Universidad ORT Uruguay

Facultad de Ingeniería

Osciloscopio USB

Entregado como requisito para la obtención del título de:
Ingeniero en Electrónica

Pablo Hoffman – 139082
Martín Szmulewicz – 139123

Tutor: Ing. Claudio Misail

2006

Índice

Abstract

El presente documento es un informe detallado sobre la planificación, seguimiento, y desarrollo, del proyecto de fin de carrera de los alumnos Pablo Hoffman y Martín Szmulewicz.

El proyecto consta del diseño y construcción del prototipo de un osciloscopio de prestaciones típicas para laboratorio. A diferencia de un osciloscopio estándar, éste será un dispositivo portátil, y de funcionamiento conjunto con un PC. Es decir que se requiere de un PC para su operación, y a través de él se podrá analizar la información capturada por el equipo, ya sea visualmente o almacenada en algún dispositivo. Todos los controles y funciones necesarias se realizarán desde el ordenador.

El documento contiene toda la información recopilada y los conocimientos adquiridos por los estudiantes durante la elaboración del proyecto. Se incluyen apuntes, decisiones de diseño y conclusiones, sobre la evolución y progreso del proyecto.

Esta documentación también se encuentra publicada en Internet en la siguiente dirección:

http://pablohoffman.com/oscusb/doc/

Todo el resto de la información del proyecto se encuentra disponible en:

http://pablohoffman.com/oscusb/

Distribución de los capítulos

Esta documentación se encuentra distribuida en 12 capítulos y 4 apéndices. Ellos son:

Capítulo 1. Introducción

Este capítulo realiza una introducción del proyecto presentando los objetivos y aspiraciones del mismo. También cuenta como fue el proceso de planificación y las herramientas utilizadas para llevarlo adelante.

Capítulo 2. Investigación y análisis

Este capítulo muestra un estudio de las tecnologías existentes en el mercado en el momento de comenzar el proyecto para tener una noción de las prestaciones y el precio al cual debíamos apuntar. También se han investigado diferentes marcos teóricos, análisis de métodos de muestreo, y calculo de margenes de error.

Capítulo 3. Bus serie universal (USB)

Este capítulo trata sobre la estándar USB y su funcionamiento, con especial énfasis en los temas relacionados directamente con la implementación del proyecto.

Capítulo 4. Hardware

Este capítulo cubre todos los temas relacionados con el diseño del hardware del osciloscopio, desde la selección de componentes hasta la programación y el depurador por hardware.

Capítulo 5. Firmware

Este capítulo trata sobre los temas específicos relacionados con la implementación del firmware, que es el programa que corre en el PIC y controla el funcionamiento de la placa. Incluye información sobre la estructura del código e información sobre detalles particulares de la programación en el lenguaje C del PIC.

Este capítulo trata sobre la implementación del software del lado de la PC. Incluye temas como la selección de la plataforma de programación, y métodos de programa utilizados en el código necesarios para lograr una aplicación gráfica eficiente.

Capítulo 7. Protocolo de comunicación

Este capitulo contiene la especificación del protocolo de comunicación junto con las políticas adoptadas a la hora de diseñarlo.

Capítulo 8. Fabricación y puesta en marcha

Este capítulo habla sobre temas relacionados con la implementación del hardware, incluyendo las compras de componentes y la fabricación y puesta en funcionamiento de la placa.

Capítulo 9. Manual de usuario

Este capítulo es una guía del funcionamiento del software de la PC con el cual se controla el osciloscopio. Está orientado al usuario final del osciloscopio.

Capítulo 10. Temas pendientes y mejoras posibles

Este capítulo es una revisión de los temas que quedaron pospuestos para futuras mejoras del osciloscopio, especificando, en algunos casos, su prioridad.

Capítulo 11. Autoevaluación y conclusiones

Este capítulo es una evaluación final del trabajo y las lecciones que hemos aprendido durante el año de trabajo en el proyecto. Incluye una evaluación de costos y rentabilidad, y también una conclusión final.

Apéndice I: Esquemáticos

Este apéndice contiene los esquemáticos completos del osciloscopio.

Apéndice II: Lista de materiales (BOM)

Este apéndice contiene la BOM del osciloscopio.

Apéndice III: Hojas de datos

Este apéndice tiene un link a las hojas de datos de todos los componentes utilizados mencionados en la documentación. Por razones de espacio, solo se publican los links a dichas hojas de datos.

Apéndice IV: Glosario

Este apéndice contiene una lista de términos técnicos utilizados a lo largo de la documentación y su significado.

Apéndice V: Especificaciones

Este apéndice contiene una lista con todas las especificaciones del osciloscopio basados en la implementación final del prototipo. Para cada especificación se comenta sobre el factor o componente limitante de la misma.

Apéndice VI: Lista de figuras

Este apéndice contiene una lista de todas las Figuras incluidas en el texto.

Apéndice VII: Lista de tablas

Este apéndice contiene una lista de todas las tablas incluidas en el texto.

Capítulo 1. Introducción

Contenido de este capítulo

Objetivos

El objetivo del proyecto es el diseño y construcción de un prototipo de un osciloscopio USB para PC que sea lo suficientemente robusto y confiable como para poder comercializarlo a nivel educativo (liceos, facultades, escuelas técnicas, etc).

El osciloscopio competirá en prestaciones contra un osciloscopio típico de laboratorio universitario pero con un costo menor. Esto permitirá su fácil inserción en esta área, e incluso extender su uso a otros niveles educativos como la educación secundaria. No obstante, por su bajo costo, el producto también será atractivo para entusiastas en electrónica ya que el mismo tendrá una relación costo/beneficio muy buena.

Prestaciones básicas

El osciloscopio deberá poseer todos los requisitos básicos para que pueda ser usado con comodidad, es decir: protección contra sobrevoltajes, manejo confortable (buena interfaz de control), y otras características similares. Si tenemos en cuenta que hoy en día un osciloscopio analógico típico en Uruguay (Tektronix 2205 / 20 Mhz) está en el entorno de los U$S 600 o más, la meta es lograr diseñar y construir un dispositivo que pueda ser comercializado a un precio menor pero con características similares o, en algunos casos, superiores, para compensar la gran robustez de un osciloscopio de marca.

Las características que deseamos obtener son:

- Ancho de banda mayor a 10Mhz.
- Gran rango de tensiones de entrada.

- Protección de entrada.
- Ser de fácil uso.
- Ser pequeño, compacto, transportable.

Prestaciones adicionales

Además de cumplir con los requisitos básicos, nuestro osciloscopio contará con algunas prestaciones adicionales derivadas de su naturaleza digital, como ser:

- Capacidad de detección de transitorios por software.
- Capacidad de procesar digitalmente los resultados, ya sea guardar imagen de las capturas o planillas con los valores capturados para luego ser analizados, o bien trabajar sobre ellos en alguna aplicación externa.
- Capacidad de realizar un análisis de espectro.

Como características adicionales que deseamos, pero que no necesariamente provienen de su naturaleza digital son:

- Bajo consumo, no requerir una fuente externa, solo la alimentación del USB (esto depende exclusivamente del diseño del hardware).
- Multiplataforma (que pueda funcionar en cualquier PC con puerto USB, sin importar su sistema operativo).
- Actualizable a través del puerto USB, de procedimiento sencillo.

Diseño y realización

La idea sobre la cual trabajamos fue la de realizar el equipo físico lo mas sencillo posible, volcando hacia el lado del PC la mayor parte del procesamiento. Es decir, el hardware solamente digitalizaría los datos y los enviaría al ordenador, luego el PC es quién se encargará de analizar los datos, realizar cuentas, etc. El hardware debería ser capaz de recibir ordenes y ejecutarlas, pero sin tener que realizar demasiada lógica o análisis, como así también tendría que ser capaz de realizar operaciones de seteo, como la velocidad de adquisición o la selección del rango de entrada.

Lo que se ha implementado a nivel hardware es un diseño basado en un microcontrolador central, quien es el encargado de realizar el control sobre todos los componentes del equipo. Podemos destacar los siguientes bloques:

- Etapa de entrada
- Digitalización
- Memoria
- Contadores

El mismo microcontrolador ya posee integrado en él los componentes para la comunicación vía USB.

Básicamente el microcontrolador, recibe las siguientes instrucciones:

- Ajustes
    - Rango de entrada
    - Selección de canales
- Operación
    - Captura (alta, media, y baja velocidad)

Al recibir una orden de captura se dispara el proceso, digitalizando la señal analógica de entrada y guardando los valores binarios en la memoria. Una vez que se completa la operación de almacenamiento, es decir, cuando la memoria ya esta llena, se comienza con la transmisión de los datos hacia el ordenador. Cabe destacar que esto es valido para la captura de alta y media velocidad, donde la digitalización se realiza a mucha mayor velocidad que la transferencia. Para el modo de captura a baja velocidad, cada dato digitalizado es transmitido inmediatamente al ordenador sin ser almacenado en la memoria. Esto sólo es posible en este caso dada la baja velocidad de digitalización y las limitaciones en velocidad del canal USB.

El software trata de ser lo mas similar posible a un osciloscopio estándar, permitiendo una operación sencilla e intuitiva. Éste posee controles de velocidad de barrido y rango de tensiones de entrada, con la posibilidad de seleccionar cualquiera de los dos canales independientemente o simultáneamente. Basándose en la selección de la velocidad de barrido, automáticamente elije el tipo de captura a solicitar, haciendo aún mas sencilla la tarea del microprocesador. Así también, cada vez que se modifica la selección de canales o el rango de tensiones, se envía el comando correspondiente al equipo.

A continuación se muestra un diagrama de bloques del equipo y una pantalla de muestra del programa que debe correr en el ordenador:

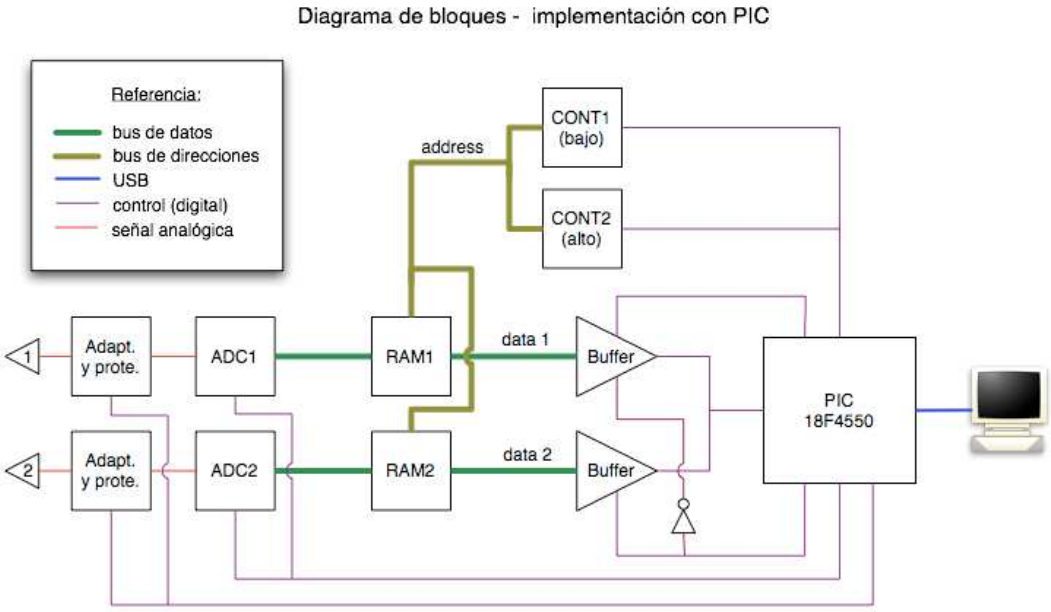


Fig 1.1 Diagrama de bloques del osciloscopio

Si bien se ha logrado que este prototipo sea una realidad y funcione de forma correcta, hay ciertos puntos que aún no han podido ser completados. Cabe destacar que la etapa de entrada del equipo no contiene protección ni aislación alguna, de modo que si la señal que se esta midiendo sobrepasa los limites de operación, el equipo puede ser dañado seriamente. Tampoco cuenta aún con una aislación hacia el lado del PC, de modo que de ocurrir algo de esta naturaleza, el PC también se encuentra en riesgo. Dentro de esta documentación se encuentran todos los aspectos que deben mejorarse, sin embargo es importante destacar las recién mencionadas ya que están relacionadas con la seguridad de los equipos y las personas. Debemos aclarar que esto se debe a qué se ha logrado fabricar este prototipo en poco tiempo y es por ello que ciertas Características deseables o incluso necesarias, quedan como pendientes para el futuro.

Planificación

Las primeras dos semanas del proyecto fueron dedicadas exclusivamente a la planificación del mismo. En estas semanas fijamos las pautas generales a seguir, definimos las etapas a cubrir en el proyecto, objetivos, plazos y expectativas. Además, escogimos la herramienta a utilizar para realizar el seguimiento y trabajar sobre la documentación.

Herramientas de trabajo

Wiki

Como primera instancia elegimos una herramienta a utilizar para realizar el seguimiento del proyecto, así como para trabajar conjuntamente. Dicha herramienta debía tener las siguientes especificaciones:

- permitir la edición conjunta del contenido guardando un registro de cambios para tener un seguimiento de los mismos
- tener un sistema de notificación por e-mail cada vez que se realiza algún cambio
- servir como herramienta para escribir la documentación de forma colaborativa
- permitir llevar una lista de las tareas pendientes y un registro de las tareas terminadas
- tener accesibilidad global sin restricciones de plataforma (Windows, Linux, etc) o lugar de trabajo (por ejemplo, tener que estar en la ORT para usarla)

Luego de evaluar estas especificaciones encontramos que la herramienta que se ajustaba a nuestras necesidades era una herramienta wiki. El uso de herramientas *wiki* han crecido mucho últimamente en popularidad gracias a proyectos como la Wikipedia (http://www.wikipedia.org) que hacen uso de ellas. Básicamente, una aplicación Wiki permite tener un sitio web cuyas páginas son, a su vez, editables por los mismos usuarios que las acceden.

Esto permite generar documentación y trabajar sobre la información de forma colaborativa, editando conjuntamente las páginas sin que ocurran conflictos de edición ya que todos los cambios quedan registrados. Por lo tanto, al realizarse algún cambio los involucrados en el proyecto reciben una notificación por mail de la página que cambió e inmediatamente puede consultar que parte fue la que cambió. Esto brinda una gran flexibilidad a la hora de trabajar con la información de forma remota.

Por consiguiente, esta fue la herramienta principal que escogimos para trabajar con el proyecto. La aplicación se llama **TWiki** y fue utilizada para realizar las siguientes tareas:

- bitácora de reuniones con el tutor
- apuntes y seguimiento de las investigaciones
- registro de mails importantes relacionados con el proyecto
- repositorio de documentos e información administrativa concerniente al proyecto
- tormenta de ideas para el diseño
- desarrollo de documentos a entregar (este documento esta siendo fue escrito en TWiki)
- seguimiento del progreso del proyecto con la posibilidad de ver cuales fueron los últimos documentos o secciones que cambiaron con notificaciones vía email
- al ser web automáticamente nos permite tener una web del proyecto sin ningún esfuerzo adicional (acceso público solo lectura).
- almacenamiento de información administrativa (carta del proyecto, pautas, etc)
- glosario de términos relacionados con el proyecto

A continuación se listan, a modo de ejemplo, algunas de las páginas pertenecientes al TWiki de nuestro proyecto.

- http://pablohoffman.com/cgi-bin/twiki/bin/view/Oscusb/BitacoraDeInvestigacion
    - aquí fuimos anotando información que íbamos recabando, en la etapa de investigación y análisis
- http://pablohoffman.com/cgi-bin/twiki/bin/view/Oscusb/BrainstormingDiseno
    - utilizada en los comienzos de la etapa de diseño para tirar ideas (tormenta de ideas)
- http://pablohoffman.com/cgi-bin/twiki/bin/view/Oscusb/TareasPendientes
    - aquí llevamos, en todo momento, un registro de las tareas pendiente junto con la persona que la tenía asignada
- http://pablohoffman.com/cgi-bin/twiki/bin/view/Oscusb/BitacoraDeInvestigacion
    - lo utilizamos al principio para llevar registro de las reuniones con el tutor junto con los temas discutidos en la misma
- http://pablohoffman.com/cgi-bin/twiki/bin/view/Oscusb/OscusbEsquematicos

    - aquí publicamos los esquemáticos (fuentes y PDF) cada vez que sacamos una nueva revisión, junto con los cambios realizados en esa revisión.
- http://pablohoffman.com/cgi-bin/twiki/bin/view/Oscusb/ComprasPedidos
    - aqui llevamos un registro de las compras pendientes y completadas, asi como el saldo diponsible

Dichas páginas pueden ser accedidas libremente desde Internet. Desde el comienzo del proyecto el sitio estuvo abierto a cualquier persona interesada. Esto nos pareció una idea muy interesante para explotar desde el principio, no solo para promocionar el proyecto sino también para recibir ideas y comentarios de otras personas.

La siguientes capturas de pantalla muestran distintos páginas de TWiki.

Fig 1.2 Página principal de TWiki (sitio web del proyecto)

Por último, vale mencionar que TWiki es una herramienta de software libre cuyo código es abierto y está disponible para bajar en http://twiki.org.

CAD electrónico

La segunda herramienta que tuvimos que seleccionar fue el CAD para diseñar y dibujar los esquemáticos. Al igual que el Wiki, la elección de un CAD adecuado nos parece una tarea crucial para asegurar el desarrollo óptimo del proyecto, por lo cual decidimos dedicar un tiempo considerable a evaluar alternativas.

Los puntos que consideramos importantes a la hora de evaluar fueron:

- que la interfaz de uso (para crear pistas, mover e interconectar componentes, buses, etc) del programa fuera ágil e intuitiva
- que tuviera una gran librería de partes y un buen editor de componentes
- que tuviera un simulador potente
- que tuviera la posibilidad de encapsular sub-circuitos como bloques tipo caja negra

Los programas que evaluamos fueron los siguientes:

1. Circuit Maker 2000
2. CadSoft Eagle
3. Orcad 10
4. Electronics Workbench Multisim 8

Los resultados de dicha evaluación se detallan a continuación:

1. Circuit Maker 2000

El primer CAD que probamos fue el Circuit Maker 2000, ya que es el CAD electrónico oficial de la ORT. Si bien la interfaz en si es bastante sencilla de usar, el simulador que trae es muy pobre. Otra carencia que le encontramos fue la reducida librería de componentes y una interfaz no muy cómoda para diseñar nuevos componentes. Todas estas razones nos llevaron a descartar el Cicuit Maker como CAD de diseño.

2. CadSoft Eagle

En segunda instancia probamos el Eagle de CadSoft. Este CAD es muy popular entre entusiastas y aficionados ya que posee dos grandes ventajas:

- es gratuito para usos sin fines de lucro (como el nuestro)
- hay versiones para Windows, Linux y Mac OS X

Las características de ser multiplataforma y gratuito le han hecho ganar mucha popularidad en el mercado. Sin embargo (y a diferencia de los otros) no tiene simulador, razón por la cual optamos por descartarlo. Si bien al principio no sabíamos si íbamos a utilizar el simulador, consideramos importante tener prevista esta necesidad a priori.

3. Orcad 10

El Orcad nos pareció un CAD extremadamente potente (incluso más que el Multisim) y con una muy buena librería de componentes. Sin embargo, el gran problema que le encontramos fue su interfaz extremadamente compleja y algunas carencias de la misma como la de la conexión rápida de buses.

4. Electronics Workbench Multisim 8

El Multisim fue el programa que decidimos utilizar como CAD electrónico pues contaba con las siguientes características carentes en todas (o algunas) de las demás alternativas:

- capacidad de trazar e interconectar rápidamente buses (muy necesario para nuestros esquemáticos)
- interfaz sencilla e intuitiva de manejo, colocación de componentes y conexión de los mismos
- poderoso y cómodo editor de partes para poder crear componentes de forma rápida, completa y concisa
- buena librería de componentes
- muy buen simulador: potente y fácil de usar

Cronograma

En las primeras dos semanas de planificación se definieron las etapas del proyecto junto con el cronograma a seguir, de fue el siguiente:

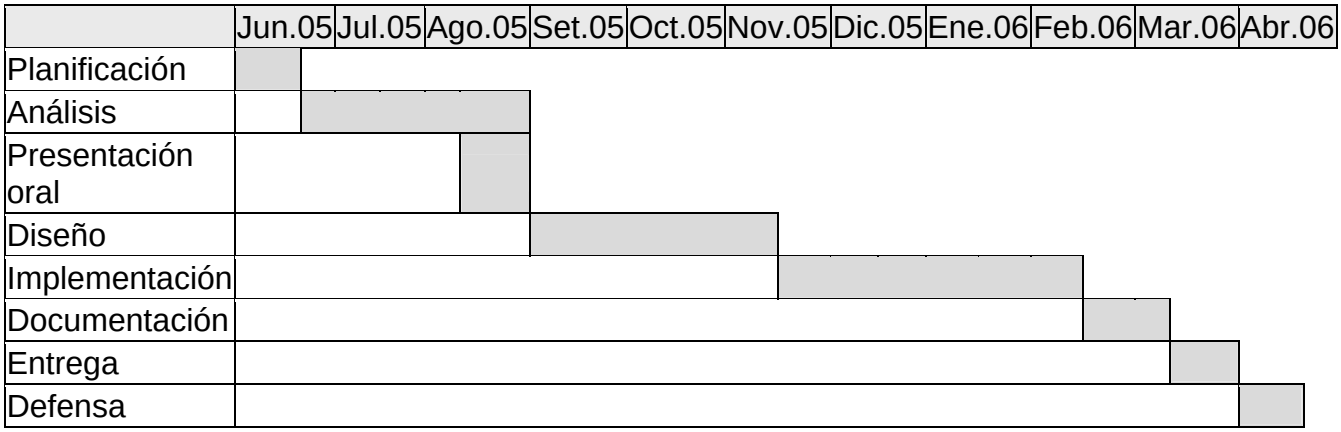


Fig 1.3 Diagrama de Gantt del proyecto

Del cronograma se puede distinguir 3 grandes etapas: análisis, diseño e implementación que son las etapas donde ocurriría realmente el desarrollo del proyecto, y otras etapas administrativas como la presentación, entrega y defensa.

Afortunadamente, hoy podemos afirmar que el cronograma fue seguido muy rigurosamente, practicamente sin desvíos respecto a las plazos propuestos inicialmente, con la excepción de que la etapa de diseño se solapó con la de implementación, aunque esto también lo habíamos previsto (y fue comentado en el documento de la presentación).

Referencias

- TWiki (herramienta wiki para trabajar en el proyecto)
  - http://www.twiki.org
- Microchip MPLAB (programador y depurar del PIC18F4550)
  - http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en019469&part=SW007002

- Microchip MPLAB C18
  - http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en010014
- Electronics Workbench Multisim
  - http://www.electronicsworkbench.com/
- Circuit Maker
  - http://www.microcode.com/
- Orcad
  - http://www.orcad.com/
- CadSoft Eagle
  - http://www.cadsoft.de/

Capítulo 2. Investigación y análisis

Contenido de este capítulo

Tecnologías y productos existentes

Una parte importante de la investigación fue el estudio de las tecnologías y productos existentes en el mercado, junto con sus especificaciones y precios. Puesto que que, si bien el objetivo de nuestro osciloscopio no es competir en un mercado comercial (sino a nivel académico), es importante conocer el mercado de los mismos para saber donde estábamos parados, y a que precio final debíamos apuntar para tenerlo en cuenta a la hora de calcular costos y rentabilidad.

Una carencia que padecía la mayoría, era que no traían software para correr en Linux, algo que consideramos básico para un ambiente académico, por lo cual que el software fuera multiplataforma fue una de las metas desde el principio.

A continuación se listan algunos de los productos que encontramos disponibles en Internet. La lista fue compilada en abril de 2005 y por lo tanto no contempla

modelos que hayan salido posteriormente. Todos los precios están en dólares y no incluyen impuestos. No pudimos encontrar ninguno de estos osciloscopios en el mercado local.

Pico Technology

Pico Technology es una empresa que se fundó en 1991 e inmediatamente obtuvo una gran popularidad por sus osciloscopio y capturadores de datos para PC.

Tiene la línea productos PicoScope 3000 donde el modelo más económico mide señales de hasta 50 Mhz (2 canales) y cuesta U$S 760. Tiene un software propietario para Windows y drivers para C, Pascal, Delphi y Visual Basic.

También tiene un modelo que mida hasta 200 Mhz por U$S 1530.

TiePie Engineering

TiePie Engineering es una empresa holandesa que desarrolla y vende instrumento de medida controlados por computadora. Siempre ha trabajado en el mismo rubro desde su fundación, en el año 1987.

Algunos de los modelos que ofrecen son el Handyscope 4 (HS4) de 50 Mhz, 12 bit y 4 canales con comunicación USB 2.0. por U$S 765, y el Handyscope 2 (HS2) de 200 kHz y 2 canales por U$S 500.

Tanto el software como el driver viene exclusivamente para Windows.

ETC

ETC es una empresa fundada en 1996 y dedicada al diseño y producción de dispositivos de medida para PC, utilizando FPGAs y CPLDs.

Uno de sus productos es el M521 de 60 Mhz y 2 canales por U$S 550, mientras que el M524 de 120 Mhz cuesta U$S 780. El software es para Windows y tiene disponibles drivers para Delphi y Visual Basic con un costo adicional.

