo del modo data logger.



Fuente: Los autores

A continuación se muestra el código fuente del modo de trabajo de *data logger*.

```
/*conversion en modo data logger*/
void conver_cont(void){
bool ch0,ch1,ch2,ch3,mem_pro;
int dur;
ch0=blk1&0x01;
ch1=blk1&0x02;
ch2=blk1&0x04;
ch3=blk1&0x08;
dur=blk1&0xf0;
dur=dur>>4;
dur++;
mem_pro=blk2&0x02;
dur=1048574/dur;/*mem=1048576*/
if(mem_pro==0){/*si memoria no esta protegida*/
```

```
  CLK=0;
  CLK=1;
  CLK=0;
  CLK=1;
  while(dur){
    if(ch0){
      A0=0;
      A1=0;
      conver_guar(1);
    }
    if(ch1){
      A0=1;
      A1=0;
      conver_guar(1);
    }
    if(ch2){
      A0=0;
      A1=1;
      conver_guar(1);
    }if(ch3){
      A0=1;
      A1=1;
      conver_guar(1);
    }
    dur--;
  }
blk2=0x06;                    /*prueba terminada, protege memoria*/
RESET4040=1;
RESET4040=0;/*resetea contadores*/
CLK=0;
CLK=1;
DDRB=255;
PTB=blk2;          /*escribe bloque de configuración*/
WE=0;
WE=1;
}
}
```

Este código fuente es la traducción a lenguaje C del diagrama de flujo de la figura 27. La función *conver_guar()* se encarga de convertir y guardar en la memoria SRAM, está incluida en la librería ***subrutinas.h*** y su código fuente aparece a continuación.

```
void conver_guar(int num){
  int i;
 char datoh,datol;
 DDRB=255;
 for(i=0;i<=num;i++){
  RC=0;
  RC=1;
  while(SPSCR_SPRF==0);
```

```
 datoh=SPDR;
 while(SPSCR_SPRF==0);
 datol=SPDR;
 PTB=datoh;
 WE=0;
 WE=1;
 CLK=0;
 CLK=1;
 PTB=datol;
 WE=0;
 WE=1;
 CLK=0;
 CLK=1;
 while(BUSY==0);
   }
}
```

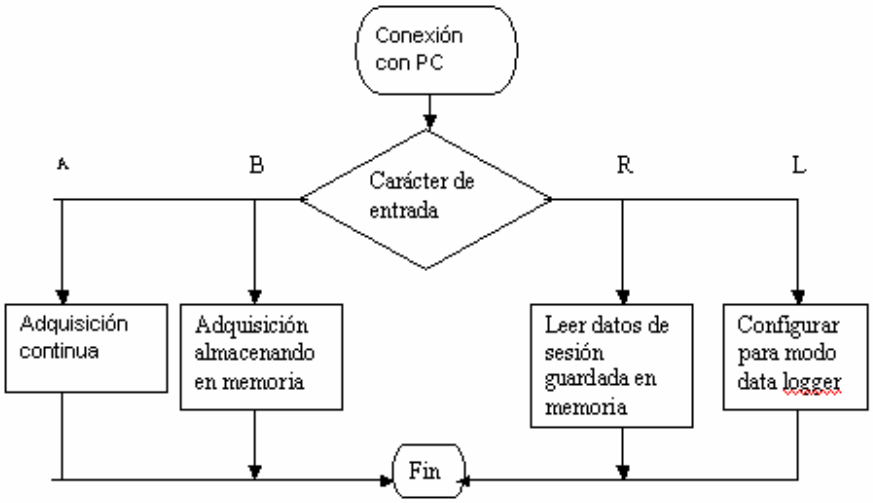
Esta subrutina recibe como entrada el número de conversiones que se solicitan.

5.1.2.2 CONEXIÓN DE LA TARJETA CON EL PC

Si la tarjeta entra en el modo de conexión al PC, ya sea por haber recibido una interrupción por haber recibido un carácter en la interfaz SCI del microcontrolador, o por entrar en este modo debido a la información que se encontró en el bloque de configuración, la tarjeta esperará un carácter de control que le será enviado por la aplicación en labview y que indicará que tarea deberá ser ejecutada por la tarjeta:

- Hacer una adquisición continua sin guardar en memoria SRAM externa.
- Realizar una prueba de cierta duración guardando en memoria.
- Recuperar los datos de una sesión anterior leyéndolos desde la memoria SRAM externa.
- Configurar una adquisición en modo de *data logger*.

Figura 28. Diagrama de flujo cuando la tarjeta esta conectada al PC



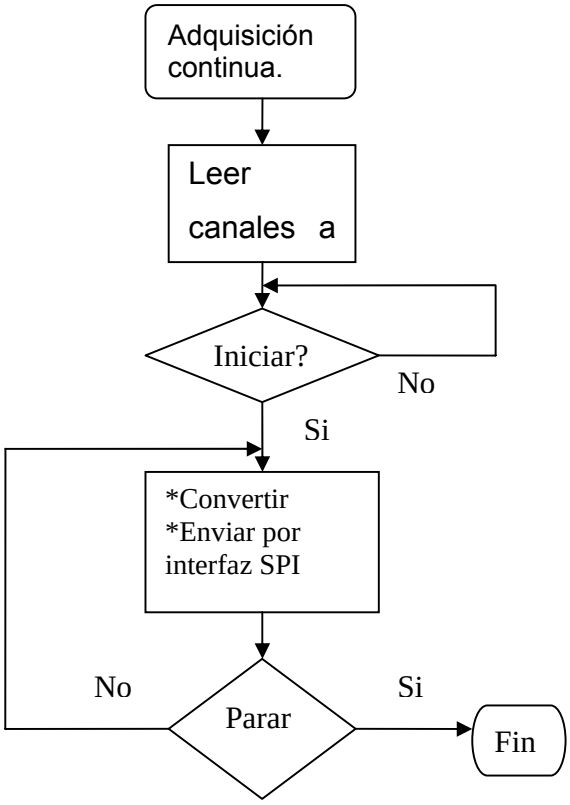
Fuente: los autores.

- Adquisición continua: En este modo se realizan continuamente conversiones y se envían de inmediato a través de la interfaz SCI del microcontrolador al controlador de USB FT232BM, que los transfiere al PC a la aplicación en Labview.

Al comienzo de la subrutina se recibe la información de cuáles canales se van a convertir.

La subrutina termina si se recibe un carácter de parada.

Figura 29. Diagrama de flujo de la subrutina de adquisición continua.