Bitscope

Bitscope es una empresa australiana con firmes creencias en el software y hardware libre, y su osciloscopio es quizás el más peculiar de todos puesto que es el único que se caracteriza por ser de hardware abierto, lo cual significa que en su página se encuentran publicados todos los esquemáticos del mismo, junto con abundante documentación sobre su construcción y decisiones de diseño, como así también protocolos, lo cual implica que uno mismo puede construirse el mismo osciloscopio que ellos venden, si tiene los medio apropiados y conocimientos necesario. Como fieles adeptos al software libre

encontramos esa postura fascinante y no nos avergüenza confesar que muchas ideas para nuestro osciloscopio fueron tomadas de allí.

Uno de sus osciloscopios es el BS300 de 75 Mhz y 2 canales por U$S 495. El software, por supuesto, corre tanto en Windows como en linux.

Áreas de diseño

Una de las tareas realizadas en la etapa de análisis fue la de definir las áreas de diseño que tendríamos que abordar. Finalmente llegamos a la decisión de que el diseño se dividiría en 3 grandes etapas:

1. Diseño del hardware analógico
2. Diseño del hardware digital
3. Diseño del software

A su vez cada una de ellas se divide en pequeñas sub-etapas de: estudio, diseño y realización. En otras palabras, encaramos cada área de diseño como un pequeño proyecto en sí mismo. Las áreas tampoco son temas de estudio independientes puesto que todas están relacionadas entre si.

La etapa más importante (y la que tuvimos que definir primero) fue la del Hardware digital puesto que allí se definía la arquitectura del sistema y de ella dependía todo el resto del diseño, como ser procesador, placa y entorno de desarrollo, etc.

Diseño del hardware analógico

Esta etapa consistiría del diseño y especificación de:

- componentes a usar en la etapa de entrada para adaptación y protección de la misma
- protección de transitorios del puerto USB
- estudio del consumo de potencia y la necesidad de utilizar un regulador (7805 o similar)

Diseño del hardware digital

Esta etapa consistiría en:

- selección de la arquitectura a usar
- diseño y especificación de todos lo componentes digitales a utilizar
- especificar la interconexión de los componentes (esquemáticos)

Diseño del software

Esta etapa (posiblemente la más larga) consistiría en:

- diseño del firmware (software a correr en el PIC), algoritmos de captura, etc
- diseño del software a correr del lado de la PC para controlar y manejar el osciloscopio
- especificación del protocolo de comunicación a usar entre el firmware y el software

Bitácora de investigación

La bitácora de investigación es una sección en donde hemos colocado un resumen de todo lo que hemos investigado en cuanto a tecnologías y diseño. Aquí se encuentra todo lo relacionado a los diferentes tipos de muestreo, tipos de memoria, consejos que nos han dado ingenieros, etc.

Este diario tiene relación con lo que sería un *brainstorming*, pero de investigación. Allí se colocaron referencias y pequeños resúmenes de todo lo que se fue investigando previo a la etapa de diseño.

Toda esta información se encuentra disponible en la siguiente dirección:

http://pablohoffman.com/cgi-bin/twiki/bin/view/Oscusb/BitacoraDeInvestigacion

A continuación presentamos una selección de temas allí investigados, y que no hemos tratado en otras secciones.

Mecanismos de disparo

En los osciloscopio existen 3 tipos de mecanismos de disparo (trigger) según la finalidad con la que se utilice. Estos son:

- **Mecanismo de disparo básico**
  - Este es el trigger que usan los osciloscopios analógicos para poder mantener fija la imágenes que se muestra en la pantalla.
- **Mecanismo de disparo por detección de transitorios**
  - Utilizado por los osciloscopios digitales (con memoria) para capturar eventos anómalos de una señal y desplegar la forma de la señal en el momento en que estos ocurren.
- **Disparo externo**
  - Este es el mecanismo de disparo que permite observar lo que ocurre en una de las puntas del osciloscopio cuando llega un disparo (pulso, transitorio, etc) en la otra punta (de allí el nombre *disparo externo*).

Métodos de muestreo

Tiempo real

- Método ideal para 2fs < fm (fs = frec. señal , fm = frec max del ADC)
- única forma de capturar transitorios

- interpolación:
    - lineal: unir puntos con líneas
    - senoidal - sin(x)/x - preferido para 3 fs < fm < 5fs

Este método de muestreo es el mas simple de todos y permite digitalizar señales no periódicas y transitorias.

Cada muestra y el tiempo en que fue tomada tiene una correspondencia directa con su equivalente en tiempo real.

A mayor tasa de muestreo en comparación al ancho de banda de la señal, se obtiene una mayor definición en el resultado.

Nyquist desarrolló un teorema que dice que para reconstruir una señal de frecuencia Fm, se debe muestrear a un índice mayor a 2Fm. Sin embargo, esto no se aplica tan directamente como parece. Esta teoría se aplica solamente a señales de ancho de banda limitado que no contienen ningún componente sobre la frecuencia Fm, y los bordes rápidos de las señales encontradas en circuitos digitales de alta velocidad pueden contener armónicos significativos sobre sus frecuencias fundamentales. Mas aún, cuando uno muestrea una señal, no lo hace por un tiempo infinito, sino que esta señal se ve acotada en el tiempo. Al recortar esta señal se le agregan componentes de frecuencia mas altas. He aquí entonces el problema de muestrear dichas señales: no se puede muestrear sólo al doble de la frecuencia Fm, sino que hay que hacerlo a una frecuencia mayor. Por lo tanto, no importa cuan rápido se muestree, nunca se podrá recomponer esta señal a la perfección.

Sin embargo, en la práctica se ha encontrado que muestreando a una velocidad cuatro veces mayor que la mayor componente de frecuencia de la señal, se puede obtener un resultado muy confiable y con bajo error.

Si se muestrea a una frecuencia menor a 4Fm pero mayor a 2Fm, obtendremos un resultado mas lejano al ideal, y cuanto mas nos acerquemos al limite de 2Fm, más errores contendrá nuestra reconstrucción. Así mismo, si pasamos por debajo del umbral de 2Fm para la frecuencia de muestreo, el *aliasing* es inevitable.

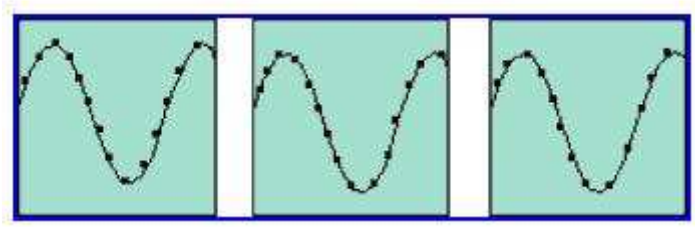


Fig. 2.1 Método de muestreo en tiempo real

- Ventajas:
    - La única opción para una medida correcta de señales no periódicas.
- Desventajas:

- Ancho de banda relacionado directamente con la tasa de muestreo.
- Susceptible al *aliasing* a velocidades de muestreo lentas.

Tiempo equivalente

- única forma para 2fs > fm
- solo para señales periódicas
- la señal se va construyendo en barridos sucesivos
- 3 tipos de barridos:
  - aleatorio repetitivo
  - secuencial
  - submuestreo

Muestreo aleatorio repetitivo

Este tipo de muestreo se utiliza para aumentar la frecuencia máxima de medicion de la señal de entrada.

El muestreo aleatorio repetitivo captura datos sobre la forma de onda adquiriendo puntos en más de una ocurrencia del *trigger*. Esto significa que la forma de onda en sí misma debe ser repetitiva (periódica) y no un acontecimiento transitorio, puesto que la tasa de muestreo en estos casos es típicamente demasiado baja para señales transitivas de alta frecuencia.

En cada ocurrencia del disparador se adquieren algunos puntos de referencia, luego todos los puntos muestreados se juntan en un cuadro compuesto de la forma de onda. Cada punto es puesto en su lugar apropiado midiendo el tiempo transcurrido entre el *trigger* y la propia muestra.

A medida que muestreamos mas ciclos, más muestras obtenemos, y luego son ubicadas en un mismo ciclo pero en la posición correspondiente medida a partir del *trigger*.

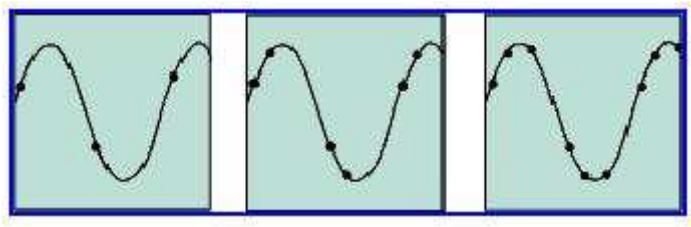


Fig. 2.2 Método de muestreo aleatorio repetitivo

- Ventajas:
  - Ofrece mayor ancho de banda que el muestreo en tiempo real.
  - No es susceptible al *aliasing* en señales repetitivas.
- Desventajas:
  - No es apto para mediciones de señales no periódicas y de alta velocidad.

Muestreo secuencial

Los capturadores digitales de gran ancho de banda tienden a utilizar el muestreo secuencial.

Este método captura una muestra por ciclo (o cada *x* cantidad de ciclos) pero con un determinado tiempo muy exacto entre el disparador y el punto de captura.

Para este tipo de muestreo es necesario, al igual que en el caso anterior, una señal periódica.

Cada muestra es tomada pasado cierto intervalo de tiempo luego de disparado el *trigger*. Para la siguiente captura, el intervalo de espera es incrementado, y por lo tanto dicha muestra va a ser tomada un instante de tiempo después en el ciclo que la muestra anterior.

Los puntos son tomados siempre en puntos diferentes del ciclo de la señal de entrada, sin importar en qué ciclo fue tomada la muestra. De este modo, al finalizar la captura de todas las muestras, cada una de ellas es posicionada en un único ciclo, pero en la posición que le corresponde a partir del tiempo transcurrido desde el *trigger*.

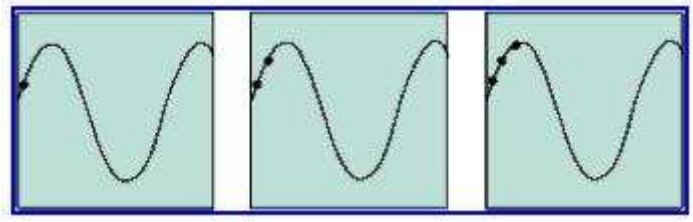


Fig. 2.3 Método de muestreo secuencial

- Ventajas:
  - Ofrece el mayor ancho de banda disponible (hasta 50Ghz).
  - Muy bajo ruido.
  - puede utilizar un A/D lento y de alta precisión.
- Desventajas:
  - No puede capturar eventos previos al *trigger* e incluso pasado un mínimo tiempo tras el.
  - Susceptible al *aliasing* a velocidades de muestreo lentas.
  - No puede medir transitorios.

Submuestreo

Se conoce que el teorema de Nyquist dice que la frecuencia de muestreo tiene que ser mayor al doble del ancho de banda de la señal a muestrear. Sin embargo muchos confunden *ancho de banda* con *frecuencia mas alta*. El error aquí es que una señal contenida en los 100Khz, puede tener un ancho de banda de 1Khz. De este modo, según el teorema de Nyquist bien aplicado, se podría muestrear dicha señal a una frecuencia mayor a 2Khz (la que es mucho menor a 200Khz). La imagen siguiente muestra gráficamente cómo se superponen los espectros de señales de gran ancho de banda muestreadas a

una frecuencia menor a su ancho de banda. Este método de explicación es conocido como el método *fanfold*.

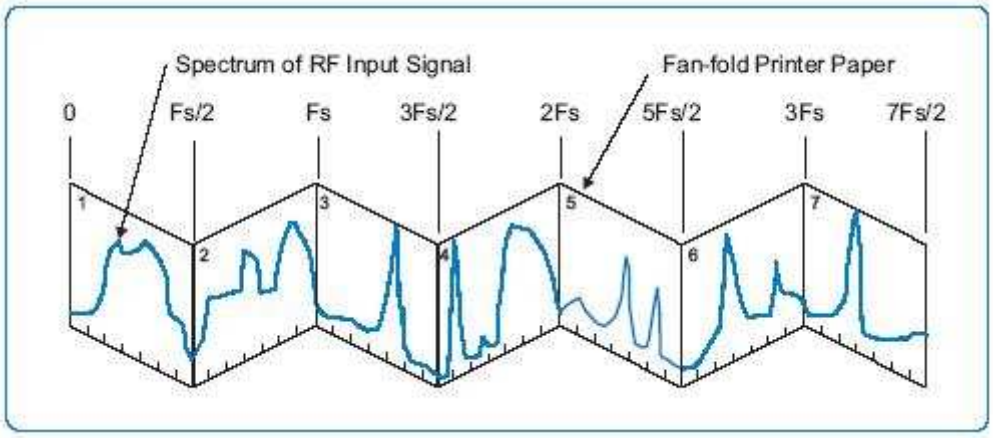


Fig. 2.4 Espectro completo de una señal

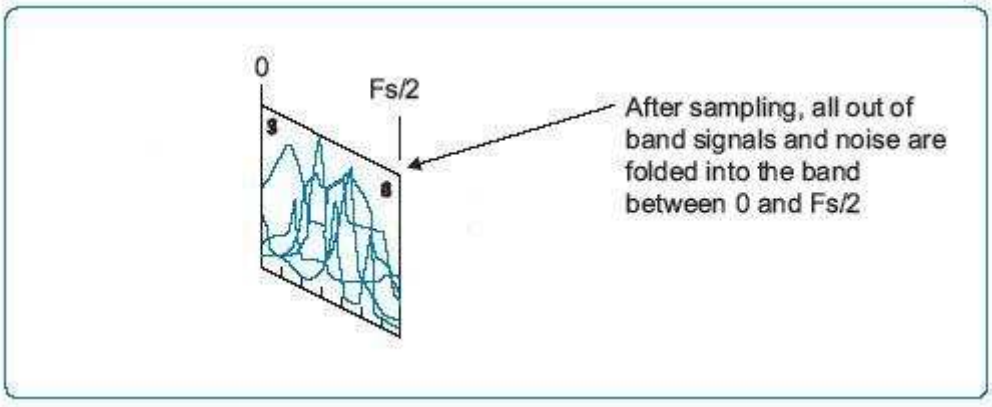


Fig. 2.5 Efecto del *aliasing*

Ahora tomamos por ejemplo el caso antes mencionado, de una señal de ancho de banda acotado, y centrado en una frecuencia alta. A su vez hacemos que pase por filtros para estar seguros de que evitamos el *aliasing*, entonces tenemos un resultado fiable de la señal muestreada.

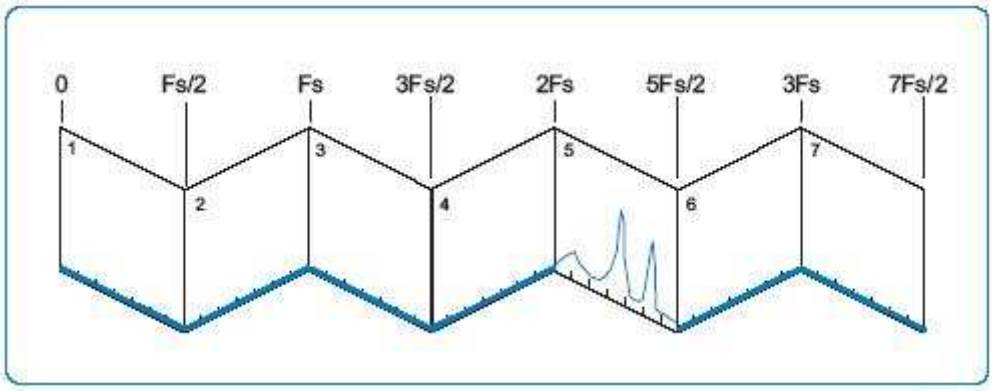


Fig. 2.6 Espectro de la señal filtrada

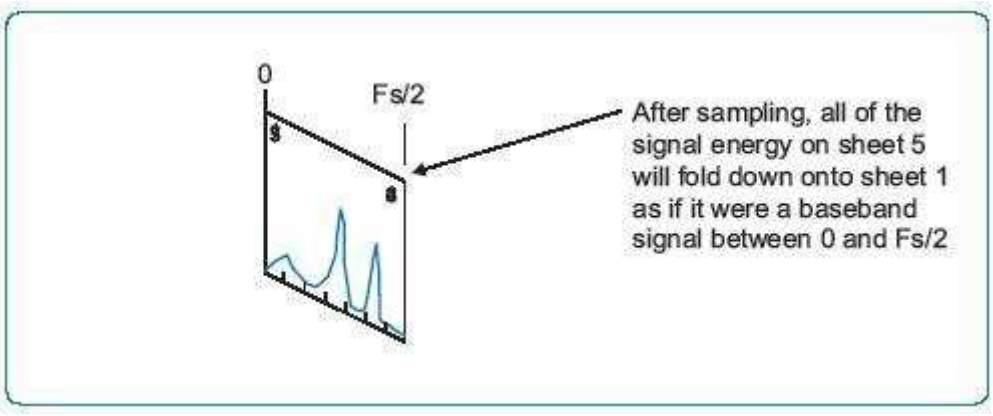


Fig. 2.7 La señal muestreada no es afectada

- Ventajas:
  - Permite medir señales de alta frecuencia con conversores de baja frecuencia.
- Desventajas:
  - La señal debe ser de ancho de banda acotado.
  - Necesidad de filtrado *anti-aliasing*.
  - No apto para medición de grandes anchos de banda.
  - Carece de sentido en señales de banda base.

Métodos de conversión y transferencia

- **Tiempo real**
  - como funciona: la señal se va muestreando y procesando de manera continua, sin interrupciones
  - cuello de botella: tasa de muestreo *(sample rate)* del ADC, y velocidad de transferencia hacia el ordenador.
  - cuando y porque se usa:
    - para **medir transitorios** (es la única forma de poder medirlos)
    - para medir señales lentas (periódicas y no periódicas)
- **Por ráfagas**
  - como funciona:
    1. se toman muestras de la señal rápidamente hasta llenar una buffer
    2. se detiene el proceso de muestreo mientras se procesan (lentamente) los datos
    3. se vuelve a repetir el proceso
  - cuello de botella: velocidad del microprocesador, velocidad transmisión por el bus serie/USB.
  - cuando y porque se usa:
    - se usa cuando el cuello de botella NO está en el ADC sino en otra etapa posterior (velocidad del microprocesador, transmisión placa-PC, etc)
    - se usa para medir señales periódicas rápidas
    - se puede usar también para medir señales muy lentas (tanto periódicas como no periódicas)
    - no se puede usar para medir señales de frecuencia media

Sugerencias del Ing. Juan Pechiar

Agradecemos a Juan Pechiar por habernos aconsejado sobre el curso de nuestro proyecto. Nos ha dado pautas y consejos sobre el equipo y el diseño del mismo. Nuestra reunión con él ha sido de estilo informal, y básicamente la forma de discusión del tema fue a modo de *brainstorming* con ideas sueltas, características, etc. A continuación presentamos un diagrama resumido de los puntos tratados de la charla que mantuvimos con él.

- estabilidad:
    - tolerancia de los componentes
    - corrientes de fuga
    - temperatura
    - cristal (frecuencia)
    - linealidad de filtros
    - ejemplo de un osciloscopio "bueno"
        - tienen los cristales en un *horno*
        - al encender hay una demora hasta que el hornito se estabiliza a una temp (por ejemplo, 60ºC)
        - manteniendo estable la temp de los cristales, funciona el osc.
- comento que el acondicionamiento de la señal era una parte importante y complicada
- funcionamiento de un muestreador secuencial (sequential sampling):
    - para señales muy rápidas y **repetitivas**
    - se toma un set de muestras justo cuando ocurre el trigger
    - al siguiente trigger se deja un delay (por ejemplo, un clock) y se toma un nuevo set de muestras
    - luego de hacer esto varias veces, se termina de reconstruir la señal
- el submuestreo es un mal recurso:
    - la señal de entrada tiene que ser de banda angosta, señales conocidas
    - o bien, filtros muy buenos y complejos a la entrada para acondicionar el espectro
    - para señales desconocidas no es recomendable
- sugerencias:
    - ADC muy rápido
    - PIC, FPGA, microcontrolador
    - memoria
    - interfaz con PC y procedimientos:
        - se espera al trigger (analógico o digital)
        - se toman las muestras
        - se procesan
        - van a memoria
        - lentamente se pasan al PC
        - cuando termino, recién ahí espero un nuevo trigger

Al momento del diseño tuvimos mucha consideración sobre lo que él nos ha sugerido. Se trata de un reconocido ingeniero a nivel nacional, y con vasta

experiencia en estos temas. Se puede observar que el diseño final del equipo tiene muchos parecidos con las puntos expuestos por Pechiar.

Errores de cuantización

Estos errores son parte de la etapa de conversión de la señal analógica en una representación digital de la misma. A continuación presentamos los diferentes tipos de errores que encontramos en este proceso.

Error de compensación

El error de compensación esta definido como la diferencia entre la compensación nominal y la que realmente se tiene. Para el caso de un conversor analógico-digital, el punto de compensación nominal es el valor que se tiene en la entrada para obtener una salida igual a cero.

Cuando en la entrada se coloca este supuesto valor y en su salida se obtiene un valor diferente de cero, es justamente cuando existe un error de compensación. Esto bien puede corregirse ajustando los valores de compensación, y si no es posible ajustarlos, entonces se puede realizar en la etapa de análisis de los datos obtenidos, haciendo los ajustes necesarios en estos valores.

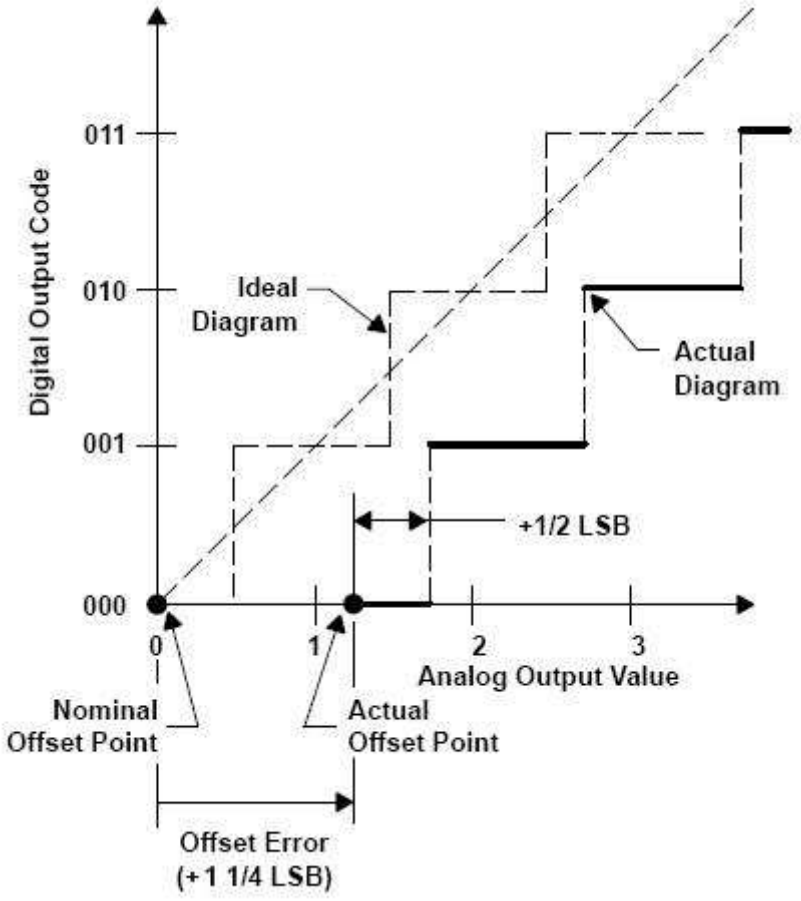


Fig. 2.8 Error de compensación

Error de ganancia

El error de ganancia se define como la diferencia entre el valor de ganancia nominal (ganancia esperada, calculada) y la ganancia real. Este valor se calcula cuando el error por compensación es nulo, caso contrario se obtendrá un error en un calculo generado por otro error.

La figura siguiente muestra de forma simple el significado de este error. Cuando se tiene un valor conocido a la entrada del conversor, de espera un valor de salida determinado. Cuando la diferencia entre el valor esperado y el obtenido es constante e igual sin importar cual sea el valor en su entrada, entonces se trata de un error de ganancia. Este error es lineal y constante en todo el rango posible de entrada. Este error también es posible solucionarlo, aunque también se puede corregir en etapas posteriores.

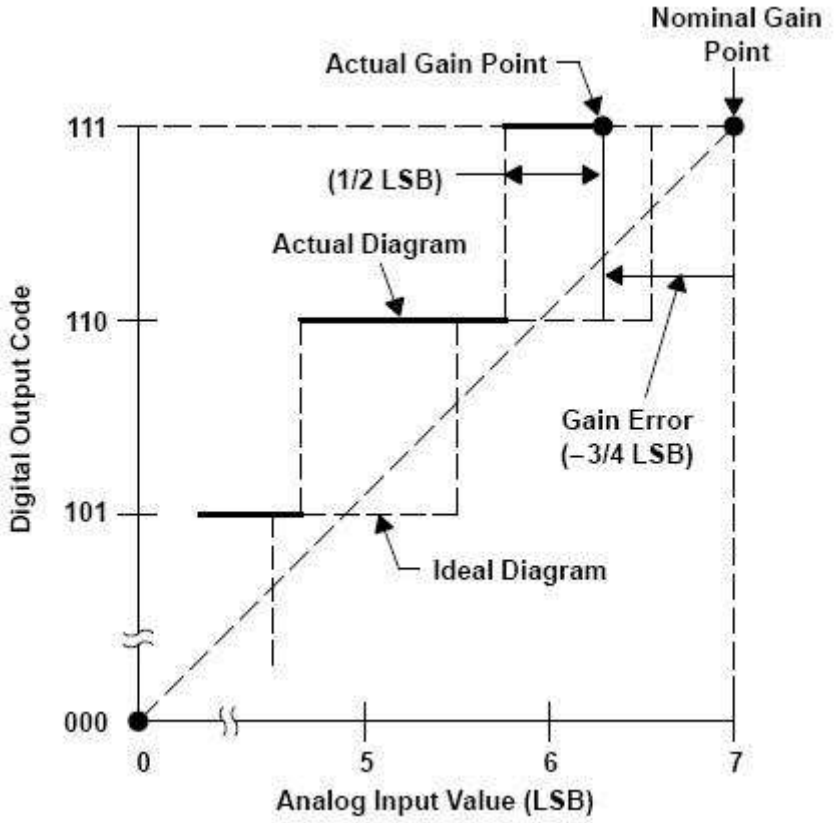


Fig. 2.9 Error de ganancia

Error de apertura

El error de apertura es un tipo de error generado en los propios componentes y no es corregible. La incertidumbre en la señal de entrada en el momento en que se muestrea y retiene dicha señal *(sample&hold)* esta dado desde el instante en que se la muestrea hasta que se la retiene, antes de pasar al proceso de conversión. El error de apertura puede ser generado por ruidos o también por variaciones en la señal de reloj. Este error también influye en las características finales del sistema.

Este error puede ser reducido si el tiempo de muestreo y retención disminuye. Si este valor es lo más pequeño posible, entonces se tiene mayor certeza de que la conversión realizada es correcta.

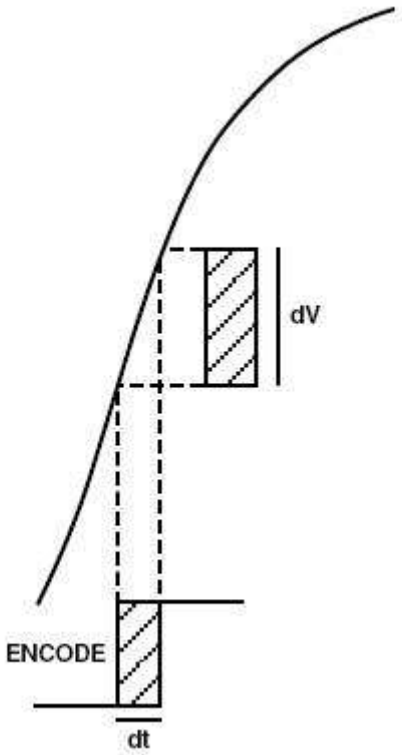


Fig. 2.10 Error de apertura

Error de no-linealidad diferencial

El error de no-linealidad diferencial *(Differential Nonlinearity* | *DNL Error)* es la diferencia entre la variación de tensión nominal que genera un cambio en un bit a la salida del conversor (1 LSB) y la variación real que debe ocurrir. Es decir, que si el cambio de tensión que debe ocurrir a la entrada para que haya un cambio en la salida de un bit es exactamente el esperado, entonces el error DNL es cero. En cambio, si para que haya un cambio en la salida, el diferencial de tensión a la entrada debe ser mayor (o menor) al que se especifica, entonces existe este tipo de error. Incluso este tipo de error puede generar que existan valores binarios de salida que nunca se lleguen a dar, debido que el rango de entrada tiene como respuesta una menor cantidad de valores debido al error citado (cuando el diferencial de tensión de entrada es mayor que el especificado para 1 LSB).

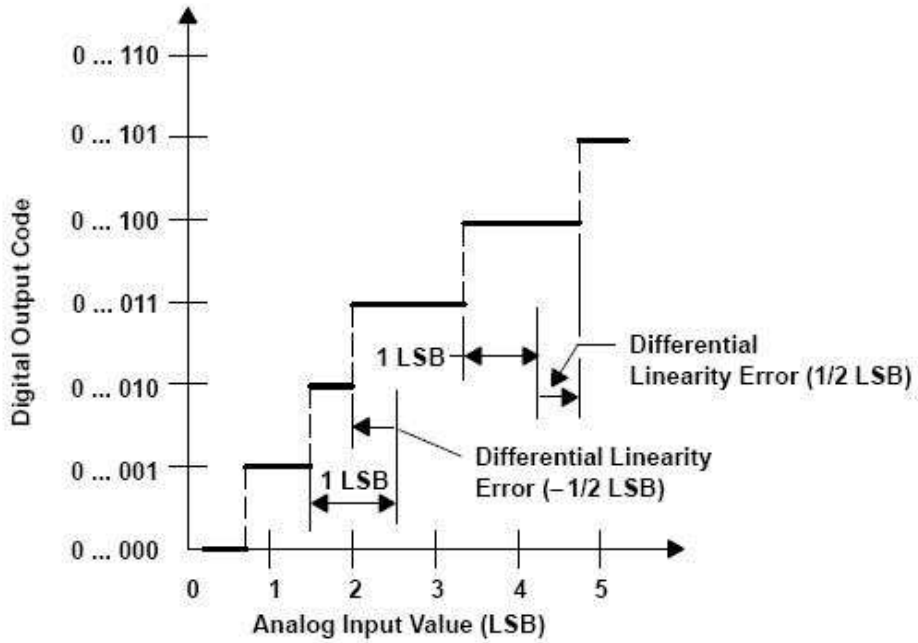


Fig. 2.11 DNL Error

Error de no-linealidad integral

El error de no-linealidad integral (Integral Nonlinearity | INL Error)_ esta definido como la desviación de los valores de la función de transferencia real sobre una linea recta. Esta línea recta puede ser definida como la mejor aproximación que minimice estas desviaciones, o bien entre los puntos extremos de la función de transferencia una vez que se hayan anulado los errores de ganancia y compensación. Este segundo método es conocido como *linealidad de punto final*, y es el más usado ya que su verificación es más simple.

En un conversor analógico-digital esta desviación se mide en la transición de un paso al siguiente (es decir, un aumento en 1 LSB), y el nombre de error *integral* proviene de que se trata de la suma de estas desviaciones desde el nivel más bajo hacia un valor determinado. Con esta definición se puede conocer entonces el error causado por la no-linealidad integral en cada valor.

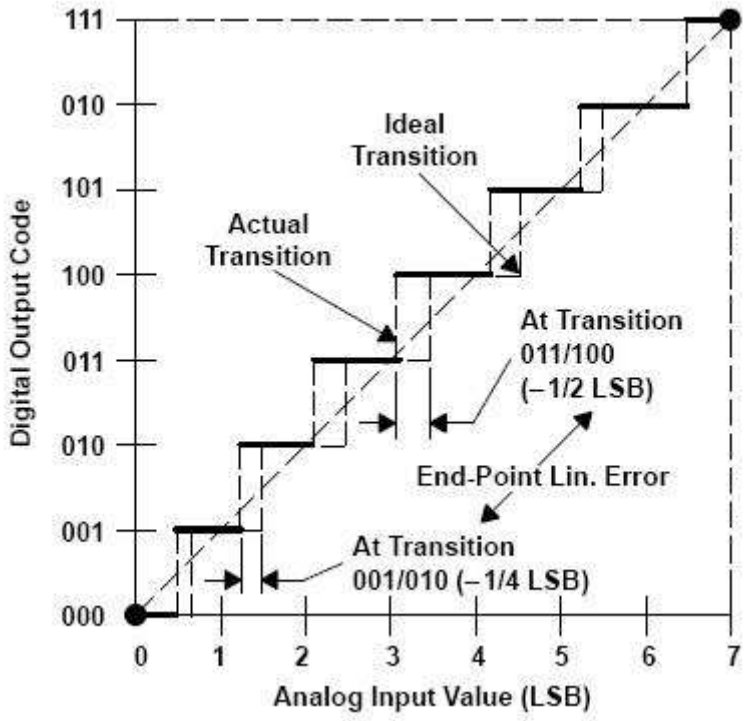


Fig. 2.12 INL Error

Error de cuantización

El error de cuantización esta basado en la naturaleza de las señales analógicas y las digitales. Una señal analógica es continua, y puede tener *infinitos* valores (por mas que la señal este acotada), mientras que una señal digital es discreta, tiene una cantidad *finita* de valores posibles.

Aquí entonces radica este error. Al intentar traducir una señal que puede tener *infinitos* valores en otra que solo puede tener valores *finitos*, esta claro que se pierde información. La cantidad de valores posibles (o *estados* posibles) que puede tener la señal digital esta relacionada con la cantidad de bits con la cual ésta se representa. Sin embargo la cantidad de bits es una cantidad *finita*. La única forma de hacer que este error sea nulo, es haciendo que la cantidad de bits con la que se representa el valor digital sea infinito. Esto implica que cierto valor digital representa a *muchos* (de hecho, infinitos) valores analógicos posibles. De esta forma, un valor analógico produce una salida digital, y esa salida digital si se vuelve a convertir a un valor analógico, puede que no corresponda con el valor original. Cierta información se ha perdido en este proceso, y esto se conoce como error de cuantización.

En el caso de un conversor ideal, donde la función de salida puede creerse como una *escalera perfecta*, el error entre la señal real de entrada y su correspondencia con la salida digital, tiene una función de densidad de probabilidad uniforme en el caso de que se asume que la entrada es totalmente aleatoria. Este error varía en el rango de ±1/2 LSB, o bien, ±q/2, donde q es el ancho de un escalón, tal se muestra en la siguiente figura.

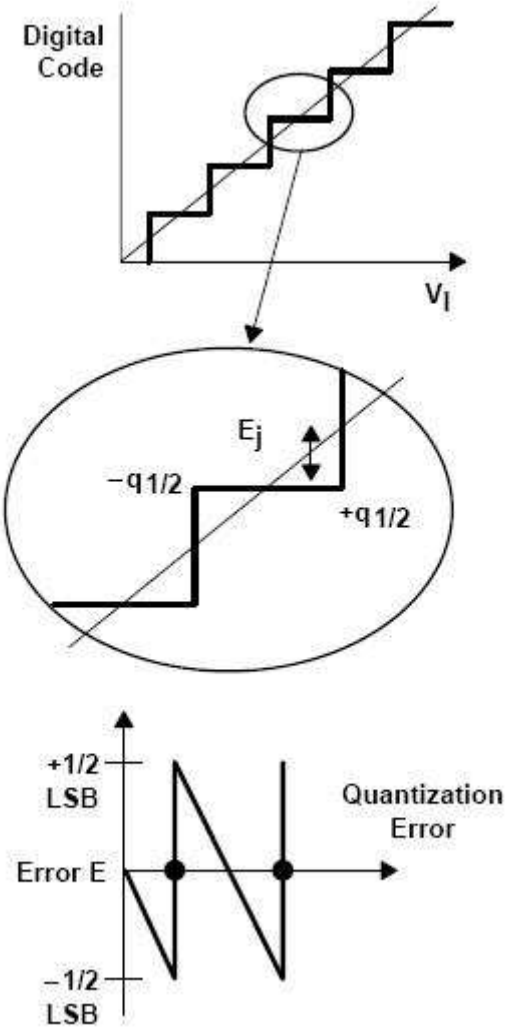


Fig. 2.13 Error de cuantificación

Error absoluto

El error absoluto en la exactitud del proceso de conversión es la diferencia máxima que se encuentra entre el valor analógico de la señal de entrada con el valor medio que se define para ese mismo código binario de salida. Este error es absoluto, y es por ello que incluye a todos los errores citados previamente (compensación, ganancia, no linealidad, e incluso cuantización).

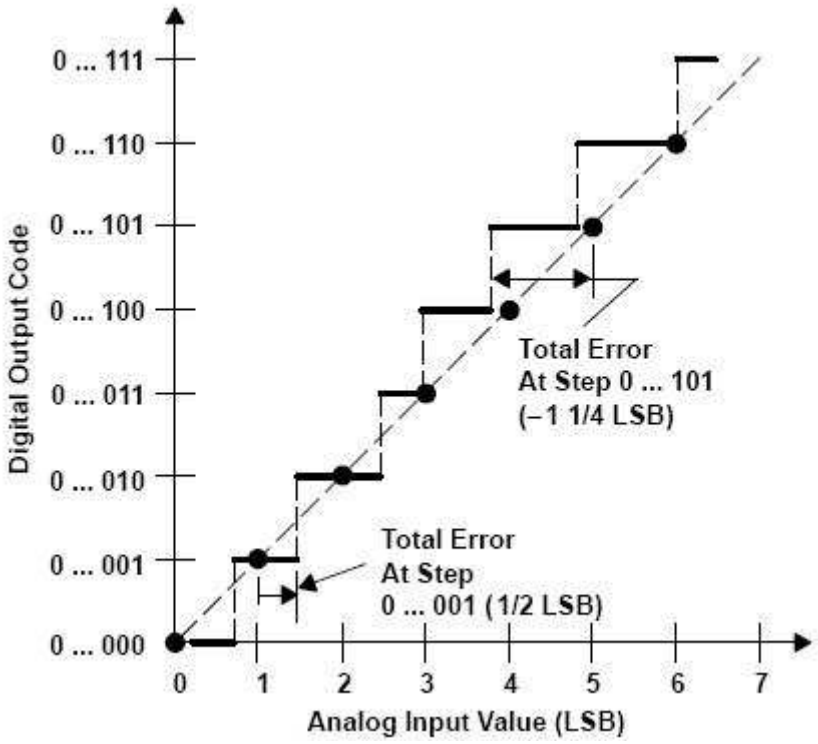


Fig. 2.14 Error absoluto

Referencias

- Pico Technology
  - http://www.picotech.com
- TiePie Engineering
  - http://www.tiepie.nl
- ETC
  - http://www.etcsk.com
- BitScope
  - http://bitscope.com/

- Diferentes tipos de muestreo
  - http://www.systems.caltech.edu/dsp/students/bojan/journ/Nyquist7decades.pdf
  - http://dolphin.wmin.ac.uk/~artur/pdf/Paper12.pdf
  - http://www.hiraeth.com/alan/papers/DSP99/DSP99.pdf
  - http://www.edn.com/article/CA293235.html
  - http://tritium.fis.unb.br/Fis3Exp/www.tmo.hp.com/tmo/pia/BasicInstrument/TUTnBRIEF/English/BI-T-Sampling-010.html
  - http://www.answers.com/topic/nyquist-shannon-sampling-theorem
  - http://www.pentek.com/deliver/TechDoc.cfm/PutUndersamp.pdf?Filename=PutUndersamp.pdf
  - http://www.maxim-ic.com/appnotes.cfm/appnote_number/3190
  - http://www.bitscope.com/design/hardware/convertor/?p=3#sub

- Errores de cuantización
  - http://focus.ti.com/lit/an/slaa013/slaa013.pdf

- http://www.analog.com/UploadedFiles/Application_Notes/5356940929547373956522730668848977365163734AN501.pdf
- http://www.maxim-ic.com/appnotes.cfm/appnote_number/1197
- http://ww1.microchip.com/downloads/en/devicedoc/adn010.pdf
- http://ww1.microchip.com/downloads/en/devicedoc/adc.pdf
- http://today.answers.com/topic/quantization-error

- Imágenes
  - Fig. 2.1 - http://tritium.fis.unb.br/Fis3Exp/www.tmo.hp.com/tmo/pia/BasicInstrument/TUTnBRIEF/English/BI-T-Sampling-010.html
  - Fig. 2.2 - http://tritium.fis.unb.br/Fis3Exp/www.tmo.hp.com/tmo/pia/BasicInstrument/TUTnBRIEF/English/BI-T-Sampling-010.html
  - Fig. 2.3 - http://tritium.fis.unb.br/Fis3Exp/www.tmo.hp.com/tmo/pia/BasicInstrument/TUTnBRIEF/English/BI-T-Sampling-010.html
  - Fig. 2.4 - http://www.pentek.com/deliver/TechDoc.cfm/PutUndersamp.pdf?Filename=PutUndersamp.pdf
  - Fig. 2.5 - http://www.pentek.com/deliver/TechDoc.cfm/PutUndersamp.pdf?Filename=PutUndersamp.pdf
  - Fig. 2.6 - http://www.pentek.com/deliver/TechDoc.cfm/PutUndersamp.pdf?Filename=PutUndersamp.pdf
  - Fig. 2.7 - http://www.pentek.com/deliver/TechDoc.cfm/PutUndersamp.pdf?Filename=PutUndersamp.pdf
  - Fig. 2.8 - http://focus.ti.com/lit/an/slaa013/slaa013.pdf
  - Fig. 2.9 - http://focus.ti.com/lit/an/slaa013/slaa013.pdf
  - Fig. 2.10 - http://focus.ti.com/lit/an/slaa013/slaa013.pdf
  - Fig. 2.11 - http://focus.ti.com/lit/an/slaa013/slaa013.pdf
  - Fig. 2.12 - http://focus.ti.com/lit/an/slaa013/slaa013.pdf
  - Fig. 2.13 - http://focus.ti.com/lit/an/slaa013/slaa013.pdf
  - Fig. 2.14 - http://focus.ti.com/lit/an/slaa013/slaa013.pdf

Capítulo 3. Bus serie universal (USB)

Contenido de este capítulo

Introducción

El Universal Serial Bus (USB) es un estándar diseñado para conectar dispositivos, a través de un bus serie. Fue originalmente pensado para conectar dispositivos a computadoras, eliminando la necesidad de conectar tarjetas PCI (o similares), como así también conectar y desconectar los dispositivos sin tener que reiniciar la PC (hot-swap). Sin embargo, hoy en día también se utiliza en consolas de juegos e incluso en algunos equipos de audio y video.

El diseño del protocolo USB está a cargo del USB Implementers Forum (USB-IF), una organización compuesta por varias empresas del ramo de la computación y la electrónica, entre las que se encuentran Apple Computer, Hewlett-Packard, Microsoft e Intel.

Existen tres versiones del protocolo (1.0, 1.1 y 2.0). A diferencia de las anteriores, la última versión (2.0) soporta tasas de transferencia de altas velocidades, comparables (o incluso superiores) a la de un disco duro o almacenamiento magnético, lo cual ha permitido ampliar el uso del USB a aplicaciones de video y almacenamiento (discos duros externos). Una de las razones a la cual se atribuye su gran popularidad es que todas las versiones del protocolo son compatibles hacia atrás. Es decir, que cualquier dispositivo 2.0 puede ser conectado a un dispositivo 1.0, aunque funcionará la velocidad del más lento.

Existen tres tipos de velocidades en la comunicación. Ellas son:

Tipo	Velocidad
Baja velocidad (low speed)	183 Kbytes/s (1.5Mbit/s)
Velocidad completa (full speed)	1.4 Mbytes/s (12Mbit/s)
Alta velocidad (high speed)	57 Mbytes/s (480 Mbit/s)

Table 3.1 Velocidades de transferencias USB

Los de baja velocidad generalmente son dispositivos de interacción con la computadora, como mouses, teclados y joysticks.

Topología

USB tiene un diseño asimétrico ya que consiste de un host controlador conectado a múltiples dispositivos conectados en daisy-chain.

USB conecta varios dispositivos a un_host controlador a través de un cadenas de hubs. Los hubs (al igual que en redes) son dispositivos que permiten, a partir de un único punto de conexión, poder conectar varios dispositivos, es decir, disponer de varios puntos de conexión. De esta forma se crea una especie de estructura de árbol. El estándar admite hasta 5 niveles de ramificación por host controlador con un límite absoluto de 127 dispositivos conectados al mismo bus (incluyendo los hubs). Siempre existe un hub principal (conocido como el hub raíz) que está conectado directo al host controlador.

Un mismo dispositivo USB puede cumplir varias funciones. Por ejemplo, un mouse puede ser también lector de tarjetas, y de esa forma sería como dos dispositivos conectados al bus USB. Por lo tanto es muy común hablar de funciones en lugar de dispositivos.

Funcionamiento

Los dispositivos (o mejor dicho, las funcionas) tienen asociados unos canales lógicos unidireccionales (llamados *pipes*) que conectan al host controlador con una entidad lógica en el dispositivo llamada *endpoint*. Los datos son enviados en paquetes de largo variable (potencia de 2). Típicamente estos paquetes son de 64, 128 o más bytes.

Estos endpoints (y sus respectivos pipes) son numerados del 0 al 15 en cada dirección, por lo cual un dispositivo puede tener hasta 32 endpoints (16 de entrada y 16 de salida). La dirección se considera siempre desde el punto de vista del host controlador. Así un endpoint de salida será un canal que transmite datos desde el host controlador al dispositivo. Un endpoint solo puede tener una única dirección. El endpoint 0 (en ambas direcciones) está reservado para el control del bus.

Cuando un dispositivo es conectado al bus USB, el host controlador le asigna una dirección única de 7 bit (llamado proceso de enumeración) que es utilizada luego en la comunicación para identificar el dispositivo (o, en particular, la función). Luego, el host controlador consulta continuamente a los dispositivos para ver si tiene algo para mandar, de manera que ningún dispositivo puede enviar datos sin la solicitud previa explícita del host controlador.

Para acceder a un endpoint se utiliza una configuración jerárquica de la siguiente manera: un dispositivo/función conectado al bus tiene un único descriptor de dispositivo, quien a su vez tiene uno (o varios) descriptores de configuración. Estos últimos guardan generalmente el estado del dispositivo (ej: activo, suspendida, ahorro de energía, etc). Cada descriptor de configuración tiene uno (o más) descriptores de interfaz, y éstos a su vez tienen una

configuración por defecto (aunque puede tener otras). Y éstos últimos finalmente son los que contienen los endpoint, que a su vez pueden ser reutilizados entre varias interfaces (y distintas configuraciones).

Como puede verse, la comunicación USB es bastante compleja y extremadamente más complicada que una simple comunicación serie.

Tipos de transferencia

Los canales también se dividen en cuatro categorías según el tipo de transmisión:

- **transferencias de control** - usado para comandos (y respuestas) cortos y simples. Es el tipo de transferencia usada por el pipe 0
- **transferencias isócronas** - proveen un ancho de banda asegurado pero con posibles pérdidas de datos. Usado típicamente para audio y video en tiempo real
- **transferencias interruptivas** - para dispositivos que necesitan una respuesta rápida (poca latencia), por ejemplo, mouse y otros dispositivos de interacción humana
- **transferencias masivas** - para transferencias grandes y esporádicas utilizando todo el ancho de banda disponible, pero sin garantías de velocidad o latencia. Por ejemplo, transferencias de archivos.

En realidad las transferencias interruptivas no son tales ya que los dispositivos no pueden enviar datos sin recibir autorización del host controlador. Por lo tanto, las transferencias interruptivas simplemente le dan más prioridad al sondeo del host controlador.

Señalización y conectores

Las señales USB son transmitidas en un par trenzado (cuyos hilos son denominados D+ y D-) utilizando señalización diferencial half-duplex minimizando el ruido electromagnético en tramos largos. El diseño eléctrico permite un largo máximo de 5 metros (sin precisar un repetidor intermedio).

Existen dos tipos de conectores: estándar y mini. Los estándar son los que típicamente encontramos en un computador y vienen en dos tipos: A y B. El tipo A es el que es chato y se encuentra del lado del host controlador, mientras que el tipo B es el cuadrado y se encuentra del lado del dispositivo. Todos los cables son machos, mientras que los enchufes (ya sea en la computadora o los dispositivos) son hembra. No existen intercambiadores de género puesto que las conexiones cíclicas no están permitidas en un bus USB. Los conectores mini siguen la misma política que los estándar pero son utilizados para dispositivos pequeños como Palm y celulares.

Los pines de un cable USB son los siguientes:

Pin	Color	Función

1	Rojo	BUS (4.4 - 5.25 V)
2	Blanco	D-
3	Verde	D+
4	Negro	Tierra
5	en corto con pin 4 en conector Mini-A, utilizado para USB On-The-Go	

Tabla 3.2 Pines del bus USB

A continuación se muestra un diagrama de los conectores (las medidas están en mm) y los números de los pines se corresponden con la tabla anterior.