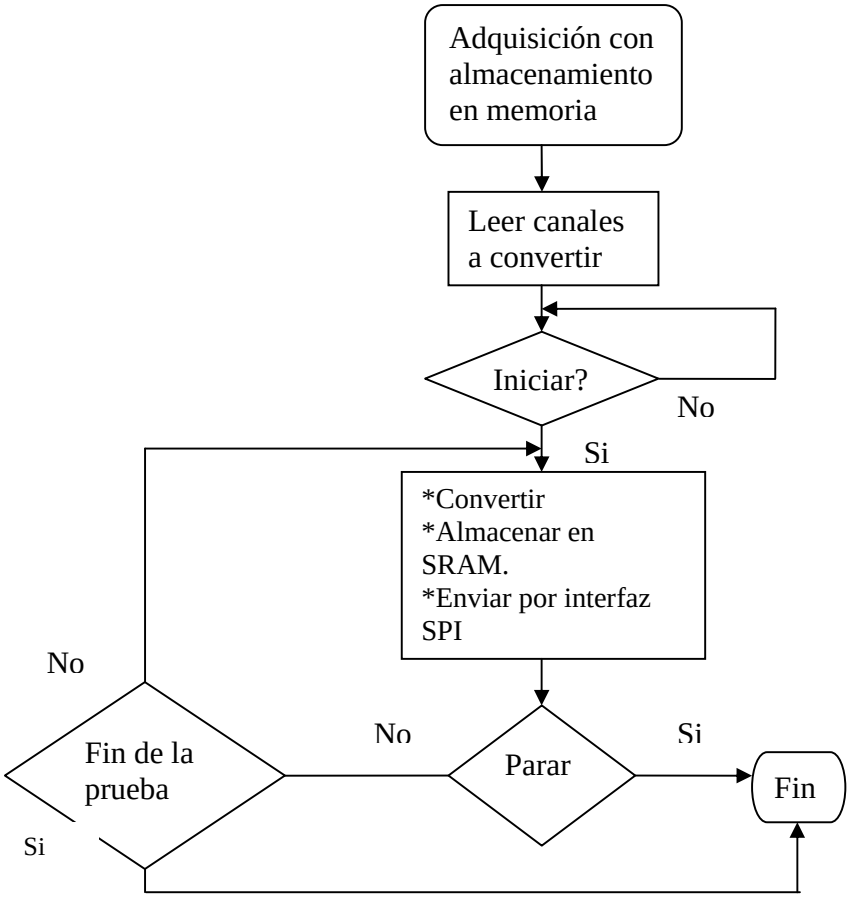
Fuente: Los autores.

- Adquisición almacenando en memoria: En este modo se configura el tiempo de duración de la prueba y el número de canales que se van a convertir.

Por cada conversión se guardan los datos en memoria y se envían utilizando la interfaz SPI.

Al inicio de la prueba se lee el número de canales y el tiempo de duración de la misma.

Figura 30. Subrutina de adquisición con almacenamiento en memoria SRAM.

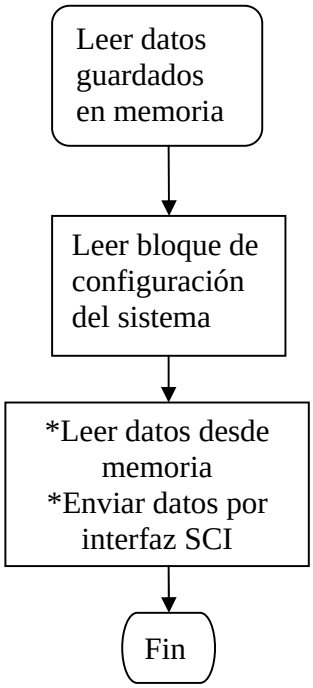


Fuente: Los autores.

- Leer datos de conversión guardada en memoria SRAM externa: en este modo se pueden recuperar los resultados de un aprueba realizada anteriormente.

El programa comienza leyendo el bloque de configuración del sistema para saber cuantos canales se han convertido y el tiempo de duración de la prueba.

Figura 31. Subrutina de lectura de datos de una sesión guardada en memoria SRAM.

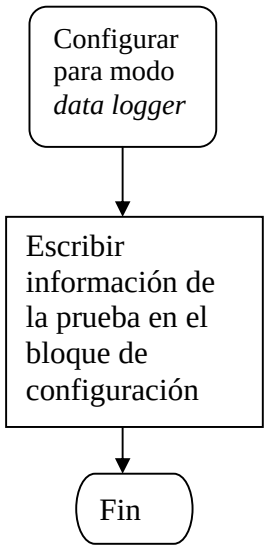


Fuente: Los autores

- Configurar conversión en modo de *data logger:* Esta subrutina permite configurar la tarjeta para que realice una adquisición sin estar conectada al PC.

En esta subrutina se eligen los canales que se van a utilizar y la duración de la prueba. Cuando la tarjeta se encienda nuevamente comenzará a convertir y almacenar en memoria de acuerdo a la información dada en el bloque de configuración.

Figura 32. Diagrama de flujo de la subrutina de configuración del modo data logger.



Fuente: Los autores.

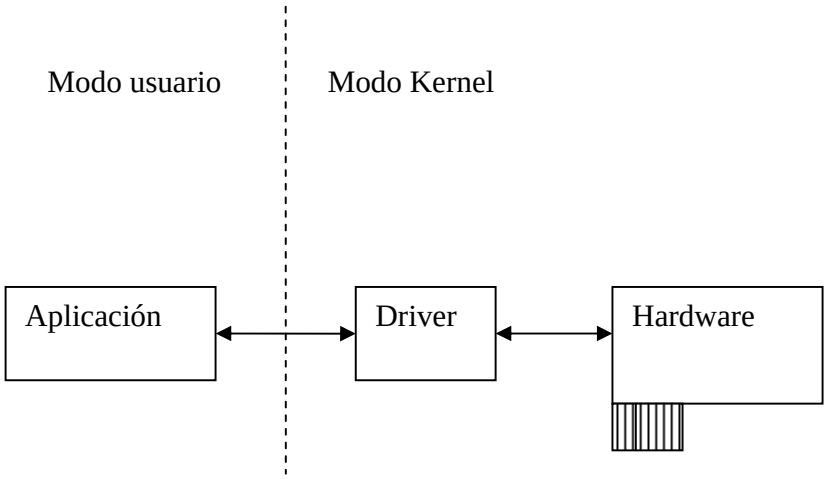
5.2 DRIVERS DE LA TARJETA ADQ USB

Un *driver* es una interfaz de software que provee comunicación entre el sistema operativo y un dispositivo de hardware tal como puede ser un disco o una memoria externa o un teclado.

El *driver* permite que las aplicaciones se comuniquen con el dispositivo de hardware.

En la figura 33 se puede ver un diagrama de bloques de la función de un driver como interfaz entre la aplicación de usuario, el sistema operativo y el hardware utilizado.

Figura 33. Función del driver



Fuente: Los autores.

El hardware es enlazado con el *driver* a través del sistema operativo mediante la información que se provee en el archivo de información .INF, en donde aparece la información de los archivos necesarios para la instalación del dispositivo. Para mas detalles acerca de la instalación del driver de la tarjeta ADQ USB se puede consultar en el anexo A.

6. DISEÑO DE LA INTERFAZ DE SOFTWARE EN LABVIEW 7.0

La tarjeta de adquisición de datos por bus USB, requiere de un interfaz gráfica para la visualización de los datos en el PC. Para ello se escogió una de las herramientas más poderosas y ampliamente estudiadas en el mercado para la implementación de la instrumentación y el control industrial como es el programa LABVIEW (*Laboratory Virtual Instrument Engineering Workbench*) de la empresa *National Instruments*. Este programa constituye un entorno de desarrollo basado en Lenguaje G o gráfico con la cual se puede estructurar cualquier algoritmo. Además cuenta con los elementos típicos de cualquier lenguaje de programación como son estructuras, arreglos, *for, while*, etc, también posee herramientas de depuración que proporciona cualquier compilador tradicional como los puntos de ruptura o correr los programas paso a paso. De igual forma posee gran variedad de VIs Express ya elaborados que permiten desarrollar rápidamente cualquier aplicación.

Los programas hechos en LABVIEW se conocen con el nombre de VIs (Instrumentos Virtuales) y se componen de dos partes fundamentales: el panel frontal, que puede asemejar en aspecto a los aparatos de instrumentación electrónica y el diagrama de bloques. Tiene la ventaja de facilitar una integración con el hardware de periféricos al PC, específicamente con tarjetas de medición, adquisición y procesamiento de datos.

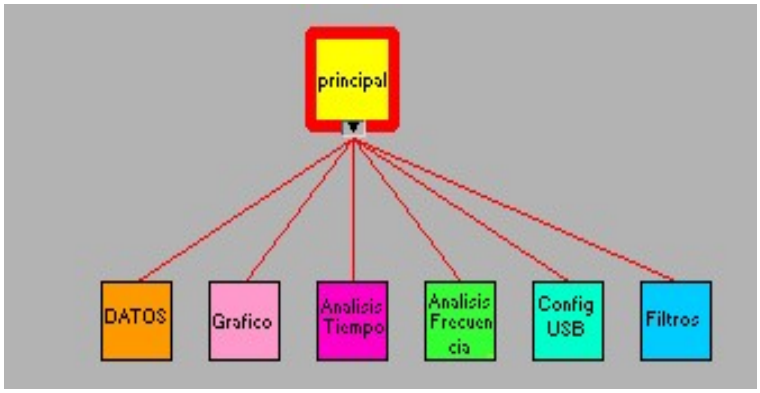
Los VIs se caracterizan por ser un bloque con su respectivo símbolo relacionado con su funcionalidad, posee una interfaz con el usuario, con entradas con su respectivo color de identificación de datos, y una o varias salidas. Un VI puede utilizarse en cualquier otra aplicación como una subfunción dentro de un programa general; lo que facilita eventualmente la

programación estructurada haciéndola más ordenada; modular y fácil de entender.

6.1 DESCRIPCIÓN DE LA APLICACIÓN DE LA TARJETA DE ADQUISICIÓN DE DATOS POR BUS USB

El programa principal consta de un conjunto de VIs que pueden ejecutarse en tiempo real compuesta de varios subVIs como se aprecian en la Tabla 13 diseñados en LabView 7.0 Express. Cada uno de estos subVIs tiene una función particular como por ejemplo configurar el puerto de comunicaciones, el almacenamiento de los datos y gráficos, el análisis en frecuencia, el análisis en el tiempo. La jerarquía de los subVIs implementados en la aplicación principal se muestra en la figura 34.

Figura 34. Esquema de la aplicación implementada en LABVIEW



Fuente: Los autores

Tabla 13. SubVIs que conforman el programa principal

SubVI Configuración de la comunicación al bus USB
SubVI Análisis en el tiempo
SubVI Análisis en la Frecuencia
SubVI Guardar Datos
SubVI Análisis de Filtros
SubVI Almacenar Gráficos

6.1.1. SUBVI DE ANÁLISIS EN EL TIEMPO

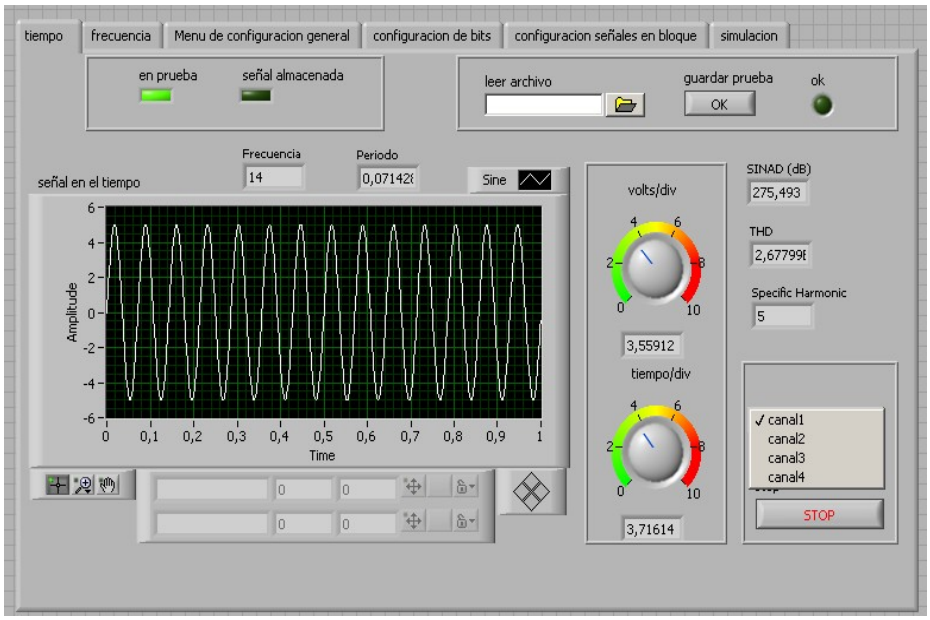
La aplicación principal desarrollada en LabView, se muestra en la figura 35. El usuario activa el inicio de prueba, permitiendo a LabView inicializar la captura de los datos adquiridos con la tarjeta de adquisición de datos USB 2.0. La tarjeta de adquisición de datos posee las perillas estándar de un osciloscopio digital para poder elegir las divisiones del voltaje y el tiempo, también puede seleccionar abrir un archivo de una prueba que se almacenó en formato gráfico de LabView (*.lvm) o archivo de texto (*.txt). A continuación se abre un cuadro de diálogo que permite escoger la ruta y el nombre del archivo que contiene la prueba que se desea visualizar. Cuando se escoge la ruta y el archivo, la prueba es desplegada para su análisis por el operador. Además en este programa se puede conocer los datos relacionados con la frecuencia de la señal de entrada y su amplitud.

6.1.2. SUBVI ANALISIS EN LA FRECUENCIA

En el SubVI de análisis de la señal en frecuencia, se visualizan todo los parámetros correspondientes a los armónicos de la señal, se implementan los filtros digitales eligiendo el canal que se desea analizar, como se muestra

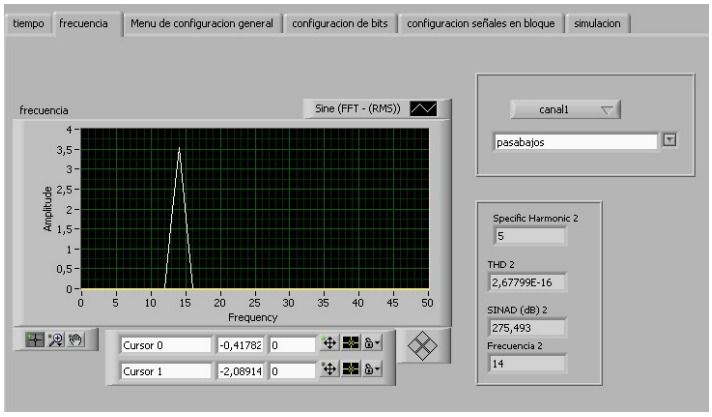
en la figura 36 con sus respectivos cursores para identificar el armónico y su amplitud.