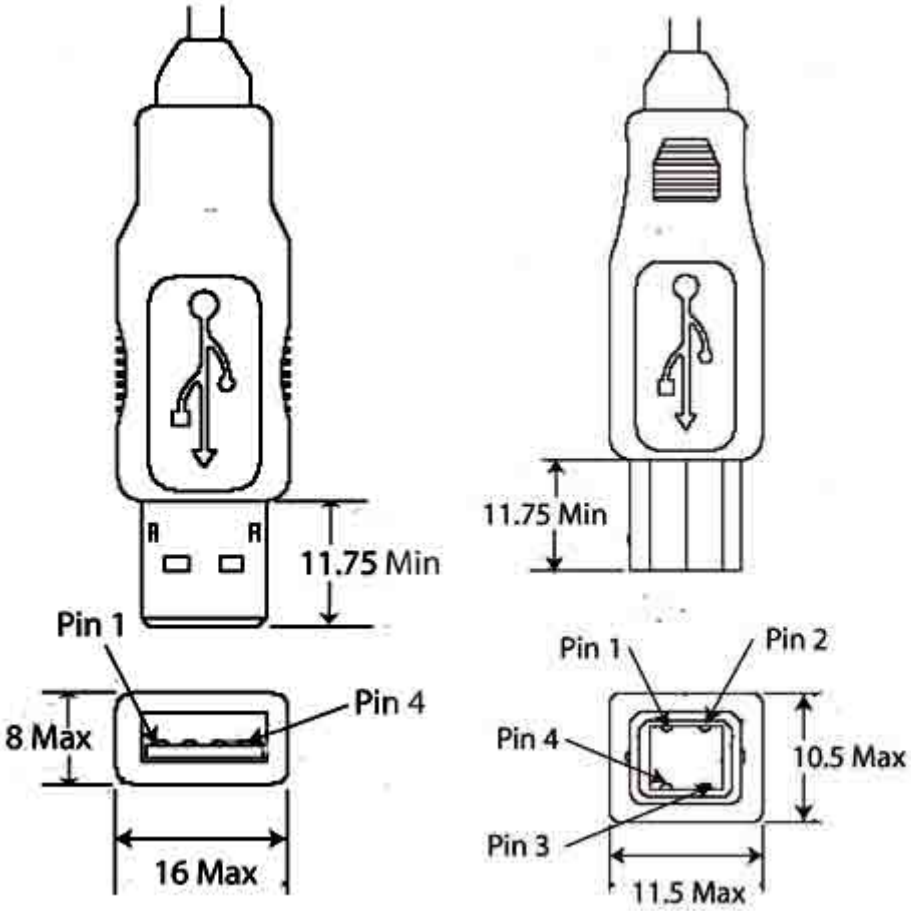


Fig. 3.1 Conector USB tipo A (izquierda) y tipo B (derecha)

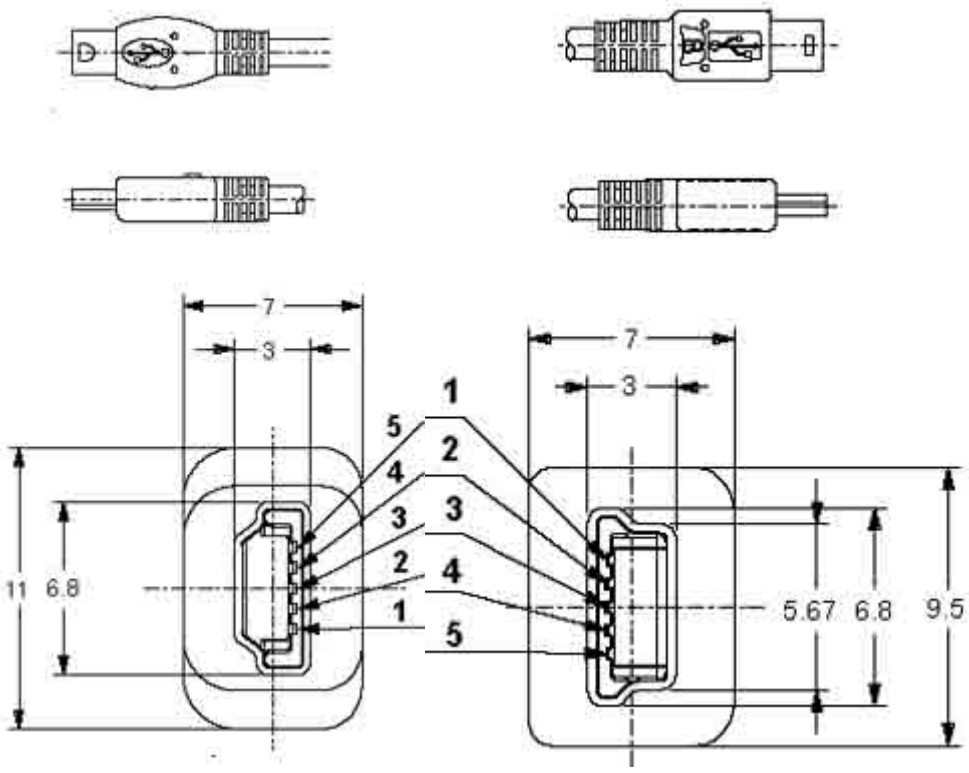


Fig. 3.2 Conector USB Mini tipo A (izquierda) y tipo B (derecha)

Potencia

El bus USB suministra 5V de continua regulados por cada uno de sus puertos, entre los pines 1 y 4. Por lo tanto, dispositivos de bajo consumo de potencia (que de otra forma vendría con un fuente de alimentación) puede obtener de allí la corriente necesaria para el funcionamiento. El límite de corriente suministrada es de 500mA por cada puerto. Además, el estándar exige no más de 5.25V en ningún caso, ni menos de 4.375V en el peor caso. Típicamente el voltaje se mantiene en los 5V.

Algunos hubs se alimentan directamente del bus USB, en cuyo caso la corriente total de todos los dispositivos conectados a él no puede superar los 500mA. Sin embargo, la especificación permite solo un nivel de hub alimentados por el bus, de forma que no es posible conectado un hub sin alimentación a otro hub sin alimentación. Los hubs con alimentación propia no tienen esta restricción y generalmente son necesario para conectar dispositivos de alto consumo como impresoras o discos duros.

Cuando un dispositivo es conectado le reporta al host controlador cuando potencia va a consumir. De esta manera el host controlador lleva un registro de los requisitos de cada puerto y generalmente cuando un dispositivo se excede generalmente se apaga, cortándole el suministro de corriente, de forma de no afectar al resto de los dispositivos. El estándar exige que los dispositivos se conecten en un modo de bajo consumo (100 mA máximo) y luego le comuniquen al host controlador cuanta corriente precisan, para luego cambiar a un modo de alto consumo (si el host se lo permite).

Los dispositivos que superen los límites de consumo deben utilizar su propia fuente de poder.

Los dispositivos que no cumplan con los requisitos de potencia y consuman más corriente de la negociada con el host puede dejar de funcionar sin previo aviso 0, en algunos casos, dejar todo el bus inoperativo.

Clases de dispositivos

Los dispositivos que se conectan puede tener sus propios drivers personalizados. Sin embargo, existen lo que se llaman clases de dispositivos que son pequeños estándar para distintos tipos de dispositivos y especifican como deben compartirse los dispositivos en términos de los descriptores de interfaz y de dispositivo, endpoints, etc. Esto permite que todos los dispositivos sean fácilmente intercambiables y/o sustituibles puesto que hablan el "mismo idioma". Por su parte, los sistema operativos solo tienen que implementar drivers genéricos para todas las clases conocidas de dispositivos lo cual alivia el alto costo de desarrollar y mantener un driver particular para cada producto y modelo.

En conclusión, las clases de dispositivos son una estandarización de funcionalidades a un nivel superior al del bus BUS y que utiliza a éste último como medio de comunicación e intercambio de datos.

Tanto los descriptores de dispositivos como los descriptores de interfaz tiene un byte que identifica la clase. En el primer caso, todo el dispositivo/función pertenece a la misma clase, mientras que en el segundo un mismo dispositivo puede tener diferentes clases, cada una asociada a su descriptor de interfaz.

Dado que el identificador de la clase es un byte, existe un máximo de 253 clases diferentes (0x00 y 0xFF están reservados). Los códigos de las clases son asignados por el el USB Implementers Forum, y a continuación se presenta una lista de los más comunes.

Código	Clase
0x00	Reservado. Usado en el descriptor de dispositivo para indicar que la clase está identificada en él (o los) descriptores de interfaz
0x01	Dispositivo de audio. Por ejemplo; tarjetas de sonidos
0x03	Dispositivo de interfaz humana (HID). Por ejemplo: mouses, teclados, joystick
0x07	Impresoras
0x08	Dispositivo de almacenamiento masivo. Por ejemplo: discos duros, lectores de memoria, cámaras digitales, reproductores MP3
0x09	Hubs
0x0A	Dispositivo de comunicación (CDC por sus siglas en inglés). Por ejemplo: módems, tarjetas de red
0x0E	Dispositivo de video. Por ejemplo: webcams
0xE0	Controladores inalámbricos. Por ejemplo: adaptadores Bluetooth

0xFF	Reservado. Usado para indicar que el dispositivo tiene un driver personalizado propio que no pertenece a ninguna clase

Tabla 3.3 Clases de dispositivos USB

Nuestro osciloscopio USB, en particular, pertenece a la clase CDC (0x0A).

Referencias

- USB Implementers Forum (USB-IF)
  - http://www.usb.org/
- Especificación USB 2.0
  - http://www.usb.org/developers/docs/

Capítulo 4. Hardware

Contenido de este capítulo

Selección de la arquitectura

La primera decisión importante que tuvimos que tomar en la etapa de análisis fue definir la arquitectura sobre la cual íbamos a construir el osciloscopio. Es una decisión crucial porque todo el resto del diseño (selección de componentes, etc) depende de ella.

El elemento más importante de la arquitectura es el microcontrolador a usar, pues define el juego de instrucciones disponibles, el lenguaje a utilizar, etc.

En nuestro caso la decisión estuvo entre las siguientes 3 arquitecturas:

1. usar un único integrado con conversor analógico-digital (ADC) y USB incluido
2. usar un sistema linux embedded similar a un PC de bajo porte
3. armar una placa con un microprocesador controlador y diferentes componentes específicos para cada tarea (ADCs, memoria, etc)

1. Arquitectura: Chip único

La opción de utilizar un único chip que ya tuviera integrado el ADC y el controlador USB resultó muy tentadora al principio pues simplificaba enormemente muchas otras decisiones de diseño. Un integrado en particular que analizamos fue el C8051F065 de Silicon Laboratories (ver referencias) el cual contaba con un ADC de 1MSPS/16 bits y un micro 8051. Su costo es de U$S 24 (el chip) + U$S 300 (placa de desarrollo). Además habría que comprarle un bridge UART-USB (U$S 5) y su opcionalmente su correspondiente placa de desarrollo (U$S 50).

Sin embargo, al profundizar el estudio nos dimos cuenta que la meta de los 20 MHz era inalcanzable siguiendo este camino, puesto que los ADC que vienen incorporados en este tipo de chips ronda en los 500 Khz - 1 Mhz y, aun aplicando técnicas de submuestreo (y suponiéndolas exitosas) la frecuencia máxima que era posible muestrear estaba muy lejos del mínimo requerido.

Ademas esta alternativa tenía la desventaja de la falta de flexibilidad, puesto que de haber escogido este modelo hubiéramos realizado un diseño completamente atado al integrado en cuestión y para nada extensible.

Por lo tanto, debido a estas dos desventajas (velocidad de captura y portabilidad del diseño) decidimos descartar esta alternativa.

2. Arquitectura: Linux Embedded

Otra opción que investigamos muy a fondo es la posibilidad de utilizar una arquitectura del tipo linux embedded, en el cual el sistema consta básicamente de un PC, pero con menor potencia de procesamiento.

En éste área analizamos en particular el procesador ETRAX de la compañía Axis (ver referencias). El procesador ETRAX es un procesador de 32 bits que trabaja a 100 MHz y está diseñado para correr el sistema operativo Linux. Esto nos brinda una flexibilidad enorme a la hora de desarrollar sobre dicha arquitectura ya que Linux es una plataforma muy popular y con excelente documentación lo cual nos brinda la sencillez de poder desarrollar en C sobre una plataforma robusta y probada.

La gente de Axis pone a disposición una placa de desarrollo que es apropiada para una gran variedad de aplicaciones, entre ellas nuestro proyecto. La misma cuenta de:

- 2 puertos USB
- 1 puerto ethernet
- 2 puertos seriales
- 8 contactos secos

Fig 4.1 Placa de desarrollo ETRAX

Otra de las grandes ventajas de este enfoque es la de poder implementar un osciloscopio de red autónomo ("standalone"), es decir, que funcione independiente de una PC, al cual nosotros llamamos "Osciloscopio IP". La interfaz por excelencia en este tipo de dispositivos hoy en día es la Web, y este caso no sería la excepción. Alguien con una Palm y acceso a la red del osciloscopio podría perfectamente manejarlo. La misma aplicación se puede extender para casos en los que el usuario no se encuentra físicamente en el

mismo lugar que el osciloscopio de forma que alguien pueda dejar conectado el osciloscopio en el laboratorio y luego sacar muestras desde la casa.

Sin embargo, debido a la naturaleza de nuestro proyecto es indispensable discutir sobre dos temas: por un lado, como se hará la conexión de los ADC (conversores analógicos-digitales), ya que la placa de desarrollo no viene con ADC incluidos, y por otro como se hará para asegurar el funcionamiento en tiempo real del dispositivo. Estos puntos se discuten a continuación:

Conexionado de los ADC

Debido a la alta velocidad de trabajo requerida para los conversores analógico-digital (40 Mhz) la única forma de conectar el ADC es usando DMA, de forma que los datos se transfieran directamente a la memoria, sin requerir el uso del procesador como intermediario. El ETRAX en particular viene con dos canales DMA externos (ademas de los típicos internos) que son ideales para conectar los dos ADC que va a tener nuestro osciloscopio.

Los canales de DMA externo del ETRAX pueden trabajar de dos formas diferentes: en modo negociación (*handshake*) y en modo ráfagas (*burst*). Sin embargo, aún trabajando en modo ráfagas un ciclo completo de DMA dura no menos de 5 ciclos de reloj, lo cual nos reduce la frecuencia de muestreo máxima a 20 Mhz, restringiendo así a 10 MHz el ancho de banda máximo de las señales a medir.

Linux en tiempo real

Debido a las exigencias tan estrictas de tiempo, uno de los requisitos para poder implementar el osciloscopio con el ETRAX es la utilización de un linux compilado para poder poder trabajar en tiempo real (característica no disponible en los linux estándar). Actualmente existen varias distribuciones de linux en tiempo real (real-time linux). Dos de las más conocidas son: RTLinux y RTAI. RTLinux nació como un proyecto de código abierto pero luego se hizo comercial, cerrando el código y la comunidad de software libre que lo apoyaba. De esta, mucha gente que trabaja en RTLinux se volcó a trabajar en otro proyecto muy similar ya existente llamado RTAI en el cual la permanencia del código bajo licencia libre es uno de sus principios básicos. Hoy en día RTAI es la distribución de real-time linux con mas movimiento, aunque RTLinux sigue siendo la preferencia para proyectos crítico de medicina y aeronáutica.

Afortunadamente para nosotros, un par de señores ya portó el RTAI para el ETRAX como tesis de maestría (ver Referencias) y por lo tanto lo tendríamos disponible para usar en el proyecto en el caso de que optáramos por esta arquitectura.

Ventajas y prestaciones adicionales de un Osciloscopio IP

La interfaz web podría tener un menú de _captura fuera de línea, es decir, una página donde se llena el formulario con los parámetros de la captura y luego se envía la solicitud de captura, devolviendo el osciloscopio inmediatamente los

datos medidos, ya sea en forma de un imagen o una tabla de datos en formato CSV (para abrir en Excel, por ejemplo).

También dentro de la interfaz web se podría proveer una **aplicación Java** para controlar el osciloscopio, evitando de esta forma tener que instalar un software en las PCs donde se lo quiera usar: bastaría solo con tener Java instalado, lo cual es algo muy común en las PCs de hoy en día y además le agregaría la ventaja de ser multiplataforma (correría en Windows, Linux y Mac).

Otra idea interesante disponible en el caso de utilizar el procesador ETRAX es la de tener la posibleidad de conectarle al osciloscopio un disco duro USB (tipo pendrive) y realizar captura de datos allí para luego analizarlos posteriormente en una PC.

Otras ventajas de esta arquitectura son las siguientes:

- es algo innovador (no hemos encontrado ningún proyecto similar existente)
- deberíamos tener menos sorpresas armando el sistema con el ETRAX puesto que hay menos partes de hardware para implementar y la placa de desarrollo ya viene diseñada para trabajar a altas frecuencias (100 Mhz)
- la sencillez y flexibilidad de programación que brinda el poder programar C bajo un entorno linux

Desventajas

Sin embargo, si bien esta alternativa es muy tentadora, el costo de la placa de desarrollo es un poco elevado (U$S 300) lo cual nos aleja un poco de los límites estipulados en los objetivos y los del propio presupuesto del proyecto.

3. Arquitectura: Microprocesador y componentes separados

Por último, la tercera arquitectura analizada fue la de utilizar un microprocesador que se encargue de controlar la placa e implementar la lógica del hardware utilizando otros componentes (más veloces) para la captura de datos.

En nuestro caso, el microprocesador que elegimos fue un PIC18F4550 ya que poseía dos grandes ventajas:

- viene con controlador USB integrado lo cual deja resuelto el tema la comunicación USB
- es de bajo coste (aprox. U$S 6 en USA)

De hecho, este PIC también contiene un ADC pero es de muy baja velocidad. Por lo tanto, se deberá usar ADCs externos (cada uno con su memoria) para realizar las capturas a alta velocidad. Para poder controlar y direccionar las memorias se utilizaran contadores binarios en cascada que serán controlados desde el PIC.

Desventajas:

- Requiere el uso de **submuestreo** para lograr llegar a las frecuencias esperadas de trabajo (10 Mhz)
- Nos atamos a usar este integrado con todos los problemas de dependencia y escalabilidad que ello genera.

Funcionamiento y diagrama de bloques

Luego de discutir por unas semanas sobre la lógica de funcionamiento llegamos al siguiente modelo:

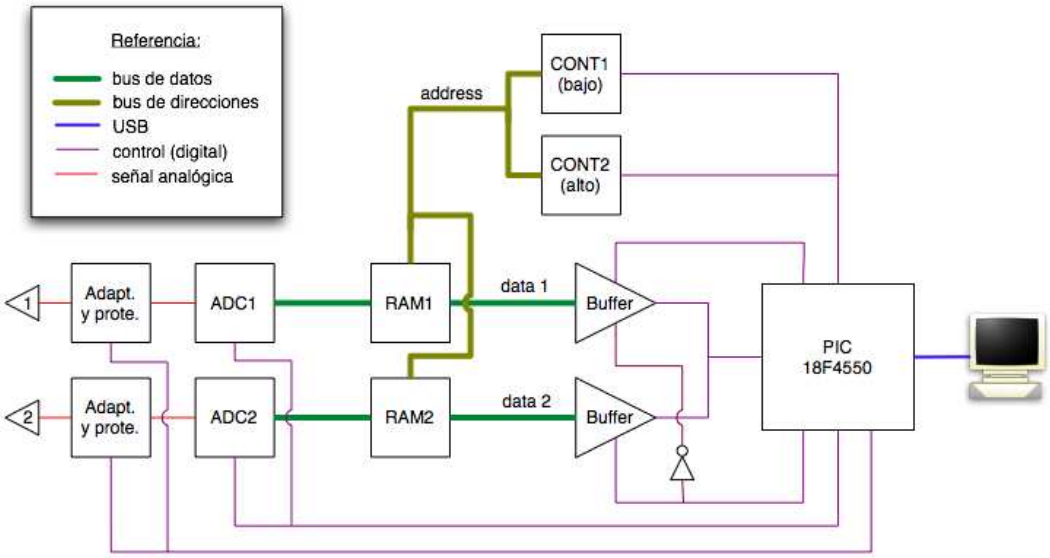


Fig 4.2 Diagrama de bloques

Si bien el PIC puede trabajar hasta 48 Mhz, su capacidad de procesamiento no le permite capturar y almacenar lo datos a muy altas frecuencias. Por lo tanto, es necesario que los conversores analógico-digitales (ADCs) se conectan directo a las memorias, y que a su vez sean direccionadas a través dos contadores rápidos. Estos contadores son comandados por el PIC.

La memoria es entonces direccionada a través del contador por dos razones:

1. porque el PIC no posee una cantidad suficiente de patas para controlar simultáneamente las memorias y el resto de la lógica
2. porque al capturar es necesario direccionar a altas velocidades (40 Mhz) imposibles de alcanzar por el PIC

Los buffers son para seleccionar una u otra memoria a leer, puesto que solo se usaran 8 patas del PIC para el bus de datos.

Por lo tanto, al correrse el proceso de adquisición el PIC habilita los contadores que comienzan contar de forma creciente mientras los ADC muestrean los datos y estos son almacenados en las direcciones de memoria presentadas por

los contadores. El hecho de que haya 2 contadores es porque no existen contadores de 16 bits tan rápidos (40 Mhz) y tuvimos que colocar 2 de 8 bit en cascada para controlar las memorias (RAM1/RAM2), dado que 8 bits no eran suficientes.

El microprocesador PIC18F4550

El PIC18F4550 es un microprocesador de propósito general versátil y económico. Pertenece a la popular familia de procesadores PICmicro de la empresa norteamericana Microchip cuya sede se ubica en Chandler, Arizona (USA).

Fig 4.3 PIC18F4550 - empaquetado DIP-40

Lo particular del procesador PIC18F4550 es que es uno de los PICs que viene con soporta nativo para USB, lo cual quiere decir que incluyen un controlador USB interno que ya brinda patas de salida para conectar directo a la PC, sin la necesidad de *pull-ups* o ninguna circuitería externa.

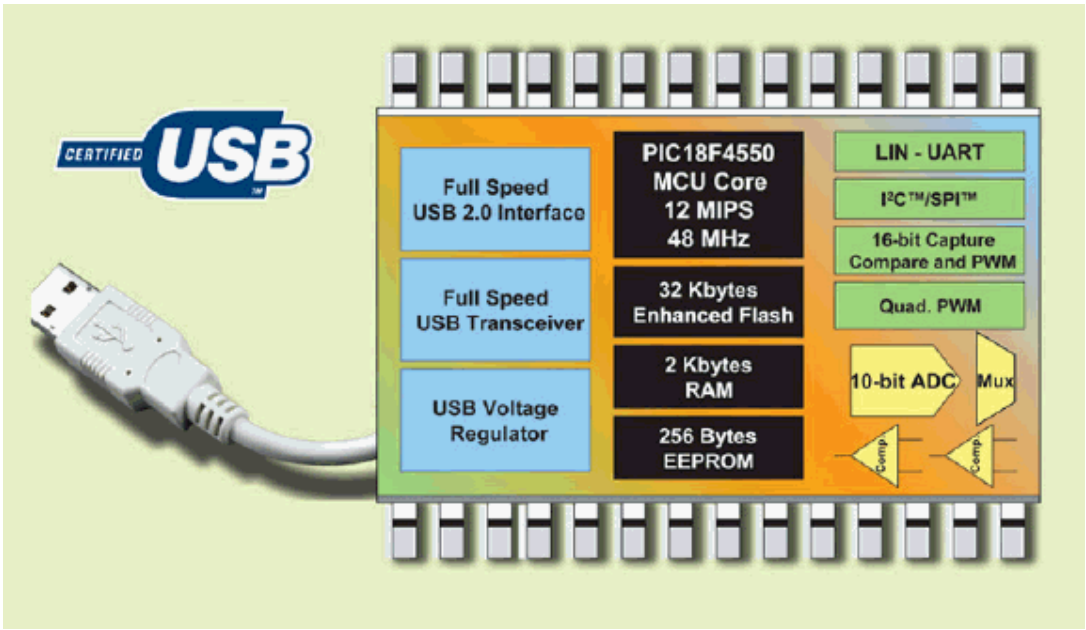


Fig 4.4 Características del PIC

Soporta cristales y osciladores de varias frecuencias como entrada y tiene post-scaler de manera que el procesador pueda trabajar a una frecuencia de 48 Mhz, independiente del oscilador que se conecte. Para ello debe configurarse

(a través de los configuration bits) el oscilador que se le ha conectado. Trabajar a 48 Mhz es un requisito para poder transferir a full-speed por el puerto USB. El controlador USB, por lo tanto, transfiere a full-speed (1.5 Mbytes/seg) por USB y es compatible con el estándar USB 2.0.

También cuenta con 35 patas de entrada/salida digitales de propósito general (ver pinout más adelante) y viene disponible en varios empaquetados, entre ellos DIP-40 lo cual lo hace una alternativa muy popular entre desarrolladores entusiastas y aficionados. Los puertos de entrada/salida son todos compatibles con la tecnología TTL. Cuando se los utiliza como salida, se comporta como un CMOS, siendo compatible con TTL, de modo de poder manejar cualquier tipo de tecnología. Sin embargo cuando son configurados los puertos como entrada, hay dos comportamientos posibles: puede ser exclusivamente TTL, o puede ser configurado para TTL o CMOS. Dado que ciertos puertos de entrada son solamente compatibles con la tecnología TTL, es que se ha optado por realizar toda la circuitería con tecnología TTL. Vale destacar que la única excepcion a esto es la etapa de entrada, en donde se han utilizado componentes CMOS, algunos con compatibilidad TTL y otros no. Esto se ha dado de este modo por la disponibilidad de los componentes, pero previo a una cuidadosa revisión para asegurar de que no existan problemas. Existen otras razones adicionales que hacen a la tecnología TTL la más adecuada para este caso, esto se explica a continuación, en la elección de los componentes.

En cuanto a memoria, posee 32Kb de flash para almacenamiento de programas, 2Kb de SRAM para memoria volátil, y 256 bytes de EEPROM (memoria no-volátil) para almacenamiento permanente de datos como configuraciones y demás.

Las instrucciones son de 1 byte de longitud con la excepción de algunas que ocupan 2 bytes (CALL, MOVFF, GOTO, LSFR). Utiliza el mecanismo de pipelining para la ejecución de código por lo cual hace que las instrucciones consecutivas se ejecutan en 4 CLK (períodos de reloj) y las que contengan saltos adicionan 4 CLK extras.

Otras características interesantes que posee son timers, interrupciones (externas e internas por timers) con dos niveles de prioridad y disparadas tanto por nivel como por flanco, un comparador analógico con un generador de voltaje de referencias de 16 niveles (útil para implementar un trigger de hardware por nivel).

Por último, el PIC también cuenta con un conversor analógico de 10-bit pero que para nuestro osciloscopio es insuficiente debido a la alta velocidad de captura necesaria. Ya que, si bien el oscilador es de 48 Mhz, entre los tiempo de ejecución de las interrupciones y otros delays (bucles, etc) no se pueden obtener velocidades de captura mayores a 200 KHz.