Figura 35. Interfaz principal



Fuente: Los autores

Figura 36. SubVI del análisis en la frecuencia

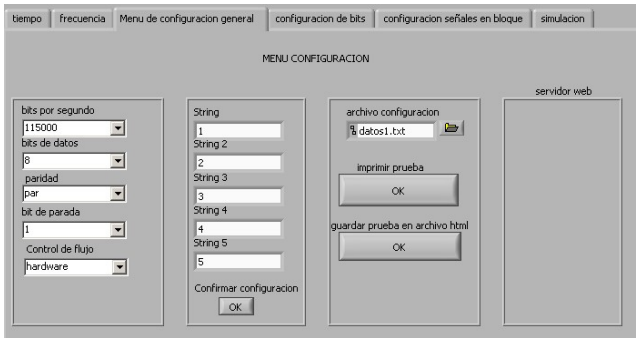


Fuente: Los autores

6.1.3. SUBVI CONFIGURACIÓN DEL PUERTO

En la figura 37 se muestra el SubVI de configuración del bus USB, en el cual se realizan todos los procedimientos y operaciones necesarias para habilitar los canales físicos que permiten iniciar la comunicación de la tarjeta de adquisición de datos por bus USB con el PC. Además se puede seleccionar un archivo almacenado con anterioridad para utilizarlo en la configuración; así mismo se puede imprimir la prueba o guardarla en archivo html. Igualmente se puede configurar la tarjeta para su conexión con el servidor web para realizar la configuración de la tarjeta vía Internet.

Figura 37. Configuración de Comunicaciones



Fuente: Los autores

6.1.4 SUBVI FILTROS DIGITALES

En este SubVI se configuran los filtros digitales que sean necesarios para eliminar los armónicos provenientes de la red eléctrica como por ejemplo el de 60 hz., también se coloca un filtro anti-solapamiento, y otros que permitan eliminar el ruido de la frecuencia admitiendo mayor exactitud en la adquisición en las señales.

6.1.5 SUBVI DATOS

Cuando ya se han adquirido los datos, y se ha realizado el análisis en el tiempo y en la frecuencia, los datos se pueden guardar en archivos de texto *.txt para hacer posteriores pruebas o comparar los resultados. El modo en que se puede almacenar estos resultados en archivo *.txt, *.html, entre otros. Se puede escoger donde guardar los archivos de acuerdo al formato a utilizar.

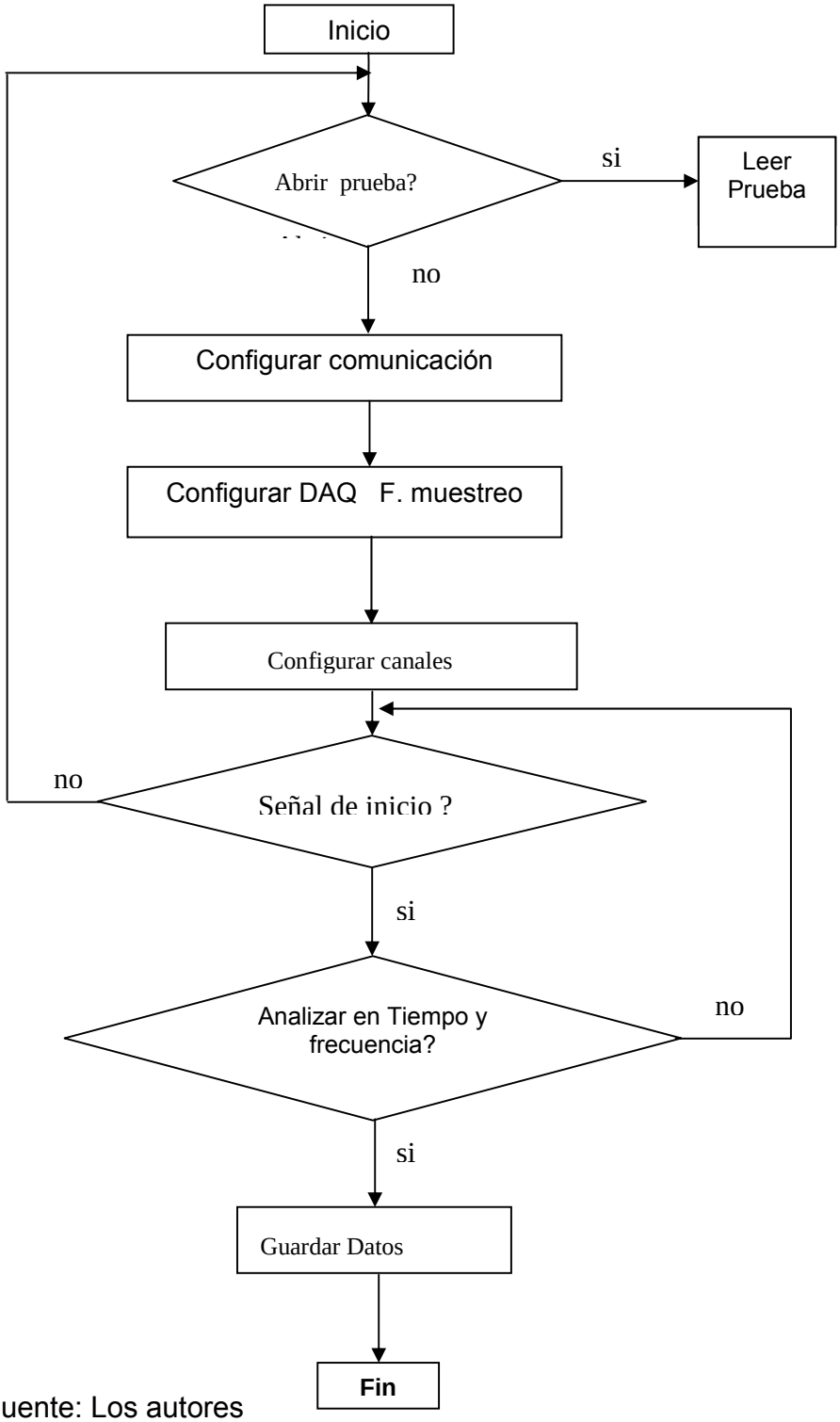
6.1.6 SUBVI DE IMPRESIÓN DE DATOS

En muchas ocasiones Las pruebas se desea imprimir en formato html para llevar a un servidor web o en formato txt para hacer análisis; para ello, de cada prueba, se puede imprimir el ensayo en diversos formatos para utilizarlos en análisis posteriores en diferentes programas como son Matlab.

6.2 DESCRIPCIÓN DEL ALGORITMO IMPLEMENTADO EN LABVIEW.

El algoritmo desarrollado muestra el proceso que se realiza para lograr una adquisición de datos empezando por la configuración de la comunicación, haciendo el análisis de datos en el tiempo, y en la frecuencia y almacenando los datos en archivos de texto. Como se muestra en la figura 38.

Figura 38. Algoritmo del programa en LABVIEW



Fuente: Los autores

7. CONCLUSIONES

1. Con el fin de optimizar la comunicación serial sincrona SPI entre el conversor A/D y el microcontrolador se utilizó el reloj interno del conversor AD974 en modo maestro permitiendo el control de los datos seriales de salida a la máxima velocidad permitida por el conversor A/D.

2. El conversor A/D posee una señal de reloj interna en modo maestro que tiene un periodo de 220ns. La frecuencia mínima de bus del microcontrolador debería ser 4.5Mhz para poder trabajar con esta frecuencia de reloj, sin embargo como se requieren dos bytes (16 bits) para transmitir el resultado de la conversión es necesario incrementar la velocidad del bus del microcontrolador a por lo menos 7.9872Mhz para evitar un error de desborde (*overflow*), al no poder leerse el primer byte desde el registro de datos del modulo SPI antes de que el segundo byte del dato se encuentre en el registro de desplazamiento, de manera que el segundo byte se pierda.

3. El controlador externo USB 2.0 de la tarjeta de adquisición de datos permite una tasa de transferencia real de hasta 960Kbps, utilizando un DSP con una frecuencia de bus de 60Mhz y una transferencia real de hasta 124.8Kbps utilizando el microcontrolador con una frecuencia de bus de 7.9872Mhz. La limitación del microcontrolador se debe a que la frecuencia máxima de bus recomendada por el fabricante es de 8.2Mhz.

4. El sistema de adquisición realiza las transferencias USB del tipo isócronas y por volumen (*bulk*) permitiendo el envio de microtramas de 1ms en paquetes de 1024 bytes.

5. El driver de puerto virtual serial (VCP) permite la comunicación entre el sistema operativo y el dispositivo de hardware desarrollado permitiendo que

la aplicación desarrollada en Labview se comunique con el dispositivo de hardware a través de un puerto serial virtual.

6. Con el fin de evitar direccionar la memoria SRAM de 16Mb directamente desde el microcontrolador, lo que significaría utilizar 21 pines de entrada – salida, se utilizaron contadores para obtener la dirección en base a una señal de reloj, con la desventaja de no poder acceder rápidamente a una posición específica de memoria. Sin embargo la forma en que se accede a la memoria es de manera secuencial, (una palabra es leída o escrita y enseguida la siguiente posición de memoria es leída o escrita y así sucesivamente), de manera que generar la dirección con contadores es una solución que ocupa pocos pines del microcontrolador y que se ajusta a los requerimientos del sistema.

7. Labview 7.0 permite la configuración de la comunicación USB con el dispositivo de hardware mediante las funciones VISA implementando inclusive velocidades de conexión no estándar para realizan todos los procedimientos y operaciones necesarios para habilitar los canales físicos que permiten iniciar la comunicación de la tarjeta de adquisición de datos por bus USB con el PC.

8. RECOMENDACIONES PARA FUTUROS PROYECTOS

1. Con el fin de incrementar la velocidad del sistema de adquisición se recomienda migrar al desarrollo de sistemas de adquisición de datos basados en procesadores digitales de señal DSP para incrementar la frecuencia de bus y aprovechar al máximo las capacidades del controlador USB 2.0.

2. Para desarrollar la conversión A/D de la señal se recomienda implementar la comunicación paralela con DSP aprovechando el numero de pines de propósito general y la velocidad de bus de estos dispositivos.

3. En aplicaciones móviles es deseable implementar el uso de una pantalla LCD grafica para desarrollar la visualización en tiempo real de las señales adquiridas en los cuatro canales analógicos de entrada.

4. Debido al auge de las memorias Nand flash de estado sólido se recomienda incursionar en el desarrollo de sistemas autónomos para el almacenamiento masivo de datos en conjunto con controladores USB 2.0 de almacenamiento masivo de datos (1Gb).

5. Se recomienda el desarrollo de drivers mediante el kit de desarrollo de drivers DDK de Microsoft en conjunto las librerías que provee FTDI para el desarrollo de aplicaciones de propósito específico.

BIBLIOGRAFÍA

[1] PALLAS Areny Ramón. Sensores Y Acondicionadores De Señal. Barcelona España. Alfaomega Grupo Editor 2001.

[2] LAZARO Manuel Antonio. Labview 6i Programación Gráfica Para El Control De Instrumentación. Editorial Paraninfo. Madrid España 2001.

[3] HOLMAN P. Jack. Métodos Experimentales Para Ingenieros Editorial Mc Graw Hill. México 1986 4a Edición.

[4] NATIONAL INSTRUMENTS. Measurement And Automation. Catalog 2003. National Instruments. Austin Texas Usa 2003.

[5] CEBALLOS F. Javier. Visual C++ Programación Avanzada En Win 32 Editorial Alfaomega. México DF México 2000.

[6] PROAKIS J.M, Dimitris G. Tratamiento Digital De Señales. Editorial Prentice Hall 3ª Edición 1988.

[7] AN EMBEDDED CONVERTER FROM RS232 TO UNIVERSAL SERIAL BUS. de Almeida Pereira Zuquim, L.D.; Coelho, C.J.N., Jr.; Fernandes, A.O.; de Oliveira, M.P.; Tavares, A.I.. Dept. de Ciencia da Comput., Univ. Fed. de Minas Gerais, Belo Horizonte, Brasil. Integrated Circuits and Systems Design, 2001, 14th Symposium. Brasil.

[8] UNIVERSAL SERIAL BUS (USB) POWER MANAGEMENT. Lynn, K.. Micrel Semicond., San Jose, CA, USA. Wescon/98 1998.

[9] HIGH SPEED DATA TRANSMISSION COMMON MODE NOISE SUPPRESSION - APPPLICATION TO USB 2.0 AND IEEE 1394. Yeh, P.; Wang, A.; Tseng, B.C.. Walsin Technology Corponition. Electronic Materials and Packaging, 2002. Proceedings of the 4th International Symposium on Pages: 488 – 491. This Conference was Held : Dec. 4-6, 2002.

[10] UNIVERSAL SERIAL BUS ENHANCES VIRTUAL INSTRUMENT-BASED DISTRIBUTED POWER MONITORING. Chung-Ping Young; Devaney, M.J.; Shyh-Chyang Wang. Dept. of Electr. Eng., Missouri Univ., Columbia, MO, USA 2000. Instrumentation and Measurement, IEEE Transactions on Pages: 1692 – 1697.

[11] ANALOG DEVICES. Practical design techniques for sensor signal conditioning. Estados Unidos. Analog Devices. 1999. p 5.26-5.28.