Pinout

A continuación se presenta el pinout del PIC18F4550, en empaquetado DIP-40. En particular se puede reconocer las pines D- y D+ de la conexión USB (patas 23 y 24).

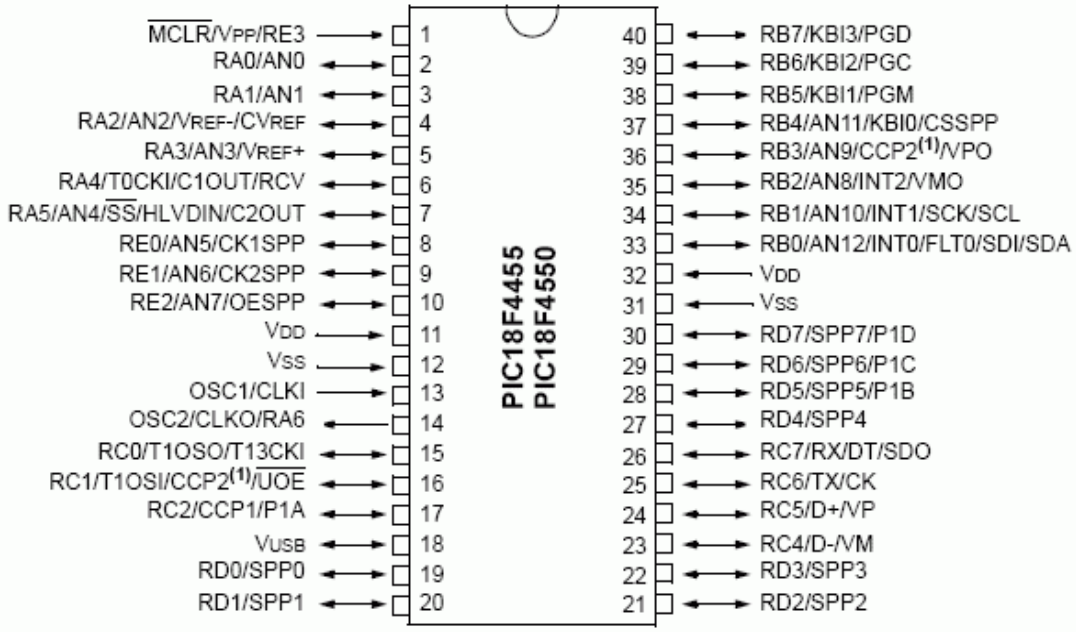


Fig 4.5 Pinout del PIC18F4550

Herramientas de programación

Una vez seleccionada la arquitectura hay que analizar cómo es que esta se programa. Al tratarse de un PIC, normalmente se utilizaría el *PICSTART Plus*, de Microchip. Sin embargo, el modelo que hemos seleccionado es el *único* que no funciona con este programador. Así mismo, una ventaja que teníamos si este programador nos sirviese, es que tendríamos uno disponible a nuestro alcance, sin la necesidad de comprar uno. Al momento de realización del proyecto, se estimaba que en el futuro una actualización iba a permitir utilizar dicho PIC, pero dado que aún esto no había ocurrido, corrimos con la necesidad de conseguir o construir nuestro propio programador.

La solución que encontramos fue el programador y depurador ICD2. Hemos conseguido uno a bajo precio en los Estados Unidos, pero al verse demorado este proceso, especialmente en que llegue a nuestras manos, optamos por comenzar construyendo nuestro propio programador.

PG2C | Programador PIC de interfaz serie

Este simple programador es el que nos permitió comenzar con la programación y prueba del microcontrolador. Antes de tener en nuestras manos el ICD2, debimos construir nuestro propio programador para poder comenzar con el proyecto.

A continuación mostramos el esquemático del programador:

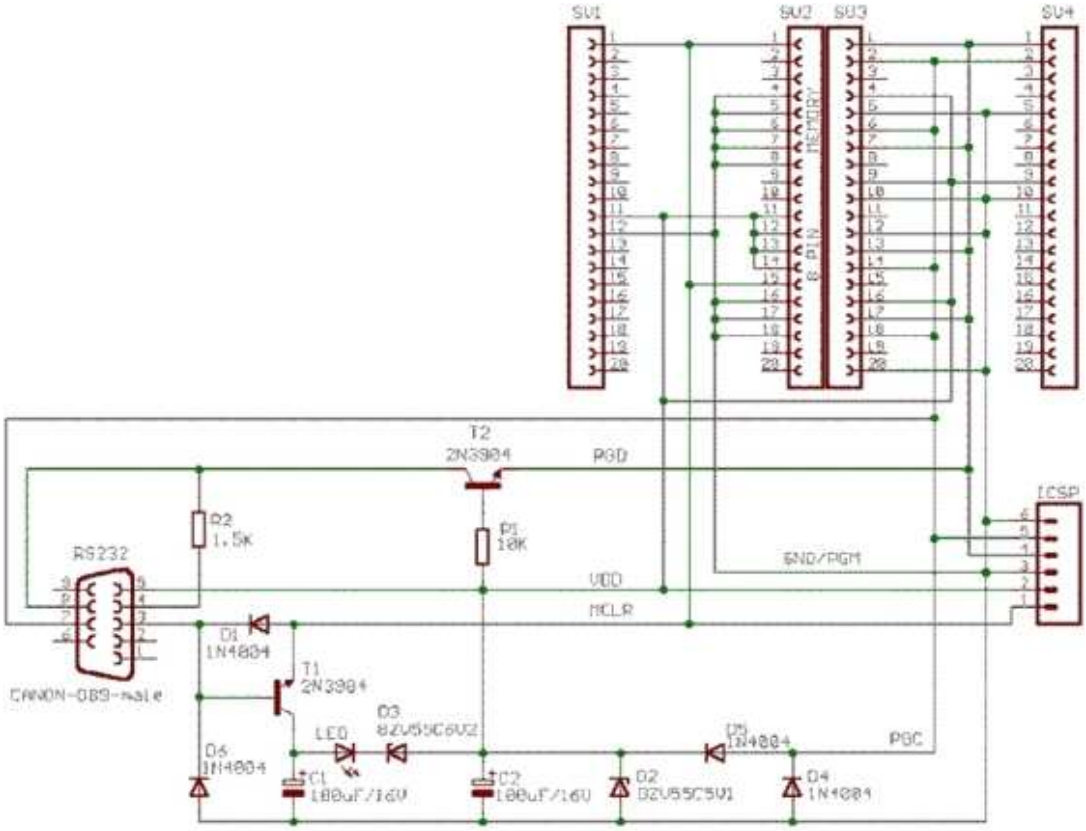


Fig 4.6 Programador serie

Para poder programar el PIC, se debe usar el software PicPgm (http://www.members.aon.at/electronics/pic/picpgm/index.html).

La simpleza del circuito nos permitió comenzar rápidamente con la programación y prueba del PIC, sin embargo esta herramienta no nos iba a servir a futuro cuando comenzáramos con la fabricación de la placa, ya que nos exigía extraer el microcontrolador para su programación, y no contaba con una herramienta de depurado, lo que creemos indispensable para la etapa de desarrollo e implementación.

CUI | Create USB Interface

Este pequeño circuito nos daba la posibilidad de programar el PIC vía USB. Se trata de pocos componentes, y de un firmware que debe correr en el PIC para permitir la conexión USB. Cabe aclarar que en una primera etapa, para poder cargarle dicho firmware al PIC, se debe poseer de un programador standard. Una vez que el PIC esta programado con este *programa base* se puede cargar nuevos programas en el sin necesidad de un programador. Esto nos permite realizar actualizaciones del firmware en el PIC sin necesidad de que nosotros o el usuario final posea un programador.

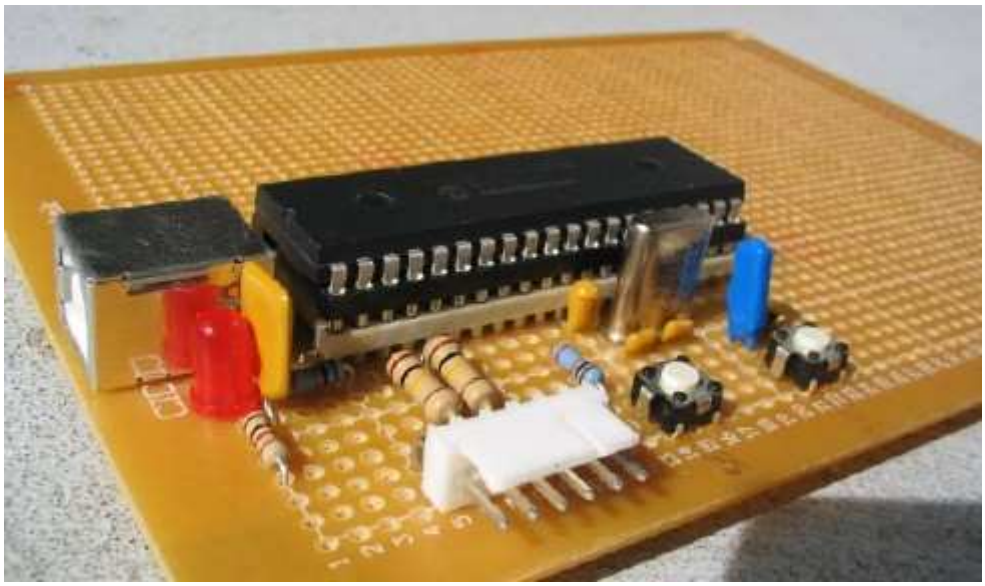


Fig 4.7 Create USB Interface

ICD2 | In Circuit Debugger

Este equipo nos permite la programación del PIC, pero adicionalmente permite depurar el programa directamente dentro del microprocesador. De esta forma no se trabaja sobre una *simulación* de cómo podría funcionar el sistema, sino que efectivamente se trabaja en tiempo real sobre el sistema real.

Hemos elegido el *Easy ICD2*, el cual es un ICD2 completamente compatible y similar al fabricado por Microchip, pero a un costo mucho menor.

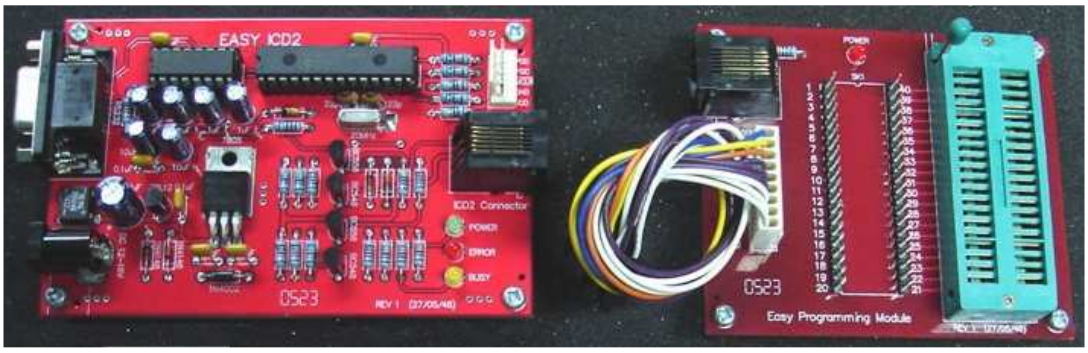


Fig 4.8 In Circuit Debugger

Como se puede observar, el dispositivo consta de dos partes. La primera y principal es la que provee la interfaz hacia el PC, permitiendo la comunicación y programación. La segunda placa es simplemente una interfaz de conexión con el PIC. Esto permite, tal cual luego hemos hecho, realizar la conexión y programación directamente sobre nuestra placa de desarrollo, sin tener que extraer el PIC y colocarlo en el zócalo de programación. Es decir, realizando las conexiones pertinentes, se puede conectar directamente el programador a nuestro circuito y programar y depurar directamente allí. Esta es la razón principal por la cual hemos escogido esta herramienta.

Características principales:

- Interfaz ,S-232 para conexión al PC
- Depurado en tiempo real
- Firmware actualizable desde el PC
- LEDs indicadores de diagnostico (Power, Busy, Error)
- Depurado con detenciones programadas y monitoreo de variables

Etapa de entrada y acondicionamiento de señal

Diseño

La etapa de entrada consta de 2 amplificadores operacionales, y un selector de rango de voltaje, o mejor dicho, de ganancia de la etapa.

Directamente de la entrada de medición, la señal entra en el primer amplificador operacional. Este se encarga de disminuir los voltajes de la señal de entrada, es decir, tiene ganancia mucho menor que la unidad para adaptar una señal de entrada que se encuentre sobrepasando los niveles máximos de tensión aceptable.

También este primer amplificador se encarga de sumar una tensión media a la señal de entrada, de modo de que luego esta señal se encuentre centrada en el valor de tensión medio entre los rangos de entrada de los conversores A/D.

La señal de entrada original se supone centrada en 0v y el osciloscopio debería ser capaz de medir señales tanto positivas como negativas. Dado que la alimentación de todos los circuitos es de 0-5v, para lograr medir señales que sean negativas a la entrada del osciloscopio debemos de sumarle una tensión continua, la cual haría que una entrada de 0v se encuentre en el valor de tensión que se encuentre justo en el medio del rango de voltajes del conversor A/D.

Una vez que la señal es "pequeña" y esta centrada, pasa a un segundo amplificador, el cual tiene una ganancia bastante alta para adaptar esta señal a valores de tensión que maximicen el rango de conversión del ADC.

La ganancia de este segundo amplificador esta dada por la selección de resistencias en su nodo de realimentación. La selección de la resistencia a utilizar esta comandada por patas de control del PIC.

Se trata de 2 patas de control, las cuales nos dan un total de 4 posibles selecciones.

Estas 2 señales de control del PIC van a un decoder/demux, el cual a partir de estas 2 señales genera 4, las cuales son mutuamente excluyentes. Luego estas van a un juego de 4 llaves analógicas. De este modo, solo una de las 4 llaves estará seleccionada a la vez.

Es aquí donde se selecciona el camino que seguirá el nodo de realimentación y ganancia del segundo amplificador, seleccionando a través de cada uno de los

4 caminos una resistencia en particular, de modo de seleccionar la ganancia deseada de esta etapa.

Por más información, ver los esquemáticos de la etapa de entrada en el Apéndice 1.

Componentes utilizados

Amplificador operacional: MAX477

En un principio habíamos elegido al MAX477 (ver *selección de componentes*). Este amplificador no lo habíamos utilizado sino hasta casi llegado el final del proyecto, ya que dados los tiempos que se han manejado, no habíamos llegado al diseño e implementación de la etapa de entrada.

Al probar el funcionamiento del MAX, hemos experimentado que, dada su altísima ganancia, este provocaba oscilaciones en su salida. También el hecho de amplificar con gran ganancia señales pequeñas, la distorsión que se obtenía a su salida era muy considerable. Esta distorsión era básicamente oscilaciones pequeñas sobre la señal original.

Al cambiar este operacional por un 741 (muy popular dentro del ambiente electrónico), observamos que su comportamiento se acercaba más al esperado. Las oscilaciones ya no existían y por lo tanto se obtenía una señal mucho mas limpia. Sin embargo la ganancia era mucho menor que la del MAX, y esto provocaba que las resistencias elegidas para la selección de rangos prácticamente no provoquen diferencia alguna en la ganancia de este segundo amplificador, en especial a altas frecuencias. Esto implicaba tener una señal mucho mas limpia a la salida de la etapa de entrada pero sin control sobre la selección de rangos.

Así también, al utilizar el 741 el ancho de banda se ve drásticamente reducido, ya que este operacional no esta preparado para trabajar a las altas frecuencias a las cuales va a ser sometido el osciloscopio. Además, al tener un ancho de banda menor, la ganancia de este operacional es modificada de acuerdo a la frecuencia de la señal de entrada, haciendo que a partir de una frecuencia media se comporte de forma no-lineal.

Luego hemos optado por probar con otro operacional, en este caso, un OP37 de Texas Instruments, pero sin lograr un resultado satisfactorio.

Finalmente hemos probado con una combinación de 741 y MAX, pero dejamos este tema como mejora pendiente para el futuro (ver *Mejoras pendientes*).

Decodificador binario: 74HCT139

Este decodificador binario cumple 2 funciones básicas: decodificar un numero binario representado en 2 bits en un juego de 4 bits de control, y, adicionalmente (por la naturaleza de un decodificador), estos bits son mutuamente excluyentes, necesario para el control de las llaves analógicas.

Es necesario utilizar inversores a la salida de estas señales de control ya que el decodificador controla señales *activo-bajas*, mientras que el integrado de llaves analógicas utiliza una lógica *activo-alta*. El inversor en cuestión, el 74HC240 es un inversor de 8 pares de entradas/salidas.

Llaves analógicas: 74HC4066

Este integrado contiene 4 llaves analógicas. Cada una de estas llaves consta de un par de transistores que permiten o no la comunicación directa entre sus 2 puntas y es habilitado mediante una pata de habilitación independiente para cada uno de estos circuitos.

Cuando esta llave se encuentra habilitada, su resistencia es de aproximadamente 50 ohm, mientras que cuando se encuentra deshabilitada, su resistencia tiende a infinito. Todas estas compuertas son independientes, y es por esto que se necesita del decodificador, el cual hace que las señales de control sean mutuamente excluyentes, de modo que nunca pueda estar seleccionado más de un canal simultáneamente.

Un detalle a tener en cuenta es que los voltajes en sus pines de entrada/salida no pueden superar por mucho a Vcc ni caer muy por debajo de tierra. Esto implicaría una conducción forzada de los transistores.

Selección de componentes

Recordamos lo mencionado en las características del PIC: Los puertos de entrada/salida del PIC son todos compatibles con la tecnología TTL. Cuando se los utiliza como salida, se comporta como un CMOS, siendo compatible con TTL, de modo de poder manejar cualquier tipo de tecnología. Sin embargo cuando son configurados los puertos como entrada, hay dos comportamientos posibles: puede ser exclusivamente TTL, o puede ser configurado para TTL o CMOS. Dado que ciertos puertos de entrada son solamente compatibles con la tecnología TTL, es que se ha optado por realizar toda la circuitería con tecnología TTL. Vale destacar que la única excepcion a esto es la etapa de entrada, en donde se han utilizado componentes CMOS, algunos con compatibilidad TTL y otros no. Esto se ha dado de este modo por la disponibilidad de los componentes, pero previo a una cuidadosa revisión para asegurar de que no existan problemas.

Otra de las razones por la cual se ha elegido a TTL, y tan importante como la mencionada, es la velocidad de las compuertas.

A modo de ejemplo podemos tomar las compuertas NAND. En tecnología TTL (74F00), el tiempo de propagación de entrada a salida típico es de 3ns. Esto mismo para tecnología CMOS (74HC00) es de 7ns y 10ns para los compatibles con TTL (74HCT00).

1. Conversor analógico-digital (ADC) | Tecnologías

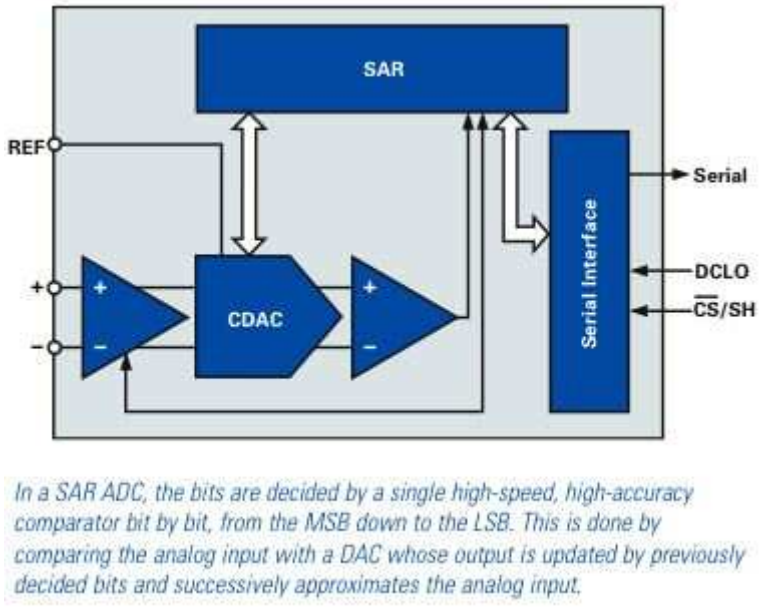
Aproximaciones sucesivas | SAR

Los conversores por registros de aproximaciones sucesivas (SAR - *successive approximation register*) son frecuentemente la arquitectura elegida por las aplicaciones de media a alta resolución a tasas de muestreo medias. Los conversores SAR tienen una resolución entre los 8 y 18 bits, y usualmente no superan las 10 millones de muestras por segundo (10MSPS). Una de sus ventajas es su bajo consumo.

Este tipo de conversores funcionan de una manera similar a una balanza de escalas, como las antiguas. Es decir, de un lado se coloca el peso desconocido, y del otro se van colocando diferentes pesos conocidos, hasta que se logra el equilibrio. Finamente, el peso desconocido se obtiene por la suma de los pesos conocidos colocados en el otro plato de la balanza. De igual forma ocurre con los ADC del tipo SAR. El voltaje analógico desconocido a la entrada es comparado con diferentes tensiones sucesivas generadas por el ADC. Una vez completadas todas las comparaciones, el resultado de cada comparación es exactamente la salida que entrega el conversor. Sin embargo, al tratarse de un componente electrónico de alta velocidad, estas comparaciones ocurren mucho mas rápido que lo que esperamos de una balanza real.

Dado que estos tipos de conversores utilizan la técnica del *sample & hold* (muestreo y retención), la arquitectura en ningún momento asume nada sobre la naturaleza de la señal de entrada, y por lo tanto esta señal no tiene que ser continua. Esto hace que los SAR sean una arquitectura ideal para aplicaciones donde se debe muestrear muchas señales y se utiliza un multiplexor a la entrada del mismo, o bien cuando las muestras no son tomadas una seguida de la otra sino que son tomadas cada algunos segundos o mas, o también donde se requiera una conversión rápida.

El tiempo de conversión se mantiene constante en todos los casos, y tiene una demora desde la adquisición hasta la conversión comparada a los conversores del tipo *pipeline* o *delta-sigma*. Los conversores SAR son ideales para aplicaciones de tiempo real, tales como el control industrial, control de motores, instrumentos portables o a batería, y equipos de adquisición de datos o señales.



In a SAR ADC, the bits are decided by a single high-speed, high-accuracy comparator bit by bit, from the MSB down to the LSB. This is done by comparing the analog input with a DAC whose output is updated by previously decided bits and successively approximates the analog input.

Fig 4.9 Diagrama del conversor SAR

Delta-Sigma

Los conversores *delta-sigma* se destacan por su alta resolución, y son ideales para la conversión de señales con un ancho de banda amplio (desde tensión continua hasta una frecuencia de algunos mega ciclos). Básicamente, estos conversores la señal de entrada es sobremuestreada (oversampling) por un modulador y luego filtrada y decimada por un filtro digital, produciendo una conversión de muy alta resolución a tasas de muestreo relativamente bajas.

La conversión propuesta por los conversores *delta-sigma* permiten que *la resolución pueda ser negociada por velocidad o consumo*. Es decir que si se precisa mucha resolución en la conversión, entonces el dispositivo será mas lento y consumirá mas potencia, mientras que si se requiere menos definición, entonces se pueden lograr tiempos de conversión más bajos o un consumo de potencia más bajo. Además, muchos de estos dispositivos permiten que este comportamiento pueda ser programado. Esto hace que este tipo de conversores sea muy flexible y permita en un mismo aparato diferentes tipos de uso de acuerdo a los requerimientos.

Dado que estos conversores sobremuestrean la señal de entrada, pueden lograr un filtrado *anti-aliasing* en la etapa del filtrado digital. Los diseños modernos con técnicas VLSI (Very Large Scale Integration, integración en escala muy grande) han llevado el costo de filtros digitales complejos muy por debajo del costo de su equivalente analógico. Por ejemplo, el filtrado de ruido de línea simultaneo en 50Hz y 60Hz que antes no era provisto, ahora se tiene integrado directamente en estos conversores.

Entre las aplicaciones típicas para los conversores *delta-sigma* se encuentran el audio, procesos de control industrial, e instrumentos médicos entre otros.

Investigaciones e innovaciones recientes en cuanto a las arquitecturas de los ADC han conducido a una arquitectura donde se usan los principios de

oversampling y pipeline simultáneamente. Estos conversores de alta velocidad ahora permiten conversiones en el rango de los MSPS (millones de muestras por segundo) manteniendo la alta resolución que antes se obtenía pero a baja velocidad, o incluso una resolución aún mayor. Estas innovaciones permiten que con conversores con tanta definición y de tan alta velocidad puedan ser aplicados en comunicaciones y en la proyección de imágenes en la medicina.

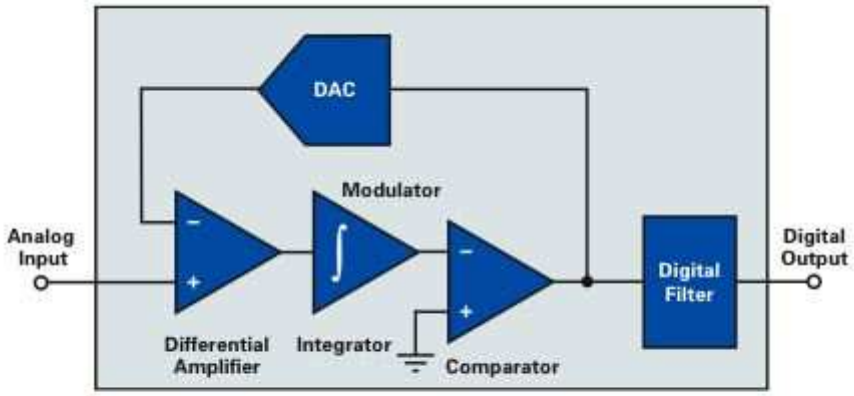
Prácticamente todos los conversores *delta-sigma* tienen entradas diferenciales. Esto significa que en realidad la medición se toma por la diferencia de voltaje entre las 2 entradas, en vez de la diferencia entre un voltaje y tierra (0v). La estructura de las entradas diferenciales permiten que estos conversores sean ideales para medir fuentes del tipo de las termocuplas o sensores del tipo puente, en donde no existe un voltaje común, sino que el mismo dispositivo genera un diferencial de potencial entre sus bornes. En la mayoría de los casos no se precisan amplificadores de entrada para estas aplicaciones.