[12] COMPAQ COMPUTER CORP., Intel Corp., Microsoft Corp., NEC Corp. “Universal Serial Bus Specification” Rev 1.1, Septiembre 23, 1998.

[13] J. AXELSON. “USB complete”. 2ed. Madison,USA: Lakeview Research, 2001.

[14] J. HYDE. “USB Designed by example”. Intel, 2ed. Capítulos 1, 2, 3 y 4. Año 2001

[15] J. AXELSON’S “LAKE VIEW RESEARCH, home page”. Disponible: http://www.lvr.com/.

[16] METROWERKS, home page. Disponible: http://www.metrowerks.com/mw/default.htm.

[17] NATIONAL INSTRUMENTS. “Labview Express 7.0. User manual”. Disponible: http://www.ni.com.

[18] MOTOROLA. “Technical Data – MC68HC908GP32”, Rev 2.1 12/2.003, Secciones 9, 10 y 12.

[19] FUTURE TECHNOLOGY DEVICES. “Technical Data DS232B – FT232BM”, Version 1.4. 2004. paginas 1-25
http://www.ftdichip.com.

[20] COMPAQ COMPUTER CORP., Intel Corp., Microsoft Corp., NEC Corp. “Universal Serial Bus Specification” Rev 2.0, 2000.

ANEXOS

Anexo A. Manual de instalación driver USB FT232BM

1. INSTALACIÓN EN WINDOWS XP, ME 2000

FTDI Win 2k Drivers - Revision Abril 20, 2004

Estos drivers han sido cambiados para permitir la compatibilidad en Windows XP, win98 y Windows 2000. Todos los controladores de FTDI usan un solo archivo INF con los datos de configuración de los controladores USB

Existen seis archivos estándar en la versión del nuevo driver para estos sistemas operativos Windows

FTDIBUS.SYS (dos archivos driver)
FTSER2K.SYS
FTDIUNIN.EXE (desistalador y asociado al archivo ini)
FTDIUN2K.INI
FTDIBUS.INF (dos archivos INF para Win98 y Win2k)
FTDIPORT.INF

Para nuevas versiones consultar la pagina web de FTDI
http://www.ftdichip.com. Puede ir a la pagina de soporte de drivers y descargar la ultima actualización

2. INSTALACIÓN EN WINDOWS 98

Estos drivers han sido cambiados para permitir la compatibilidad en Windows 98. Todos los controladores de FTDI usan un solo archivo INF con los datos de configuración de los controladores USB

Existen once archivos estándar en la versión del nuevo driver para este sistema operativo

FTSERIAL.SYS (driver files)
FTSENUM.SYS
FTSENUM.VXD
FTCOMMS.VXD
FTSERMOU.INF
FTSERMOU.VXD

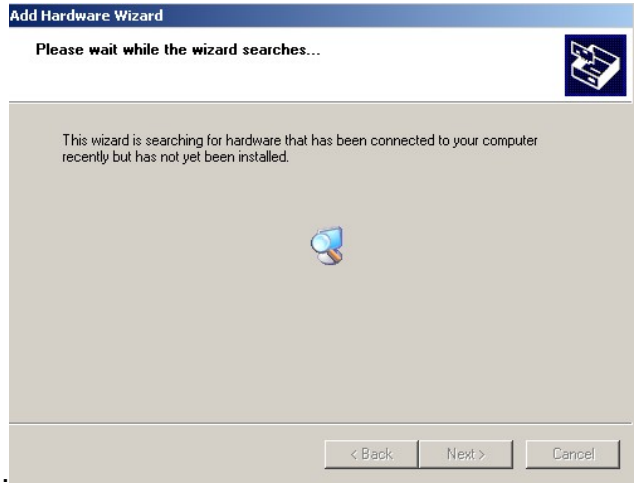
FTSERUI.DLL
FTDIUNIN.EXE (uninstaller and associated ini file)
FTDIUNIN.INI
FTDIBUS.INF (two shared INF files for Win98 and Win2k)
FTDIPORT.INF

3. PASOS A SEGUIR PARA LA INSTALACIÓN

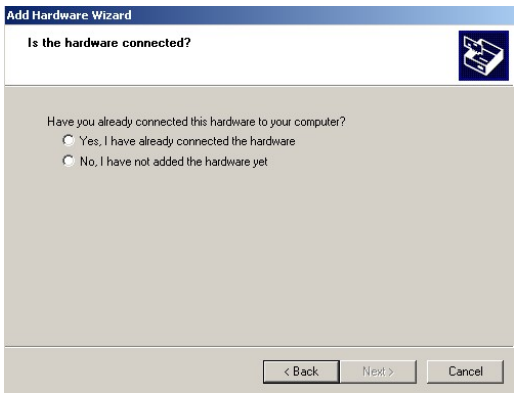
1. Inserte la tarjeta en el bus USB del PC. Si la tarjeta esta alimentada con la fuente externa el PC detectara la tarjeta, si es la primera vez Windows le pedirá los drivers para el dispositivo desconocido que acaba de detectar.

2. Vaya a panel de control agregar hardware y aparecerá un menú que lo guiará para instalar su dispositivo USB. Escoja siguiente (next)

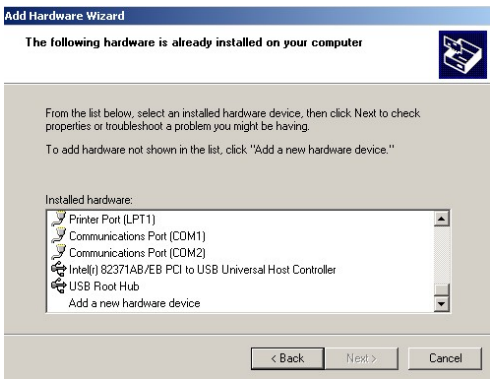
3. El sistema operativo buscara automáticamente el dispositivo y el driver mas apropiado para la tarjeta.



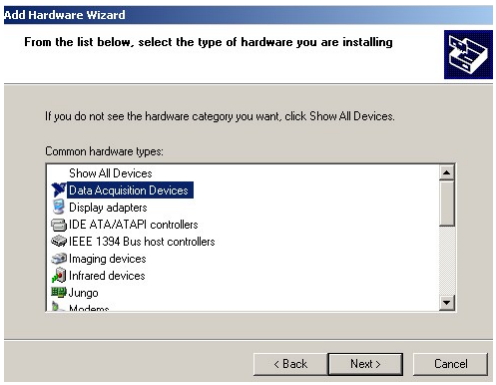
4. Vaya a la ruta donde están los drivers del controlador USB



5. Seleccione el driver e instale el mas conveniente según su sistema operativo.



6. Después de instalado el dispositivo éste debe aparecer en los drivers que están cargados en el sistema y Windows muestra un mensaje de “hardware configurado satisfactoriamente” y con el puerto virtual COM # configurado. “**Serial <=>USB”**



Anexo B. Planos generales de la tarjeta ADQ USB 2.0

ESQUEMATICO DEL MODULO DE COMUNICACIONES USB
TARJETA ADQ USB 2.0

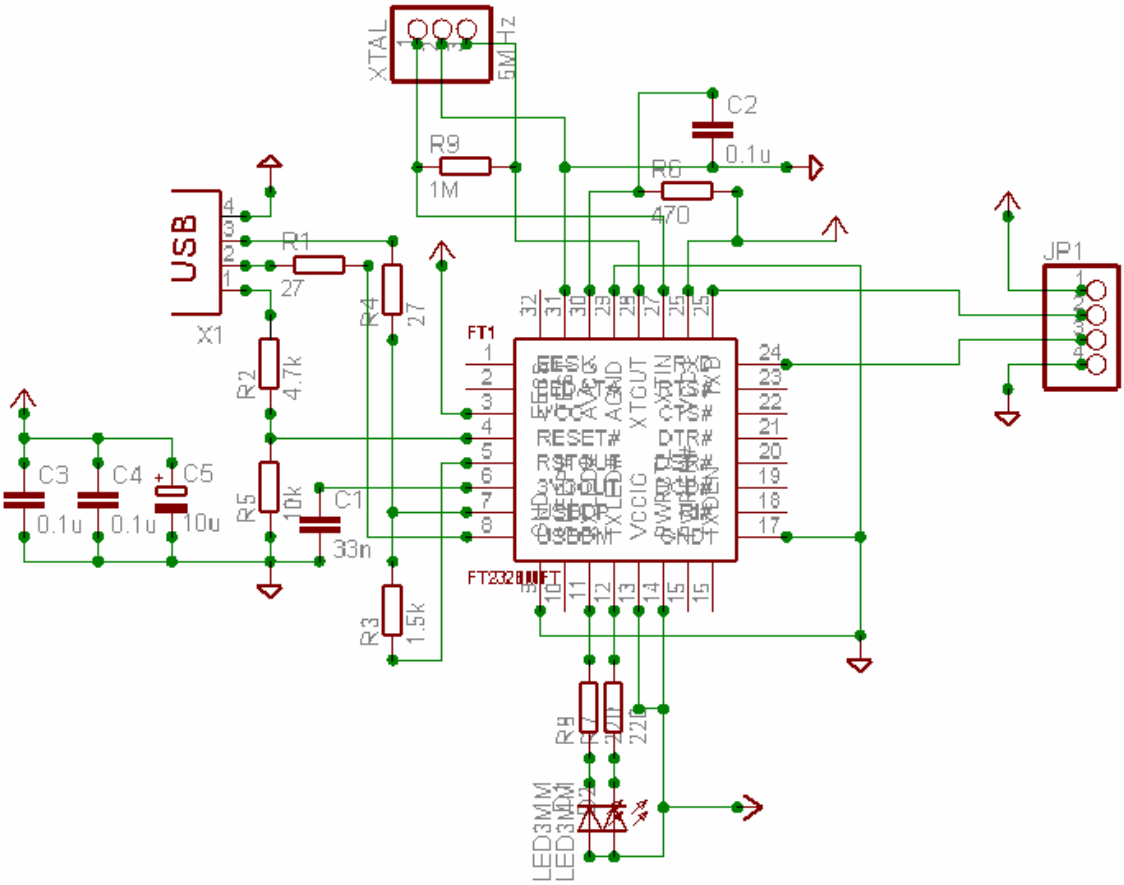
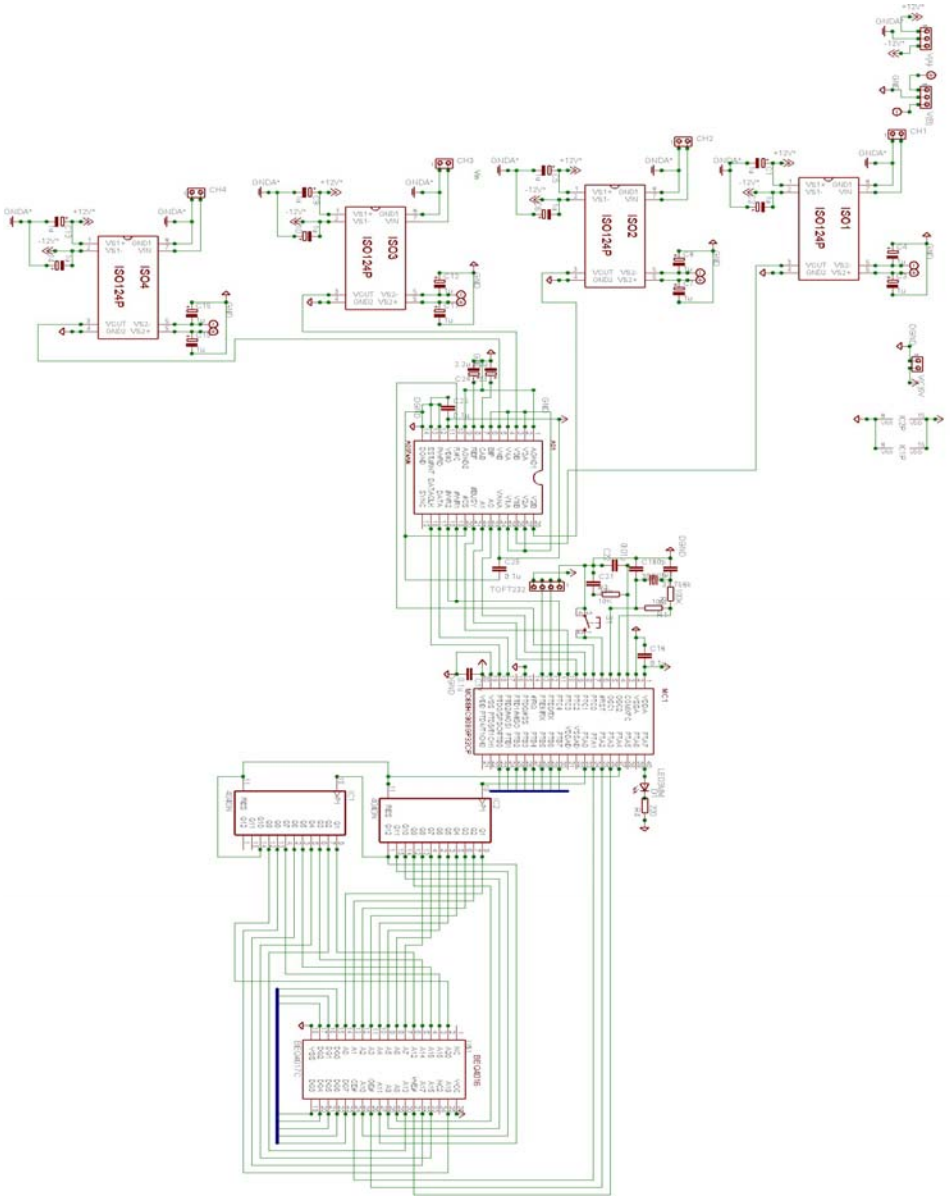
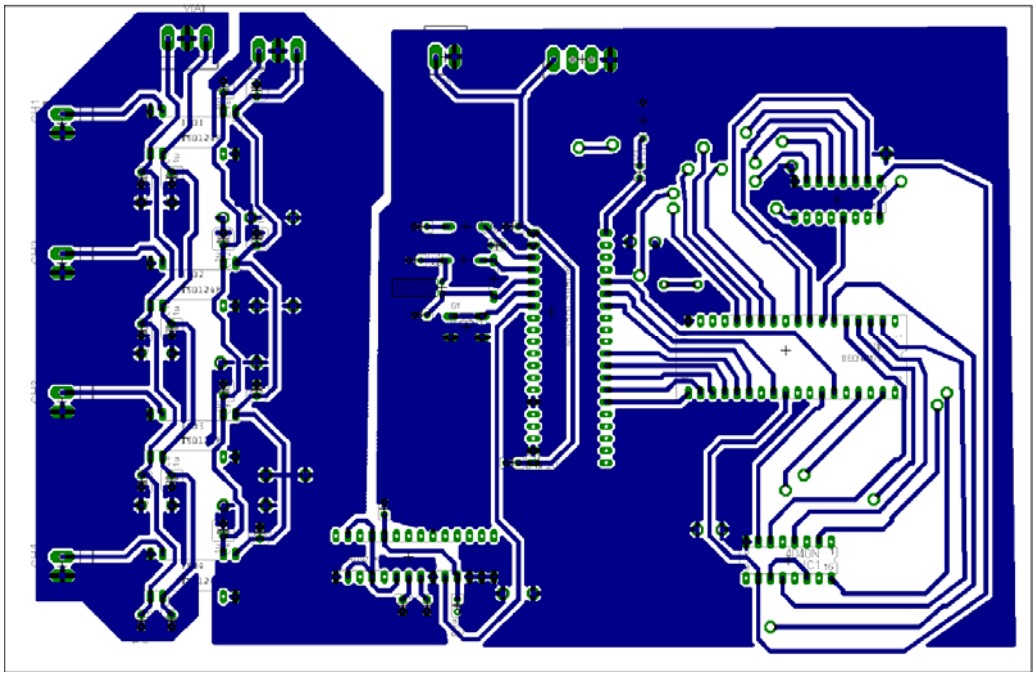