A diferencia de los conversores SAR, donde la señal de entrada es muestreada y luego analizada para obtener el resultado, los *delta-sigma* miden la señal de entrada durante un pequeño lapso de tiempo y luego a su salida se obtiene en código digital el valor promedio de la señal durante ese tiempo. Es importante recordar la forma en que estos conversores operan, particularmente en diseños que incorporan multiplexación y sincronización.

Es relativamente fácil (y es una práctica común) sincronizar varios conversores *delta-sigma* para que muestreen simultáneamente, pero es mas difícil sincronizar uno de estos conversores con un evento externo. Los conversores *delta-sigma* son poco sensibles al *jitter* en los pulsos de reloj (variación de su periodo), y esto esta dado por el hecho de que el *oversampling* efectivamente promedia las demoras de estos pulsos y por lo tanto logra una reducción del impacto del *jitter* en el ruido.

Muchos de estos conversores incluyen entradas con buffers (adaptadores de impedancia) y amplificadores de ganancia programable (PGA - programmable gain amplifiers). Los buffers de incrementan la impedancia de entrada permitiendo una conexión directa con una fuente cuya salida sea también de alta impedancia, evitando la necesidad de componentes intermediarios. La ventaja de poseer un PGA interno es que cuando se mide una señal de poca amplitud estos amplificadores logran que se pueda obtener la misma resolución que cuando se tratan señales de mayor amplitud. Los sensores del tipo puente son un claro ejemplo de una fuente de señal que puede aprovechar las ventajas de los PGA dentro del conversor.

Todo ADC requiere de una referencia para la señal de entrada, pero en especial para los de alta resolución, el bajo ruido y la baja deriva son críticos, y es por esto que la mayoría de los conversores *delta-sigma* tienen entradas diferenciales.



Delta-sigma ADCs consist of a delta-sigma modulator followed by a digital decimation filter. The modulator incorporates a comparator and integrator in a feedback loop with a DAC. The loop is synchronized by a clock.

Fig 4.10 Diagrama del conversor Delta-Sigma

Pipeline

La mayoría de los conversores del rango de las decenas de millones de muestras por segundo están basados en una arquitectura del tipo *tubería* (pipeline). Los conversores *pipeline* consisten en "N" etapas en cascada. La operación continua de todas las etapas de la tubería hacen que este tipo de arquitectura alcance velocidades de muestreo altas. Cada una de estas etapas son idénticas en su esencia, alineadas una detrás de otra, y diseñadas para convertir sólo una parte de la muestra analógica de entrada. El resultado digital de la comparación hecha por cada una de las etapas es alineada luego para obtener la salida en paralelo de estos resultados. En cierta forma, sería como colocar tantos ADC de 1 bit de resolución como bits de resolución se deseen obtener. Por cada ciclo de reloj se obtiene una nueva muestra. Sin embargo, dado este tipo de construcción, es evidente que se tiene un retardo desde que se tomó la muestra hasta que se obtiene la salida, pero en la mayoría de las aplicaciones esto no es una limitación ya que dicho retardo, expresado en ciclos de reloj, es constante y conocido.

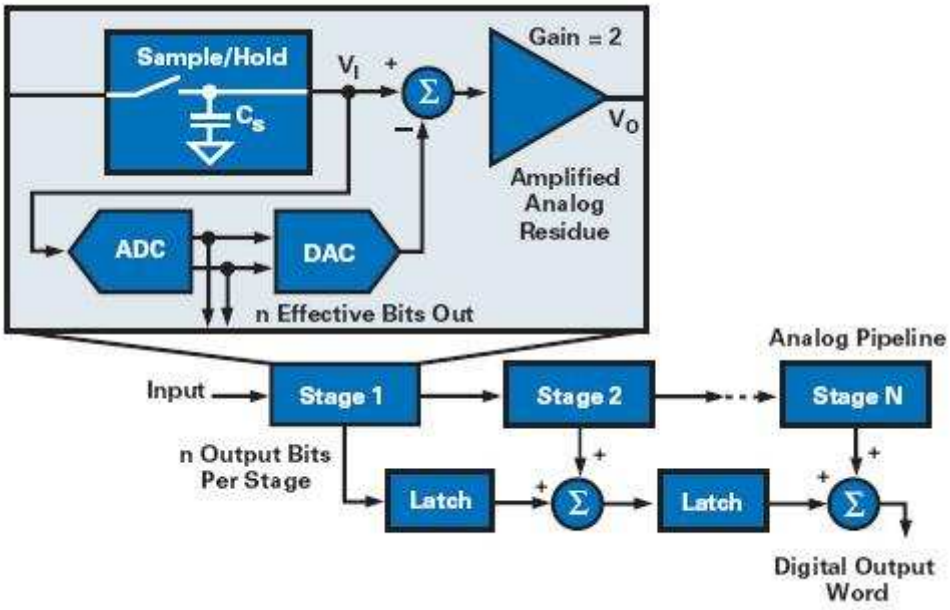


Fig 4.11 ADC tipo Pipeline

Una de las características principales que permiten que los ADC tipo *pipeline* tengan un desempeño dinámico tan bueno a altas frecuencias radica en que la señal de entrada sea del tipo diferencial. Esta configuración de entrada tiene como resultado optimizar su rango dinámico, dado que esto lleva a señales de menor amplitud y una reducción en los armónicos de orden par. La mayoría de los conversores *pipeline* de alta velocidad usan una fuente de alimentación simple, haciendo que sea necesario que la señal de entrada opere en *modo común*, y que este valor se encuentre típicamente en el medio del voltaje de alimentación. Este requerimiento de *modo común* debe entrar en consideración cuando se define la circuitería de entrada.

Los conversores del tipo *pipeline* son los mas utilizados cuando se trata de muestrear a velocidades de entre unos pocos MSPS y los 100MSPS. La complejidad de esta arquitectura crece linealmente (no exponencialmente) con la cantidad de bits de resolución que se exigen, y esto es justamente debido al tipo de construcción en tubería. Con característica se logran tener conversores de alta velocidad y alta resolución, aún manteniendo un bajo consumo. Los *pipeline* son útiles en un amplio rango de aplicaciones, más notablemente en el área de las comunicaciones digitales, donde la performance dinámica del conversor suele ser más importante que las especificaciones tradicionales sobre tensión continua, como ser la no-linealidad diferencial (DNL - differential non-linearity), y la no-linealidad integral (INL - integral non-linearity).

Los conversores *semi-flash* utilizan una arquitectura de varios *pipelines* integrados en una *flash* para lograr una mejor performance que la *flash* en cuanto a velocidad y reduciendo drásticamente su consumo.

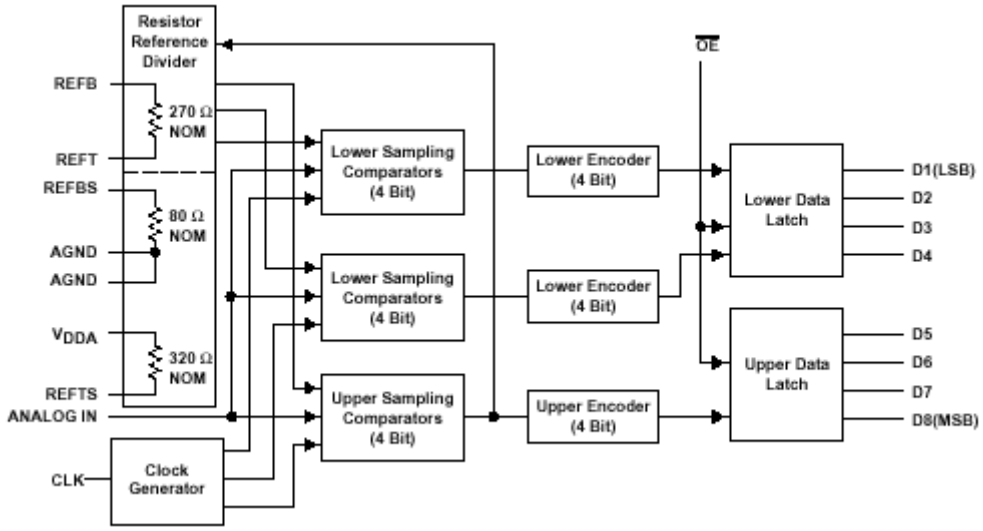


Fig 4.12 Semiflash ADC

Flash

Los conversores del tipo *flash*, también conocidos como conversores paralelos, son la arquitectura más rápida en conversores A-D.

Para aplicaciones que no requieren una resolución muy alta, sino mas bien media, típicamente 8 bits, pero que tengan la capacidad de muestrear señales de cientos de MHz o aún mayores, la arquitectura flash puede que sea la única alternativa viable. Sin embargo estos conversores tienen un consumo de potencia mayor que las otras arquitecturas y también tienen un costo mayor, lo que hace que estos conversores sean usados exclusivamente para altas frecuencias y donde no pueda ser usado otro tipo de arquitectura.

Las aplicaciones típicas para estos conversores son:

- Adquisición de datos
- Comunicaciones satelitales
- Procesamiento de radares
- Discos de datos de alta densidad

Cómo su nombre lo indica, utiliza una arquitectura de conversión en paralelo. Cada comparador representa 1 LSB *(Least Significant Bit)*, y su funcionamiento se puede comparar a un termómetro de mercurio, donde la columna de mercurio aumenta hasta el valor de apropiado de temperatura. Se puede observar en el siguiente gráfico dicho comportamiento, donde el valor de tensión de una señal generaría una salida de los comparadores del estilo *"0000111111111"*, y luego esta salida es enviada a un codificador binario tal que se represente en potencias de 2 el valor obtenido. Existen tantos comparadores como divisiones o resolución se deseen, es decir que si se desean *N* bits, entonces el conversor tendrá 2^N-1 comparadores. El siguiente gráfico explica la arquitectura:

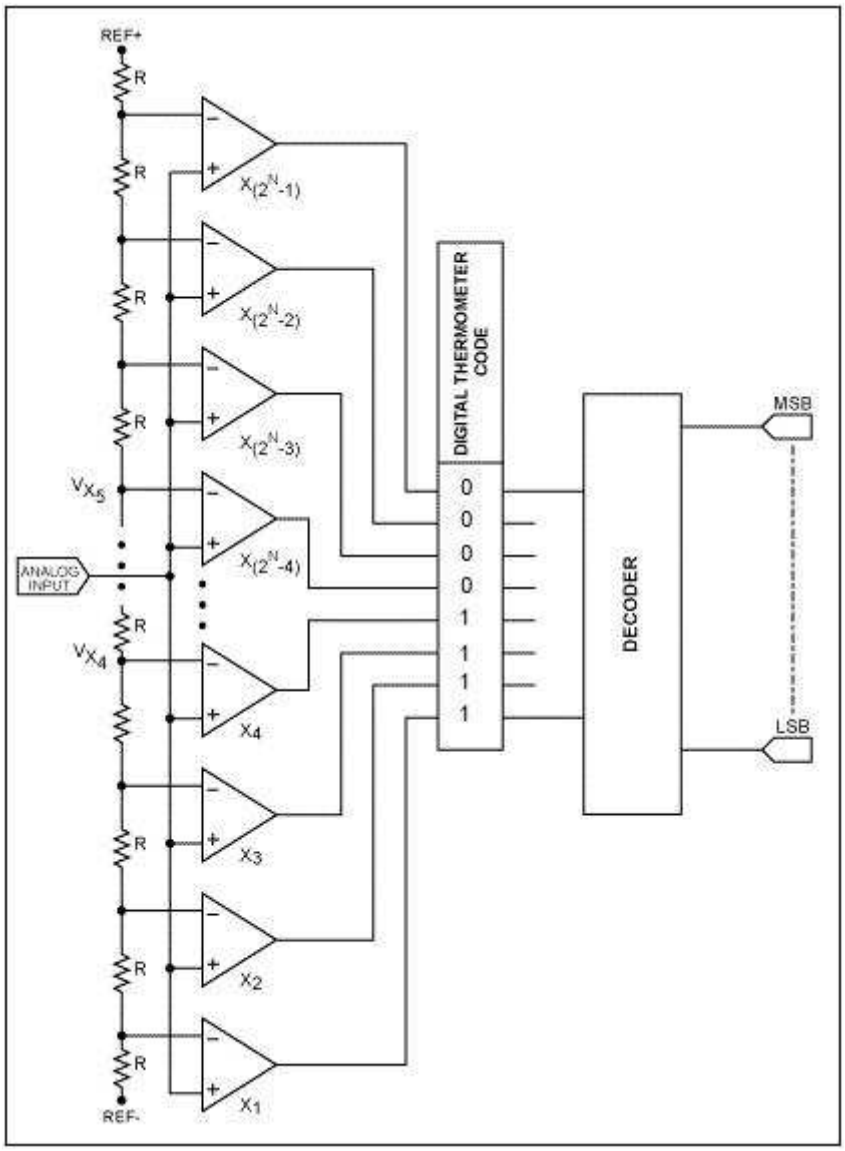


Fig 4.13 ADC tipo Flash

Dado que todos los conversores actúan en paralelo, solo se requiere de un tick de reloj para convertir la señal de entrada. El voltaje de referencia de cada conversor es de 1 LSB mayor al del comparador que lo precede, siendo para el primer comparador exactamente 1 LSB.

Este tipo de conversores requiere que el *jitter* del reloj sea lo mas pequeño posible para asegurar la mejor performance. De acuerdo a las especificaciones de cada caso, se debe utilizar un *track & hold*, el cual en la mayoría de los casos es necesario. Existen conversores de este tipo que ya incluyen esta funcionalidad dentro del mismo integrado.

Conversores integrados | Integrating ADC

Dada la baja velocidad de conversión de esta arquitectura, este tipo de conversores no será estudiado.

Comparación y elección

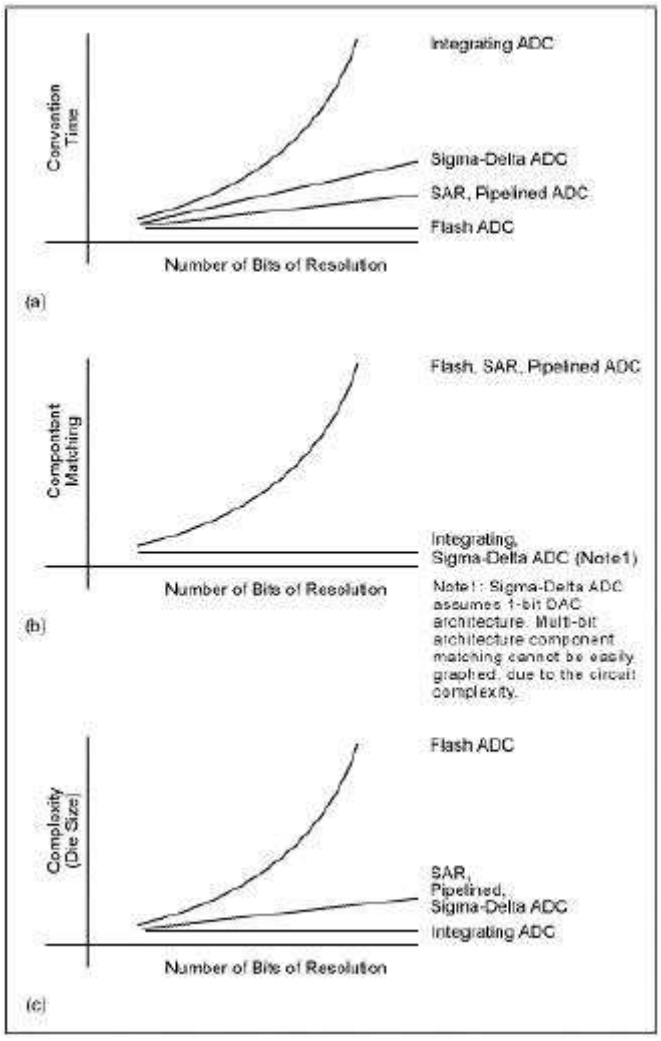


Fig 4.14 Comparación ADC

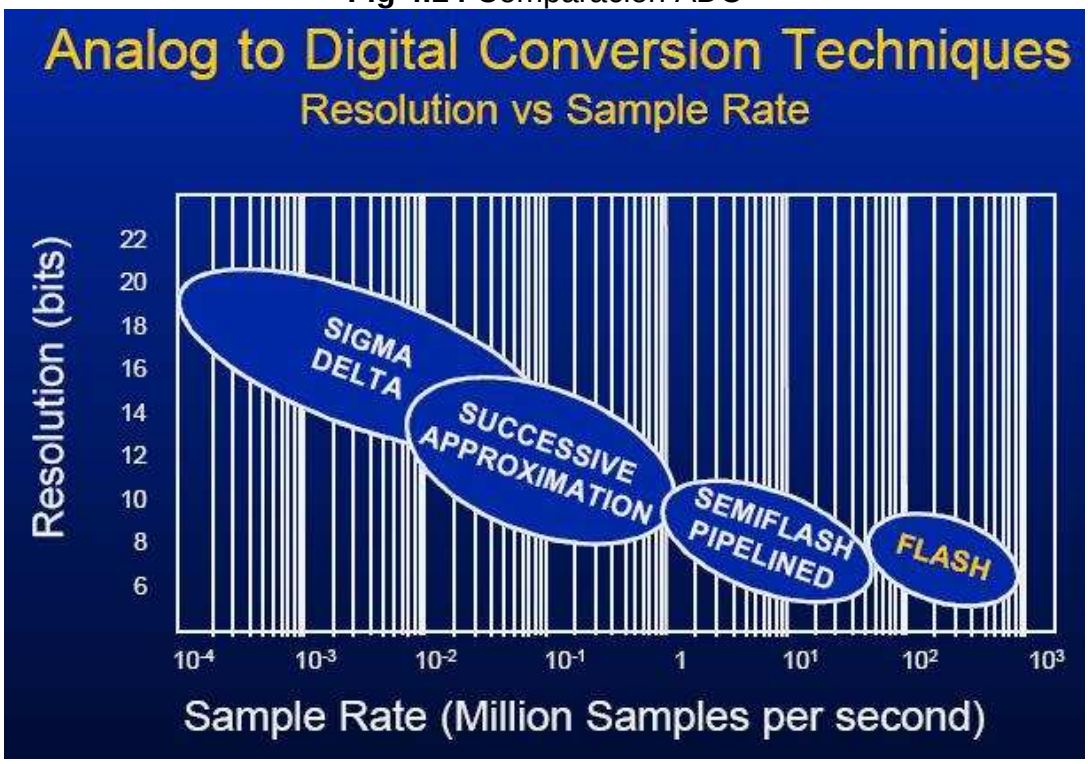


Fig 4.15 Resolución Vs. Velocidad de muestreo

	SAR	Delta-Sigma	Pipeline	Flash
Velocidad	~ 10MSPS	< 1MSPS	~ 100MSPS	> GSPS
Resolución	hasta 16 bits	entre 10 y 16 bits	entre 8 y 14 bits	hasta 8~10 bits
Consumo	Bajo (~10mW)	Muy bajo (~1mW)	Medio (~100mW)	Alto (~1000mW)

Tabla 4.1 Comparación de tipos de muestreo

- Texas TLC 5540
  - 8-Bit Resolution
  - Differential Linearity Error
    - ±0.3 LSB Typ, ±1 LSB Max (25℃)
    - ±1 LSB Max
  - Integral Linearity Error
    - ±0.6 LSB, ±0.75 LSB Max (25℃)
    - ±1 LSB Max
  - Maximum Conversion Rate of 40 Megasamples Per Second (MSPS) Max
  - Internal Sample and Hold Function
  - 5-V Single Supply Operation
  - Low Power Consumption: 85 mW Typ
  - Analog Input Bandwidth: ≥75 MHz Typ
  - Internal Reference Voltage Generators
  - Hoja de datos: ver apéndice III.
- Texas ADS831
  - High SNR: 49dB
  - Internal or external reference option
  - Single-ended or differential analog input
  - Programmable input range: 1Vp-p /2Vp-p
  - Low Power Consumption: 275mW
  - Low DNL: 0.35LSB
  - Single +5V supply operation
  - SSOP-20 Package
  - Hoja de datos: ver apéndice III.
- Texas THS0842
  - Dual Simultaneous Sample and Hold Inputs
  - Differential or Single-Ended Analog Inputs
  - 8-Bit Resolution 40 MSPS Sampling Analog-to-Digital Converter (ADC)
  - Single or Dual Parallel Bus Output
  - Low Power Consumption: 275 mW Typ Using External References
  - Wide Analog Input Bandwidth: 600 MHz Typ
  - 3.3 V Single-Supply Operation
  - 3.3 V TTL/CMOS-Compatible Digital I/O
  - Internal or External Bottom and Top Reference Voltages
  - Adjustable Reference Input Range
  - Power-Down (Standby) Mode
  - Hoja de datosver apéndice III.
- Analog Devices AD9057
  - 8-Bit, Low Power ADC: 200 mW Typical
  - 120 MHz Analog Bandwidth

    - On-Chip 2.5 V Reference and Track-and-Hold
    - 1 V p-p Analog Input Range
    - Single 5 V Supply Operation
    - 5 V or 3 V Logic Interface
    - Power-Down Mode: <10 mW
    - 3 Performance Grades
        - (40 MSPS, 60 MSPS, 80 MSPS)
    - Hoja de datos: ver apéndice III.
- Analog Devices AD9059
    - Dual 8-Bit ADCs on a Single Chip
    - Low Power: 400 mW Typical
    - On-Chip 2.5 V Reference and Track-and-Hold
    - 1 V p-p Analog Input Range
    - Single 5 V Supply Operation
    - 5 V or 3 V Logic Interface
    - 120 MHz Analog Bandwidth
    - Power-Down Mode: <12 mW
    - Hoja de datos: ver apéndice III.

Entre los dispositivos seleccionados, hemos llegado a la conclusión que el conversor que utilizaremos es el TLC5540 de Texas Instruments. Las características que nos han hecho decidir por éste son su bajo consumo, su bajo precio, la alta disponibilidad, y como ventaja adicional por sobre el resto de los conversores, encontramos suficiente documentación sobre su uso, e incluso su uso en aplicaciones similares a este proyecto.

La única contra que tiene este conversor, pero que es igual en cualquiera de los casos, es que no se encuentra disponible en encapsulado DIP, lo que prácticamente nos obliga a comprar un adaptador SOIC - DIP. Este zócalo adaptador sólo lo hemos conseguido en Estados Unidos, y lamentablemente no tiene un bajo precio. El tema del precio esta directamente relacionado con que estos casos ocurren solamente en etapa de desarrollo (de lo contrario se tendría el zócalo necesario para dicho encapsulado), la cuales obviamente no es un caso común ni masivo. Sin embargo consideramos a este problema como temporal, ya que de concluir satisfactoriamente el diseño y la prueba en la placa, se construirá un circuito impreso en donde se va a prever el encapsulado de cada componente sin necesidad de zócalos adaptadores.

2. Memoria

El funcionamiento de una memoria esta basado en celdas y el interior de cada chip se puede imaginar como una matriz o tabla en la cual cada celda es capaz de almacenar un bit. Es decir, que las memorias se basan en celdas para almacenar cada bit, y dichas celdas están organizadas en *arreglos*, tal sería la forma de una matriz, en donde se tienen filas y columnas, y cada celda tiene una ubicación única, descripta por el numero de columna y numero de fila. El numero que identifica a cada ubicación se conoce como *dirección*. Luego, a partir de una dirección se calcula cuál es la fila y columna correspondiente, con lo que ya se puede acceder a la celda deseada.

Las memorias de RAM *(Random Access Memory)* son memorias volátiles, esto significa que se pierde la información cuando no se le brinda alimentación y se clasifican en dos categorías básicas: la RAM estática y la RAM dinámica, las cuales se describen en las siguientes secciones.

Para este tipo de memorias, aún cuando su funcionamiento es secuencial y en cada avance de reloj se avanza en un bit la dirección de memoria, se debe utilizar una lógica externa de control en donde dicha dirección se incremente. Ante esta característica, hemos encontrado una memoria que tiene una pequeña lógica interna que permite evitar el uso de componentes externos.

Memorias de direccionamiento autoincrementado

Se trata de una memoria capaz de realizar operaciones lógicas no complejas, como ser el autoincremento de la dirección de memoria a la cual se accede. Este tipo de dispositivo sería realmente útil ya que simplificaría la etapa de control de memoria. Una arquitectura de este tipo fue encontrada en la búsqueda de soluciones, sin embargo, sólo se encuentra fabricada, pero aún no existe como producto. Obviamente, al no ser una posibilidad, hemos analizado las diferentes posibilidades dentro de las memorias standard en el mercado (siguiente apartado).

- RAM 200 (http://www.freepatentsonline.com/4835733.html)

Memoria RAM estática

El componente principal de estas memorias es el *flip-flop*. Se compone de 4 transistores MOSFET o CMOS en un arreglo tal que cuando se le da un valor en una de sus entradas, este valor es conservado hasta que se quite la alimentación o se le cargue un nuevo valor.

Este tipo de memoria conocida como SRAM *(Static Random Access Memory)* se compone de celdas de *flip-flops*. En la siguiente figura se observa la estructura típica de una celda de memoria de una SRAM.

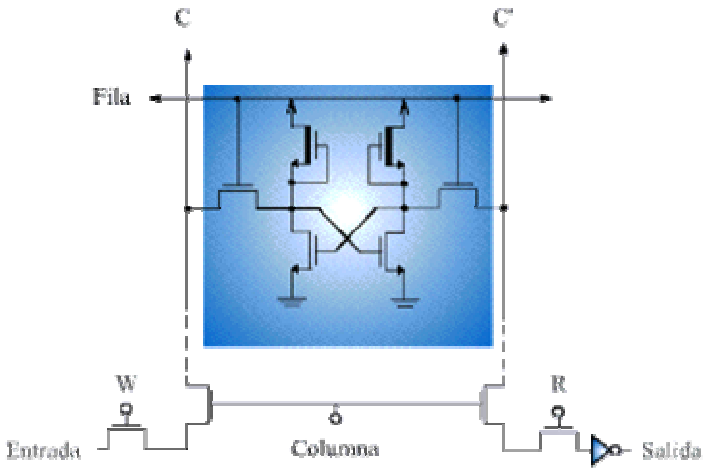


Fig 4.16 Celda SRAM

En la figura se pueden ver las 4 conexiones necesarias. El pin de entrada indica que es allí en donde se coloca el dato que se desea almacenar. Luego un pulso en **"W"** *(Write)* hará que el dato sea cargado en en flip-flop. Finalmente, para volver a obtener el dato guardado, se debe dar tensión en **"R"** *(Read)*, y en la salida tendremos el dato que anteriormente se había almacenado.

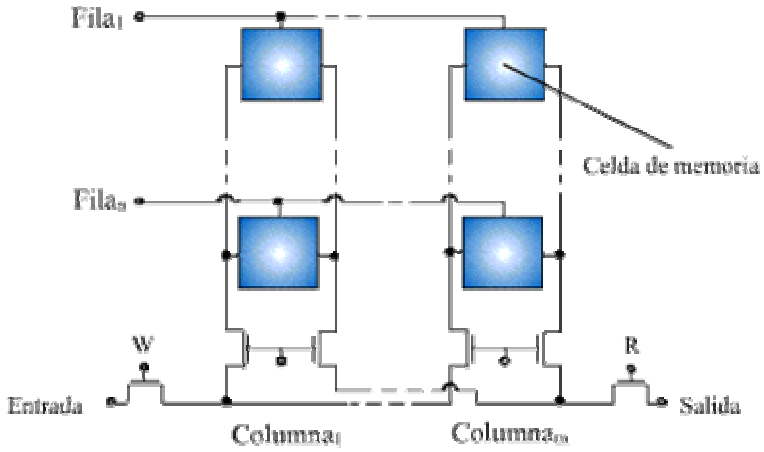


Fig 4.17 Arreglo SRAM

Memoria RAM dinámica

Las memorias DRAM *(Dynamic Random Access Memory)* son similares a las memorias estáticas, pero su diferencia radica en que en vez de utilizar flip-flops, utilizan condensadores. La utilización de condensadores implica que haya que cargarlos, pero también implica que éstos se descarguen.

Es decir, que para el funcionamiento correcto de estas memorias, una vez que se posiciona en la dirección deseada y se le carga el valor que se quiere almacenar, es estrictamente necesario volver a recurrir a la misma dirección después de cierto lapso de tiempo (este tiempo depende exclusivamente de cada memoria) para volver a cargar el capacitor con el dato que éste tenía antes de que por efecto de la descarga, éste pierda el dato almacenado.

El uso de condensadores en vez de transistores hace que su tamaño sea considerablemente menor, haciendo posible la construcción de memorias de mucha mayor capacidad.

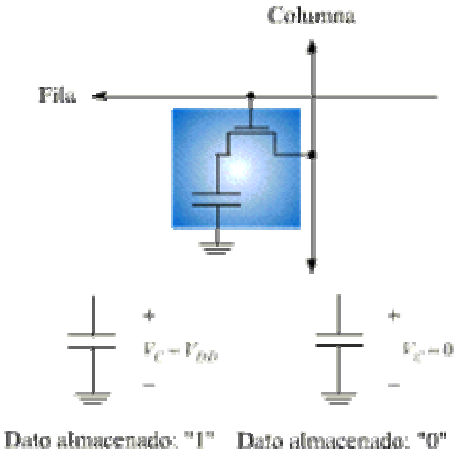


Fig 4.18 Celda DRAM

La operación de la celda es similar a la de un interruptor, cuando el estado en la fila se encuentra en alto, el transistor entra en saturación y el dato presente en el bus interno de la memoria (columna) se almacena en el condensador, durante una operación de escritura y se extrae en una operación de lectura. El inconveniente que tiene este tipo de memorias consiste en que hay que recargar la información almacenada en las celdas, por lo cual estas celdas requieren de circuitería adicional para cumplir esta función. En la siguiente figura se observa la celda completa con sus aditamentos donde se puede identificar la forma en que se desarrollan las operaciones de escritura, lectura y recarga.

La siguiente figura muestra que cuando dicha celda se encuentra seleccionada por la columna y fila correspondiente, entonces un pulso en el bit de recarga hará que el mismo valor que ya tiene (obtenido desde el dato de salida) es vuelto a cargar como entrada de datos y se vuelve a cargar el condensador. La señale R/W *(Read/Write)* habilita a que se cargue el condensador con el valor que se encuentra en el pin de entrada de datos, o bien habilita la lectura mediante el pin de salida de datos con el valor que esta cargado en el condensador. Vale aclarar que si se ha demorado en hacer una recarga de datos y el tiempo límite desde la ultima carga del condensador ha sido superado, entonces el dato que se leerá será erróneo.

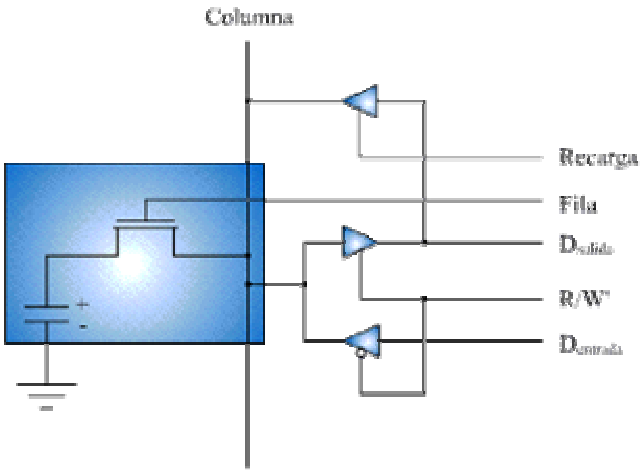


Fig 4.19 Funcionamiento DRAM

Comparación y elección

El primer punto que se debe analizar es si la memoria que utilizaremos será del tipo estática o dinámica. A continuación presentamos un cuadro comparativo de las principales características de una y otra arquitectura.

Memoria	Ventajas	Desventajas
SRAM	La velocidad de acceso es alta	Menor capacidad, debido a que cada celda de almacenamiento requiere mas transistores
	Para retener los datos solo necesita estar energizada	Mayor costo por bit
	Son mas fáciles de diseñar	Mayor consumo de Potencia
DRAM	Mayor densidad y capacidad	La velocidad de acceso es baja
	Menor costo por bit	Necesita recargar de la información almacenada para retenerla
	Menor consumo de potencia	Diseño complejo

Tabla 4.2 Comparación de tipos de memorias

Dado que el costo de los componentes no es alto (básicamente por que su capacidad de almacenamiento no es alta tampoco), utilizaremos memorias estáticas, ya que son de más fácil uso, y no requieren de una lógica externa para que la información guardada se mantenga.

Las características determinantes para la elección de la memoria son su capacidad y su velocidad. Hemos hecho una búsqueda de memorias de diferentes tamaños y velocidades en el mercado, y a continuación destacamos cada una con sus características principales:

- Texas BQ4011
  - Data retention in the absence of power
  - Automatic write-protection during power-up/power-down cycles
  - Industry-standard 28-pin 32K x 8 pinout
  - Conventional SRAM operation; unlimited write cycles
  - 10-year minimum data retention in absence of power
  - Battery internally isolated until power is applied
  - Hoja de datos: ver apéndice III.
- Cypress CY7C199
  - High speed — 10 ns
  - Fast tDOE
  - CMOS for optimum speed/power
  - Low active power — 467 mW (max, 12 ns “L” version)
  - Low standby power - 0.275 mW (max, “L” version)
  - 2V data retention (“L” version only)
  - Easy memory expansion with CE and OE features

    - TTL-compatible inputs and outputs
    - Automatic power-down when deselected
    - Hoja de datos: ver apéndice III.
- Cypress CY7C109B
    - High speed — tAA = 12 ns
    - Low active power — 495 mW (max. 12 ns)
    - Low CMOS standby power — 55 mW (max.) 4 mW
    - 2.0V Data Retention
    - Automatic power-down when deselected
    - TTL-compatible inputs and outputs
    - Easy memory expansion with CE1, CE2, and OE options
    - Hoja de datos: ver apéndice III.
- ALSC AS7C256A
    - Industrial and commercial temperature options
    - Organization: 32,768 words × 8 bits
    - High speed
        - 10/12/15/20 ns address access time
        - 5, 6, 7, 8 ns output enable access time
    - Very low power consumption: ACTIVE
        - 412.5 mW max @ 10 ns
    - Very low power consumption: STANDBY
        - 11 mW max CMOS I/O
    - Easy memory expansion with CE and OE inputs
    - Hoja de datos: ver apéndice III.

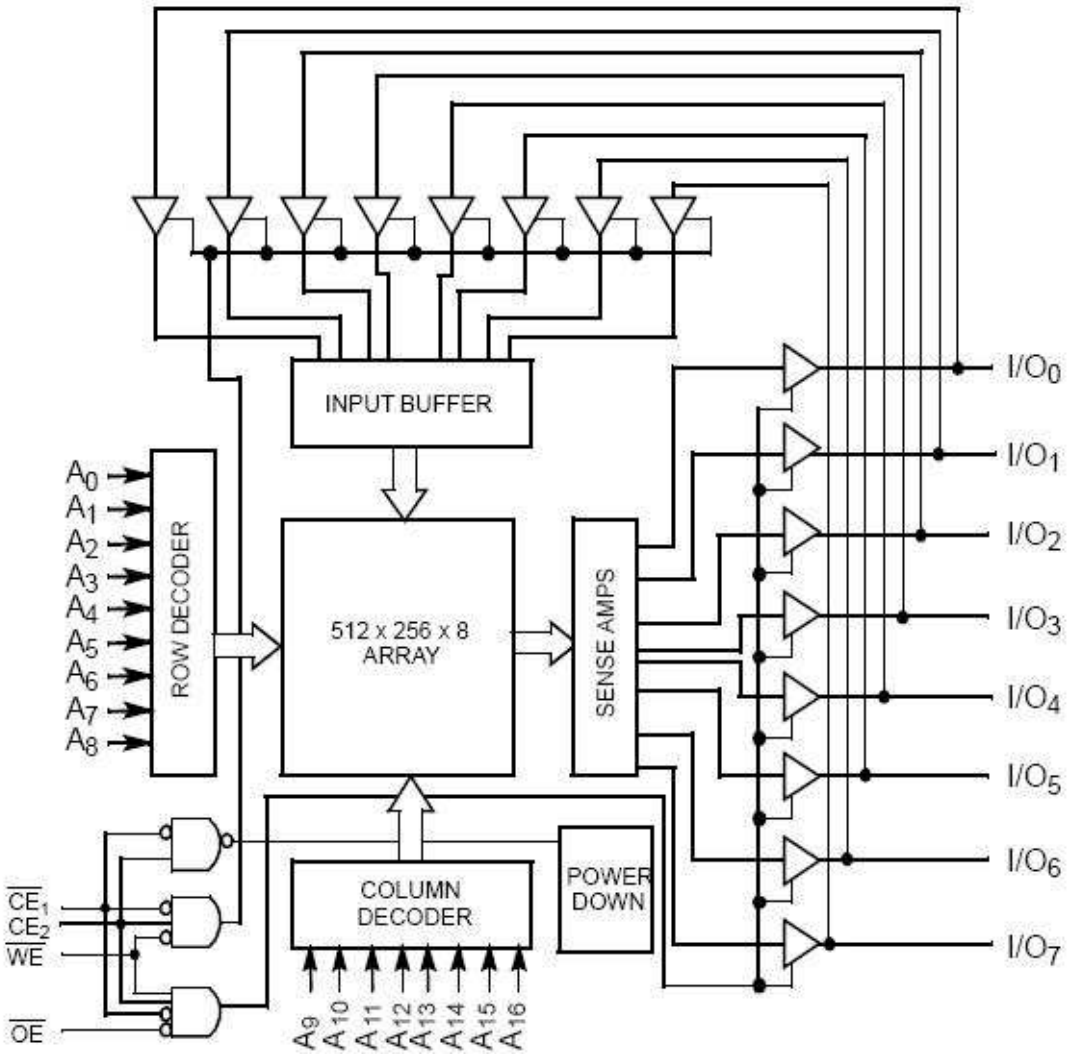


Fig 4.20 Arquitectura de la memoria Cypress CY7C-109B

Una vez que se tiene un estimado de las memorias que se podrían utilizar y su precio, se tuvo que hacer la evaluación del tamaño y la velocidad que el proyecto requería.

Dado que se tiene como objetivo tener una velocidad de trabajo del orden de los 40Mhz, la memoria debe tener tiempos de acceso menores a 20~25 ns.

En cuanto al tamaño que ésta debe tener, consideramos que con mil muestras sería en principio suficiente para el objetivo buscado. Sin embargo, si se considera la opción del disparo por hardware para obtener las muestras, entonces se precisarían mas muestras, para poder tener muestras previas y siguientes sobre un hecho que puede no repetirse, con lo cual, se podría pedir que la memoria sea capaz de almacenar diez mil muestras.

Ahora las opciones serían a partir de los 16K x 8 bits como mínimo. La intención de largo alcance del proyecto y la escalabilidad y flexibilidad deseada, hacen que dentro de lo posible, las características limitantes sean las menores posibles y se puedan tener los mejores componentes. Por esta razón es que a partir de un mínimo de 16K pasamos a tener en cuenta las memorias de 32K. Además, una memoria que exceda los mínimos nos permite tener un registro

mucho mayor sobre cada muestreo o captura que se realiza. Si en vez de mostrar en pantalla lo que se ha capturado, se desea transferirlo a un archivo para su posterior análisis, entonces esta ventaja pasa a ser fundamental, donde una captura pasa a ser prácticamente un historial sobre el muestreo realizado. Una memoria de 32K nos permite una flexibilidad y posibilidad de realizar muchas operaciones sin que el tamaño de la memoria sea una limitante.

Al igual que en el caso de los conversores analógico-digital, entre los encapsulados disponibles no se encuentra el DIP, por lo que será necesario comprar un zócalo adaptador. De todos modos, esto es solamente temporal, porque en el caso de la construcción de una placa impresa (PCB) este problema queda solucionado.

Luego de analizadas las opciones y verificar su precio, hemos observado que la diferencia de costo entre una memoria de 32K y una memoria que cuadriplique su tamaño, es decir 128K, era de aproximadamente un 15% superior, pero en precios tan bajos, esto pasa a ser casi despreciable, por lo que directamente optamos por excedernos en demasía con la memoria y dejar que este componente sea lo suficientemente grande como para que el día que los alcances del proyecto crezcan, no sea una limitante.

Otra razón por la que hemos elegido la memoria de Cypress es su disponibilidad y precio. Luego de buscar en el mercado uruguayo los componentes citados y ver que no había ninguno en plaza, se buscó en Buenos Aires, Argentina. La memoria de Cypress era una de las tres memorias seleccionadas que se podía conseguir en dicho mercado, pero teniendo ventaja en su precio. Es por esta razón por la cual decidimos utilizar la citada memoria. Esta ventaja nos dio tiempo para poder probarla y estudiarla mientras se construía la placa. Además, existe mucha documentación valiosa sobre su uso y funcionalidad.

3. Amplificadores de entrada

Los amplificadores de entrada son utilizados principalmente para separar la etapa de entrada de la de adquisición y hacer una adaptación de impedancias. Esto independiza a estas etapas.

Todos los amplificadores seleccionados tienen características muy similares, y todos son aptos para el proyecto, sin embargo el MAX477 tiene como ventaja el encapsulado DIP, que el resto no lo tiene disponible. Esta característica nos simplifica en costo y tiempo, y es por eso que lo hemos elegido. La estabilidad en cuanto a las variaciones de parámetros *(offsets)* es superior frente al resto, mientras que el ancho de banda y rango de tensiones es similar a los otros amplificadores.

- Maxim MAX477
    - High Speed
        - 300MHz -3dB Bandwidth (AV = +1)
        - 200MHz Full-Power Bandwidth (AV = +1,VO = 2Vp-p)

        - 1100V/µs Slew Rate
        - 130MHz 0.1dB Gain Flatness
    - Drives 100pF Capacitive Loads Without Oscillation
    - Low Differential Phase/Gain Error: 0.01⁰/0.01%
    - 8mA Quiescent Current
        - Low Input-Referred Voltage Noise: 5nV/√Hz
        - Low Input-Referred Current Noise: 2pA/√Hz
    - Low Input Offset Voltage: 0.5mV
    - 8000V ESD Protection
    - Voltage-Feedback Topology for Simple Design Configurations
    - Short-Circuit Protected
    - Hoja de datos: ver apéndice III.
- Texas OPA695
    - GAIN = +2 BANDWIDTH (1400MHz)
    - GAIN = +8 BANDWIDTH (450MHz)
    - OUTPUT VOLTAGE SWING: ±4.2V
        - ULTRA-HIGH SLEW RATE: 4300V/µs
    - RD-ORDER INTERCEPT: > 40dBm (f < 50MHz)
    - LOW POWER: 129mW
    - LOW DISABLED POWER: 0.5mW
    - Hoja de datos: ver apéndice III.
- Texas OPA691
    - FLEXIBLE SUPPLY RANGE:
        - +5V to +12V Single-Supply
        - ±2.5V to ±6V Dual-Supply
    - UNITY-GAIN STABLE: 280MHz (G = 1)
    - HIGH OUTPUT CURRENT: 190mA
    - OUTPUT VOLTAGE SWING: ±4.0V
        - HIGH SLEW RATE: 2100V/µs
    - LOW dG/dφ: 0.07%/0.02°
    - LOW SUPPLY CURRENT: 5.1mA
        - LOW DISABLED CURRENT: 150µA
    - WIDEBAND +5V OPERATION: 190MHz (G = +2)
    - Hoja de datos: ver apéndice III
- Texas OPA830
    - HIGH BANDWIDTH:
        - 250MHz (G = +1)
        - 110MHz (G = +2)
    - LOW SUPPLY CURRENT: 3.9mA (VS = +5V)
    - FLEXIBLE SUPPLY RANGE:
        - ±1.4V to ±5.5V Dual Supply
        - +2.8V to +11V Single Supply
    - INPUT RANGE INCLUDES GROUND ON SINGLE SUPPLY
    - 4.88V OUTPUT SWING ON +5V SUPPLY
    - HIGH SLEW RATE: 550V/ns
        - LOW INPUT VOLTAGE NOISE: 9.2nV/√Hz
    - Hoja de datos: ver apéndice III

Las características que nos han hecho elegir el MAX477 de Maxim son mas bien prácticas que técnicas. Es decir, cualquiera de estos amplificadores

satisfaría los requisitos, sin embargo, el amplificador de Maxim gana en precio, disponibilidad, y por sobre todo, es el único que se encuentra disponible en encapsulado del tipo DIP. Este encapsulado es nuestro preferido y facilita su conexión y posible reemplazo, sin necesidad de utilizar zócalos especiales, los cuales en ciertos casos son difíciles de conseguir y pueden ser muy caros.

4. Contadores

Un contador es básicamente un circuito secuencial temporizado. Un contador binario de *n* bits puede contruirse con *n* flip-flops en cascada. Los flip-flops que se utilizan son flip-flops tipo T, el cual cambia de estado (0 o 1) en cada flanco ascendente de su entrada de reloj. A continuación se puede observar la operación básica de un contador de 4 bits.

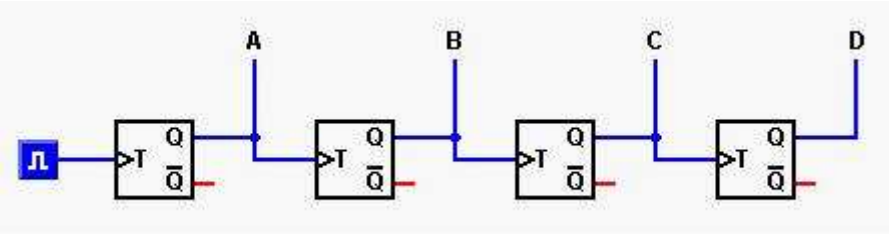


Fig 4.21 Estructura de un contador

Dentro de la familia de contadores, hemos encontrado de hasta 8 bits que trabajen a la frecuencia especificada (40Mhz) y es por esta razón que debimos colocar dos contadores. También consideramos útil que los mismos sean contadores hacia adelante y hacia atrás. Si bien puede que no lo utlicemos en este momento, consideramos que es de buen diseño poder prever futuros usos y dar la posibilidad de expandir a funcionalidad de los componentes. Así también hemos elegido un contador que tenga una señal de habilitación de funcionamiento, es decir, que si el dispositivo no se encuentra habilitado, por mas que tenga señal de reloj y se intente avanzar en la cuenta, no lo va a hacer ya que como se espera, éste no se encuentra habilitado. Dado que el direccionamiento de las memorias es de 16 bits, se deben utilizar dos contadores en cascada para obtener el funcionamiento deseado.

Entre los dispositivos TTL de alta velocidad hemos encontrado el siguiente:

- 74F269 (ver hoja de datos en apéndice III)

Hemos elegido este por ser el mas standard y utilizado mundialmente. Nos provee las características necesarias para el proyecto, y es de alta disponibilidad, bajo precio, y existe mucha documentación sobre su uso. Consideramos que siempre que se puedan utilizar componentes masivos y siempre que cumplan con las especificaciones, sería un buen punto a tener en cuenta para tomar la decisión sobre su uso.

5. Buffers bidireccionales 8-bit

Son conocidos también como *transceptores*, y son básicamente 2 separadores de tres estados. Estos separadores de tres estados hacen que en funcionamiento normal, cada bit de salida tenga exactamente el mismo valor que en su correspondiente entrada. Sin embargo, cuando se los pone en el tercer estado *(tri-state)*, su salida pasa a ser de alta impedancia, tal como si no existiese conexión alguna. De esta forma se permite a algún otro dispositivo *escribir* en ese bus. Los transceptores, o buffers bidireccionales, hacen uso de esta característica para escribir a uno u otro lado del bus. Es decir que pueden hacer que un bus de entrada se convierta en uno de salida y viceversa. Tiene como objetivo poder separar a dos sub-circuitos que tengan funciones tanto de lectura como de escritura, entonces mediante este dispositivo es posible que ambos se comuniquen, tomando previa decisión de quién es el que escribe y quién es el que lee.

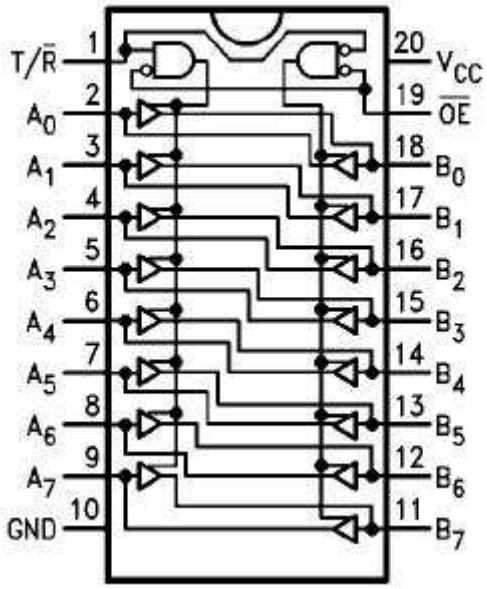


Fig 4.22 Buffer bidireccional

Al igual que en la elección del contador, hemos tomado el mismo criterio de utilizar un componente masivo. La línea 74F de Fairchild Semiconductors es la óptima para estos casos.

- 74F245 (ver hoja de datos en apéndice III)

6. Protectores USB

Estos protectores USB son supresores de transitorios de tensión que puedan ocurrir en la línea de comunicación. Estos ruidos pueden provenir de cualquier fuente, y pueden provocar daños a los equipos en ambos extremos del bus USB si son de magnitud y duración suficiente. Estos son protectores para USB 1.1, y nos son aptos para las altas velocidades del USB 2.0 dada su alta capacitancia de entrada.