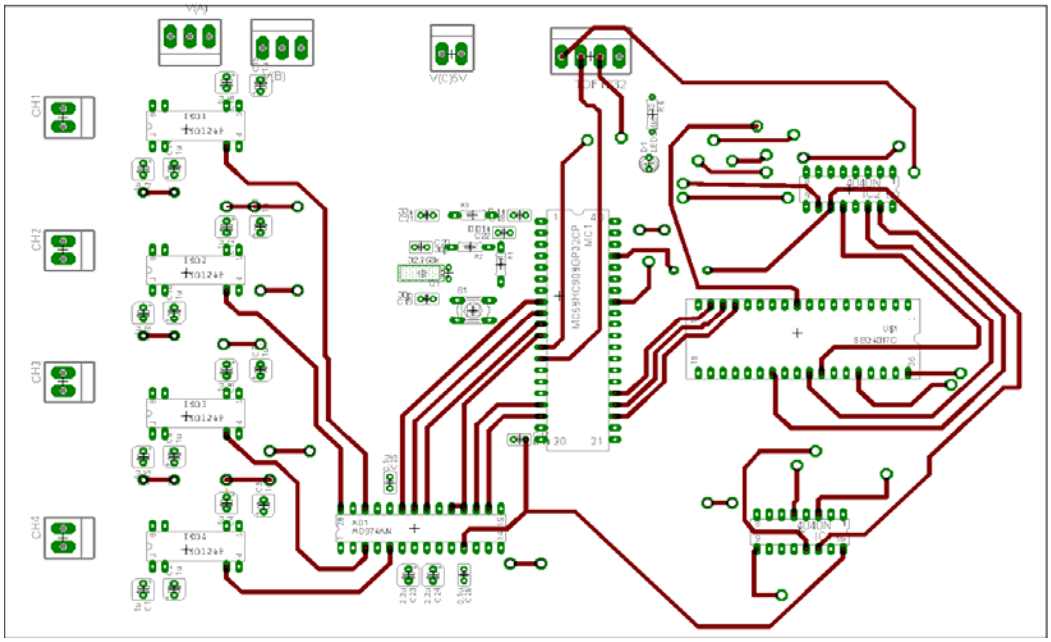
DIAGRAMA ESQUEMATICO MODULO ANALÓGICO Y DE CONTROL SISTEMA DE ADQUISICIÓN DE DATOS USB2.0



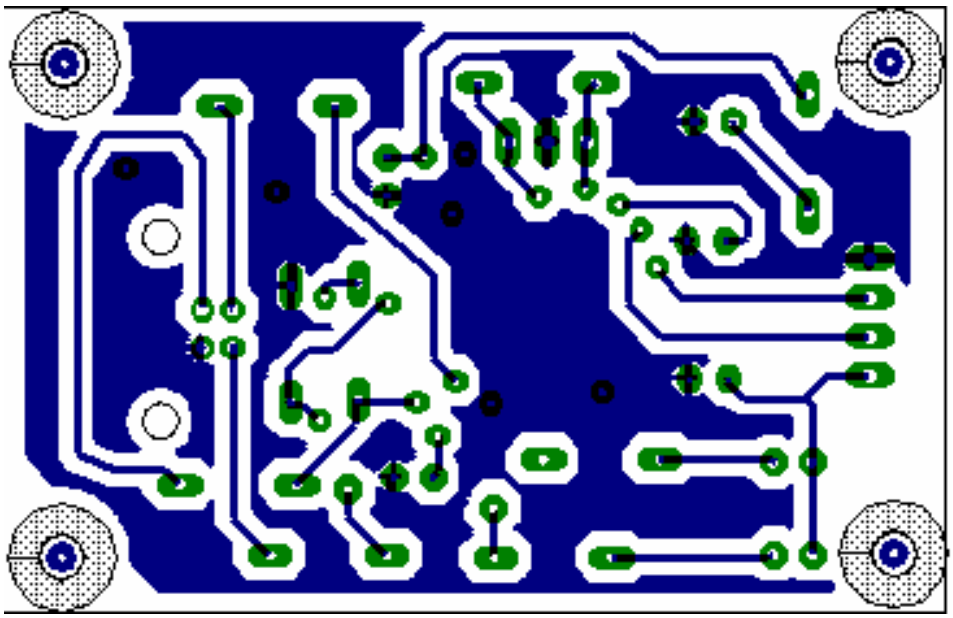
BOARD BOTTOM MODULO ANALÓGICO Y DE CONTROL SISTEMA DE ADQUISICIÓN DE DATOS USB2.0



BOARD TOP MODULO ANALÓGICO Y DE CONTROL SISTEMA DE ADQUISICIÓN DE DATOS USB2.0



BOARD BOTTOM MODULO DE COMUNICACIONES USB TARJETA ADQ USB 2.0



BOARD TOP MODULO DE COMUNICACIONES USB TARJETA ADQ USB 2.0