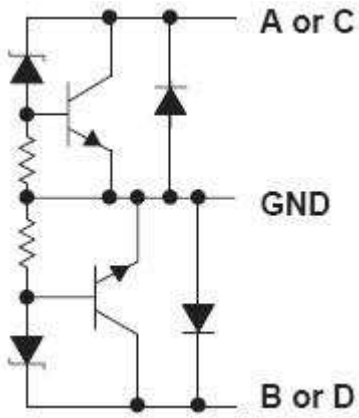


Fig 4.23 Ciurcuito de protección USB

Hemos buscado circuitos integrados que provean esta funcionalidad, y la única fabrica que provee componentes específicamente con este fin es Texas Instruments. Los modelos disponibles que hay son: SN75240, SN65240, y SN65220. Entre estos componentes, el único del cual teníamos disponibilidad y solamente en los Estados Unidos, es del SN65240. Este integrado difiere del SN65220 en que el primero tiene dos supresores mientras que el segundo tiene uno solo. Es evidente que esta característica no afecta en absoluto el diseño ni el funcionamiento del circuito.

- Texas SN65220/SN65240/SN75240 (ver hoja de datos en apéndice III)

7. Osciladores programables

Hemos evaluado también la utilización de un oscilador integrado programable en vez de cristales. Éstos tienen la ventaja de tener menor radiación, menor amplitud en componentes de mayor frecuencia, requieren menor cableado, tienen mayor estabilidad, y también evitar tener que comprar varios cristales para diferentes frecuencias cuando se pueden obtener a partir de un mismo integrado. Sin embargo los encapsulados en los que se consiguen estos integrados tienen un montaje mas complicado, y los cristales son de uso mucho más masivo y común que estos integrados. Hemos tenido en nuestro poder muestras de estos componentes, pero aún así tomamos la decisión de optar por un cristal, ya que tienen muy alta disponibilidad en cualquier mercado (incluso el local), son baratos, de fácil reemplazo, y proveen una forma segura y conocida de manejo. Además, se cuenta con extensa documentación sobre su uso. Creemos todas éstas razones suficientes para haber tomado la elección del cristal.

A continuación presentamos un ejemplo de oscilador programable.

- Linear Technology LTC6905 (ver hoja de datos en apéndice III)

Frecuencia máxima de trabajo

La frecuencia máxima de trabajo del osciloscopio puede calcularse a partir de las frecuencias máximas de los componentes que lo integran. Las compuertas

lógicas simples generalmente no son el problema, puesto que éstas, siendo de naturaleza TTL, tiene un ancho de banda muy superior al del resto de los componentes.

Por eso, para calcular la frecuencia máxima de trabajo del osciloscopio estudiaremos la frecuencia máxima que soporta cada uno de sus componentes. Sin embargo, no todos los componentes cumplen un papel relevante en el cálculo de la frecuencia máxima de trabajo. Por ejemplo, la velocidad máxima de trabajo del PIC es independiente de la velocidad máxima de trabajo del osciloscopio puesto que en las capturas a alta velocidad el PIC no juega ningún papel, más que el de disparar la captura. Por lo tanto, el análisis de la máxima frecuencia de trabajo debe realizarse a partir de los componentes que están involucrados directamente en la capturas a a alta velocidad, a saber:

- Conversores AD
- Contadores
- Memorias
- Amplificadores de entrada

De las hojas de datos de dichos componentes podemos extraer sus frecuencias máximas de trabajo, las cuales son:

Componente	**Modelo**	**Frecuencia máxima**
Contador	74F269	100 Mhz
Memoria	CY7C-109B-25	40 MHz (ciclo lectura/escritura: 25 us max)
Conversor AD	TLC5540	40 MHz
Amplificador	MAX477	300 Mhz

Tabla 4.3 Frecuencia máxima de componentes

Como se puede observar en la tabla las dos limitantes son el conversor AD y la memoria, siendo ésta última la menos importante puesto que se puede reemplazar por el modelo CY7C-109B-15 (de igual pinout) que trabaja hasta 66 Mhz. El conversor AD, en cambio, no tiene un sustituto inmediato conocido.

Por consigue, de éste análisis se desprende que la frecuencia máxima de trabajo del osciloscopio es de 40 Mhz. Para lograr dicha frecuencia se deberá utilizar un cristal de 40 Mhz y será necesario contar con un PCB bien diseñado a los efectos de minimizar ruidos e interferencias, que son muy dañinos a dichas frecuencias.

_En nuestro caso particular (en el cual no dispusimos del tiempo y los medios necesarios para fabricar un PCB) la frecuencia máxima a la cual pudimos hacer trabajar la placa fue **8 Mhz** ya que, para frecuencias superiores (probamos con un cristal de 20MHz) el funcionamiento de la placa era erróneo o nulo. Atribuimos estos problemas a la falta de haber diseñado un circuito impreso apropiado, y llegamos a la conclusión de que para trabajar a tan altas frecuencias la distribución y el diseño físico de la placa cumple un papel

fundamental, cosa que no ocurre a bajas frecuencias donde solo es necesario que los componentes estén conectados correctamente._

Por más información referirse al Capítulo 8. Fabricación y puesta en marcha, donde se pueden ver fotos de la placa.

Consumo de potencia y alimentación

Para decidir si es necesario alimentar la placa con una fuente externa (o si basta con la potencia entregada por el puerto USB) basta con estudiar el consumo de los diferentes componentes activos que integran el osciloscopio, los cuales se presentan en la siguiente tabla (todos los valores están en mA).

Comp.	Típica	Máxima	Cantidad	Típica	Máxima
PIC18F4550	200	300	1	200	300
74F269	113	135	2	226	270
74F245	95	120	2	190	240
74F00	7	11	3	21	33
TLC5540	17	27	2	34	54
CY7C-109B	80	100	2	160	200
MAX477	100	120	2	200	240
			Total	**1031**	**1337**

Tabla 4.4 Consumo de potencia de componentes activos

Dado que un puerto USB es capaz que suministrar un máximo de 500 mA de corriente por cada dispositivo conecto, resultó indispensable el uso de una fuente externa para alimentar la placa. Para ello utilizamos decidimos utilizar un regulador 7805 y optamos por una fuente de 9V DC pues es el voltaje mínimo necesario para el funcionamiento del 7805 (y por lo tanto el que disipa menos calor) y además permitiría la posibilidad de ser reemplazado por una batería de 9V, en caso de ser apropiado y/o necesario.

Referencias

Hojas de datos

Ver Apéndice III: Hojas de datos

Arquitecturas

- Procesador ETRAX de Axis
  - http://developer.axis.com/products/etrax100lx/index.html
- Referencia de diseño del ETRAX
  - http://developer.axis.com/doc/hardware/etrax100lx/des_ref.html
- RTAI - distribución de linux en tiempo real
  - http://rtai.org
- RTAI en ETRAX

    - http://www.linuxdevices.com/articles/AT6416763926.html
- Silicon Laboratories
    - http://www.silabs.com
- PIC18F4550
    - http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1335&dDocName=en010300

Conversores analogico-digitales

- Resolución
    - http://focus.ti.com/lit/an/sbaa051/sbaa051.pdf
- Conversores SAR
    - http://focus.ti.com/lit/ml/slyb115/slyb115.pdf
    - http://pdfserv.maxim-ic.com/en/an/AN1080.pdf
- Conversores Delta-sigma
    - http://focus.ti.com/lit/ml/slyb115/slyb115.pdf
    - http://pdfserv.maxim-ic.com/en/an/AN1080.pdf
- Conversores Pipeline
    - http://pdfserv.maxim-ic.com/en/an/AN383.pdf
    - http://focus.ti.com/lit/ml/slyb115/slyb115.pdf
- Conversores Flash
    - http://pdfserv.maxim-ic.com/en/an/AN810.pdf
- Conversores Integrados
    - http://www.maxim-ic.com/appnotes.cfm/appnote_number/1041

Memorias

- http://www.virtual.unal.edu.co/cursos/ingenieria/2000477/lecciones/100201.htm
- http://ortihuela.galeon.com/ram.htm
- SRAM
    - http://en.wikipedia.org/wiki/Static_random_access_memory
- DRAM
    - http://en.wikipedia.org/wiki/Dynamic_random_access_memory

Osciladores programables

- http://www.engineerlive.com/asiapacific-engineer/20051201/instrumentation--electronics/1.3.683.688/14613/silicon-oscillators-can-offer-advantages-over-crystals.thtml
- http://www.maxim-ic.com.cn/pdfserv/en/an/AN232.pdf
- http://www.jlab.org/~segal/junk/LTC6905.pdf

Figuras

- Fig. 4.1 - http://developer.axis.com/products/devboard/index.html
- Fig. 4.4 - http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=2124&param=en022613&page=wwwFullSpeedUSB

- Fig. 4.5 - hoja de datos del PIC18F4550 (ver apéndice III)
- Fig. 4.7 - http://www.create.ucsb.edu/~dano/CUI/
- Fig. 4.8 - manual de uso del Easy-ICD2
- Fig. 4.9 - http://focus.ti.com/lit/ml/slyb115/slyb115.pdf
- Fig. 4.10 - http://focus.ti.com/lit/ml/slyb115/slyb115.pdf
- Fig. 4.11 - http://focus.ti.com/lit/ml/slyb115/slyb115.pdf
- Fig. 4.12 - http://www-s.ti.com/sc/ds/tlc5540.pdf
- Fig. 4.13 - http://pdfserv.maxim-ic.com/en/an/AN810.pdf
- Fig. 4.14 - http://pdfserv.maxim-ic.com/en/an/AN810.pdf
- Fig. 4.16 - http://www.virtual.unal.edu.co/cursos/ingenieria/2000477/lecciones/100201.htm
- Fig. 4.17 - http://www.virtual.unal.edu.co/cursos/ingenieria/2000477/lecciones/100201.htm
- Fig. 4.18 - http://www.virtual.unal.edu.co/cursos/ingenieria/2000477/lecciones/100201.htm
- Fig. 4.19 - http://www.virtual.unal.edu.co/cursos/ingenieria/2000477/lecciones/100201.htm

Capítulo 5. Firmware

Contenido de este capítulo

Herramientas de trabajo

El firmware es el programa que corre internamente en el PIC y sirve para controlar el osciloscopio. Fue escrito enteramente en C utilizando el MPLAB C18 de Microchip, un compilador de C provisto por el mismo fabricante del PIC que soporta el estándar de C ANSI '89 y que ya viene pensado para trabajar de forma conjunta con el MPLAB IDE (que es la herramienta utilizada para programar y depurar el PIC). Otra de las características del MPLAB C18 es la posibilidad de generar binarios optimizados para microcontroladores de la familia PIC18F (por ejemplo, nuestro PIC18F4550) utilizando instrucciones extendidas de dicha arquitectura.

El MPLAB C18 está disponible para bajar gratuitamente de la página de Microchip (ver link en referencias). Sin embargo, la versión gratuita (llamada

versión estudiantil) tiene un duración de 60 días. A partir de esos 60 días, el programa seguirá funcionando pero sin las optimizaciones antes mencionadas, por lo cual el compilador generará binarios que seguirán funcionando pero ocuparán más espacio (al no estar optimizados) y no utilizarán las instrucciones extendidas del PIC18F4550, teniendo que resolver más funcionalidad por software y en consecuencia haciéndolo más ineficiente.

Clase de dispositivo

El estándar USB contempla varias clases de dispositivos para funcionalidades encontradas comúnmente en los dispositivos. Por ejemplo, existe un clase para las cámaras digitales, otra para los escáners, otra para las impresoras, etc. Las clases de dispositivos fueron inventadas para mejorar la interoperabilidad de los dispositivos. Así, cualquier sistema operativo que tenga un driver para trabajar con cámaras digitales puede leer fotos de la cámara digital que esté diseñada para cumplir las especificaciones de dicha clase de dispositivos. Por más información, ver el capítulo sobre USB.

En particular, para nuestro osciloscopio optamos por usar la clase de dispositivo CDC (Communication Device Class) que básicamente emula una conexión serie sobre el puerto USB. La razón por la cual optamos esta clase fue que el mecanismo de una conexión serie nos pareció un enfoque simple y efectivo para intercambiar simultáneamente información de control y datos. Además, al no haber ninguna clase prevista para un osciloscopio USB, una comunicación serie es el método más directo de implementar un driver propio puesto que solo basta con enviar y recibir cadenas de caracteres. En conclusión, escogimos la clase CDC por su *sencillez y flexibilidad*.

Firmware CDC

La comunicación USB se realiza mediante la ayuda del firmware CDC, un framework que brinda Microchip para poder establecer una comunicación (a través del puerto USB) de forma simplificada. El firmware CDC encapsula varias funciones ocultando toda la complejidad necesaria para la comunicación USB de forma de proveer una **comunicación serie tradicional** entre el PIC y el PC. Existen diferentes firmwares previstos para otras funcionalidades, como por ejemplo el de almacenamiento masivo (Mass Storage) para hacer un lector de tarjetas o el de dispositivos de interacción humana (Human Interface Device o HID) pensado para hacer un mouse o similar. Inprovee también un firmware más abierto para realizar una comunicación más avanzada.

Algunas de las características del firmware CDC son las siguientes:

- **Tasa de transferencia máxima de 80 kbytes/s**
- El librerías compiladas ocupan un tamaño relativamente chico (4 Kb)
- Resuelve toda la comunicación en software (no requiere de ningún hardware extra especial)
- El flujo de datos es manejado enteramente por el protocolo USB (no es necesario usar XON/XOFF ni control de flujo por hardware)

Limitaciones del firmware CDC

Un requisito de los programas escritos en C para el PIC es que no pueden terminar nunca puesto que si terminan el PIC automáticamente se resetea. Por lo tanto, todos los programas deben correr en un loop infinito. El nuestro no es una excepción y efectivamente así lo hace. Por lo tanto, existen dos grandes rutinas principales:

1. una rutina de inicialización (`OscInit()`)
2. una rutina que se ejecuta repetidamente en un loop infinito (`OscLoop()`)

La rutina `OscInit()` se encarga de inicializar los puertos y las interrupciones del PIC para el funcionamiento correcto del osciloscopio y se llama desde la rutina `InitializeSystem()`, quien a su vez también inicializa el firmware CDC para la comunicación USB.

Inmediatamente después de inicializado el sistema, se entra en un loop infinito que corre dos rutinas indefinidamente:

1. la rutina `UsbTasks()` que mantiene viva la conexión USB y envía/reciba los datos que haya pendientes
2. la rutina `OscLoop()` que contiene toda la lógica principal del osciloscopio

Dado que el firmware de nuestro osciloscopio corre sobre el firmware CDC fue necesario hacer un par de modificaciones mínimas al mismo para contemplar el funcionamiento paralelo de ambos (firmware CDC y firmware del osciloscopio). Para lograr ello solo bastó modificar el archivo principal (`main.c`) y colocar allí las llamadas a las funciones antes mencionadas (`OscInit()` y `OscLoop()`) en el lugar apropiado.

Captura de datos

Modos de captura

Debido a la naturaleza de los diversos integrados que componen el osciloscopio fue necesario incorporar en el firmware tres modos de funcionamiento diferentes para poder cubrir todo el rango de frecuencias en la etapa de adquisición.

En el siguiente diagrama puede observarse un bosquejo del funcionamiento de cada uno de estos modos, y a continuación se explican detalladamente estos modos de funcionamiento junto con las limitantes para cada caso.

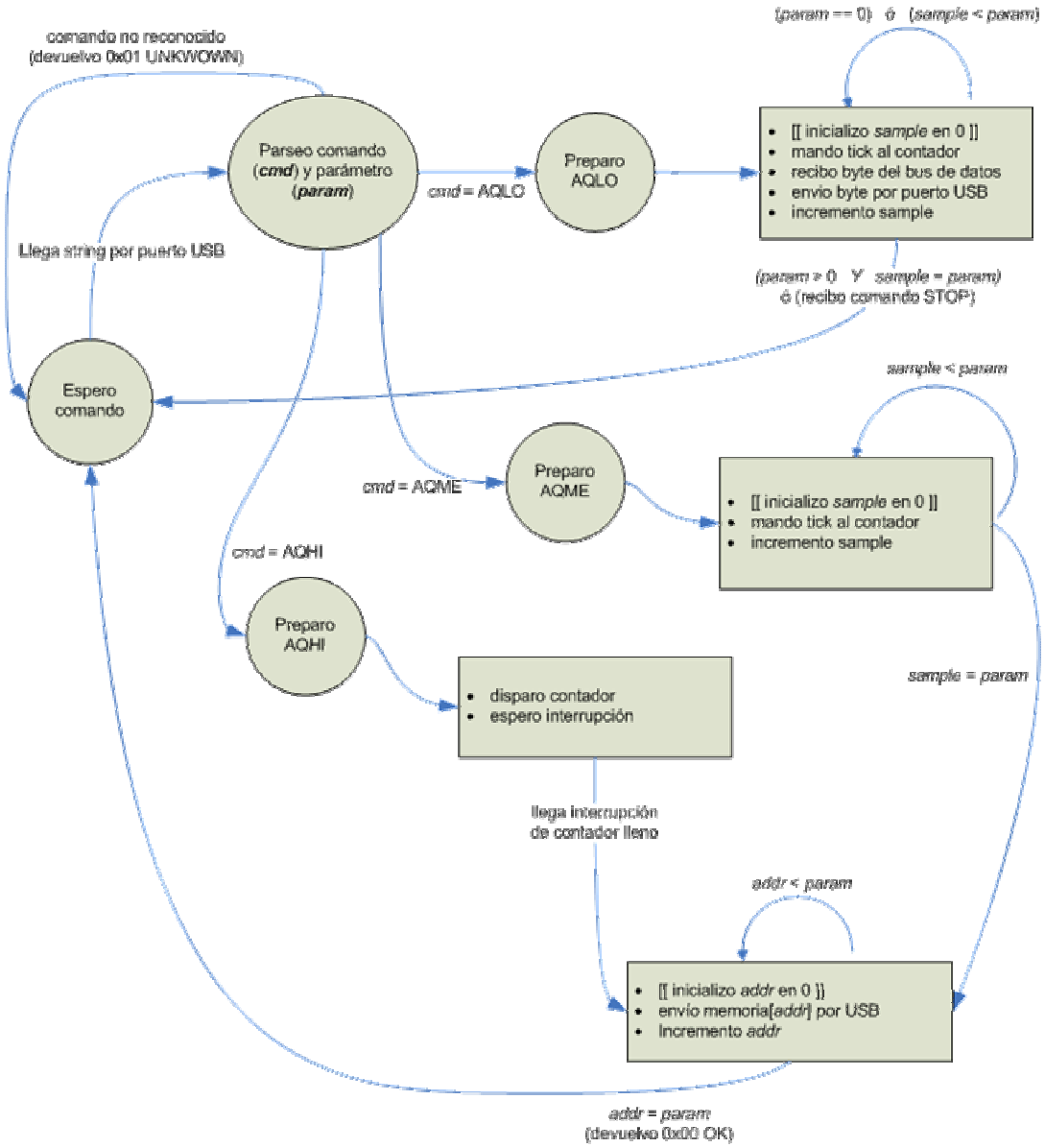


Fig 5.1 Diagrama de funcionamiento del firmware (modos de captura)

Captura a alta velocidad (comando AQHI)

La captura a alta velocidad se realiza utilizando los contadores para controlar las memorias, por lo cual es posible alcanzar velocidades de hasta 40 Mhz, que es la velocidad máxima de funcionamiento del conversor analógico-digital.

En este modo tenemos dos limitantes:

- por arriba, el factor limitante es la velocidad máxima de trabajo del

conversor conversor AD, puesto que los contadores y las memorias pueden trabajar a velocidades aún mayores.

- por abajo, la limitante es el tamaño de la memoria, puesto que el proceso de adquisición se ejecuta siempre a la velocidad del oscilador (en nuestro caso 8 Mhz) por lo cual la frecuencia mínima a muestrear (si

tomamos como requisito capturar al menos 4 ciclos de la señal) sería: f = fs / 65536 muestras / 3 ciclos, lo cual en nuestro caso (fs = 8Mhz) da cerca de un 1 Khz.

Sin embargo, en este modo de captura no se tiene ningún control sobre las memorias durante el proceso de captura: el PIC simplemente dispara los contadores y queda esperando a ser interrumpido por los contadores una vez que finaliza el proceso de captura y las memorias están llenas.

En este modo de captura la señal se muestrea en "ventanas", por lo cual solo sirve para señales periódicas.

Estos son los pasos seguidos por el PIC para ejecutar una captura a alta velocidad:

1. es presetean los contadores a cero
2. se libera el bus de datos
3. se setean los contadores para que cuenten hacia adelante (UPDN=1)
4. se selecciona el clock rápido en el bloque de control de memoria (CKSEL=0)
5. se habilita escritura del ADC en el bus (ADCOE=0)
6. se habilita escritura de memoria (WR=0)
7. se activan contadores (CKEN=0)
8. se espera a ser interrumpido por el segundo contador lleno
9. se transfieren los datos almacenados en la memoria por el puerto USB

Interrupción por contador lleno

Como se mencionó anteriormente, en el modo de captura a alta velocidad el PIC dispara el los contadores y estos se encargan de direccionar la memoria mientras los conversores AD colocan los valores digitalizados en los pines de datos de la memoria.

Debido a que el PIC, una vez que dispara los contadores, pierde el control sobre ellos se precisa un mecanismo para poder detener el proceso de captura. Para ello se ha conectado la pata TC del contador alto al PIC, de forma de que interrumpa al PIC cuando el contador se llene. La pata TC del contador, como puede observarse en la hoja de datos, vale cero únicamente cuando la salida del contador es 11111111, por lo cual se ha configurado el PIC para ser interrumpido por nivel en el pin donde se ha conectado el TC del contador. No cualquier pin del PIC puede ser utilizado para interrumpirlo. En particular, solo los pines RBx son capaces de brindar esa funcionalidad, por lo cual hubo que utilizar uno de éstos.

Al ser interrumpido, el PIC procede inmediatamente a deshabilita los contadores para evitar que se sobreescriban la mayor cantidad de valores (puesto que los contadores se resetean y la memoria se empieza a escribir a partir de la posición 0).

Otra forma de atacar este problema podría haber sido conectar la pata TC directamente a la pin que habilita los contadores de manera que al activarse TC automáticamente se deshabiliten los contadores. Esta solución efectivamente impediría que ningún valor de la memoria se sobreescribiese. Sin embargo, dado la gran cantidad de valores los pocos que resultan sobreescritos no afectan en absoluto para fines prácticos.

Sin embargo, éste mecanismo de detención por hardware resulta de mayor utilidad si se desea implementar un trigger por hardware, ya que en este caso las primeras muestras son muy relevantes.

Captura a media velocidad (comando AQME)

La captura a media velocidad se realiza controlando la escritura a memoria desde el PIC. En este modo también se capturan ventanas de la señal, para luego transferir los datos por el puerto USB, en una etapa posterior.

En este caso tenemos una limitante superior que es la velocidad de procesamiento del PIC. Este valor puede calcular exactamente partiendo de las instrucciones que son ejecutadas entre dos capturas consecutivas del PIC, y utilizando la cartilla de instrucciones del PIC donde viene especificada la duración de cada una. También sabemos que el PIC trabaja siempre a 12 MIPS, por lo cual es un calculo tedioso pero no presenta ninguna complicación. En nuestro caso, el código está escrito en C por lo cual habría que estudiar el código assembler generado por el compilar C18. De todas formas, encontramos de forma empírica que dicha limitante rondaba en los 6 Khz (tomando como requisito la captura de 4 ciclos de reloj).

Supongamos que se solicita una captura de media velocidad de N muestras. Los pasos seguidos por el PIC para realizar dicha captura son los siguientes:

1. acq_sample = N
2. se presetean los contadores a cero
3. se libera el bus de datos
4. se selecciona el clock lento en el bloque de control de memoria (CKSEL=1)
5. se habilita la escritura del ADC en el bus (ADCOE=0)
6. se habilita escritura de memoria (WR=0)
7. se ejecuta un tick en el contador bajo (CKLO = 0, CKLO = 1, CKLO = 0)
8. acq_sample = acq_sample - 1
9. si acq_sample > 0 entonces se va al paso 7, sino se sigue de largo
10. se transfieren los datos almacenados en la memoria (hasta la posición N) por el puerto USB

NOTA: Debido a que el conversor AD trabaja a una frecuencia *mínima* de 5 Khz

Captura a baja velocidad (comando AQLO)

Si bien la captura a media velocidad no tiene una frecuencia mínima definida, es de especial interés tener un mecanismo de captura en "tiempo real" para señales de muy baja frecuencia, y es por ello que existe el modo de captura a baja velocidad.

A diferencia de los 2 modos anteriores, la captura a baja velocidad captura y transfiere los datos directamente por el puerto USB, sin pasar por la memoria. Esta captura en tiempo real (es decir, sin usar ventanas) permite ver una captura continua de señales de baja frecuencia.

La limitante en este caso es la velocidad de transferencia por el puerto USB junto con la velocidad de procesamiento del PIC para llevar a cabo todas las tareas. El resultado empírico nos da una frecuencia máximo de trabajo de 4 Hz.

La captura a baja velocidad permite dibujar la señal en pantalla en el momento que se está capturando lo cual brinda una funcionalidad análoga a las provistas por los osciloscopios tradicionales.

Dado su naturaleza, este modo de captura es invocado de forma diferente que el resto. En este caso la captura se "inicia" ejecutando el comando AQLO y continua indefinidamente (transfiriendo datos por el puerto USB) hasta que es detenido con el comando STOP.

Este es un bosquejo de los pasos a seguir para realizar esta captura.

1. se presetean los contadores a cero
2. se libera el bus de datos
3. se setea el PIC para que lea del BUS de datos
4. se habilita la escritura del ADC en el bus de datos (ADCOE=0)
5. se lee el bus de datos y se almacena su contenido en un buffer
6. se transfiere el valor del buffer por el puerto USB
7. se espera una cantidad de tiempo determinada por el parámetro pasado al comando AQLO
8. se vuelve al paso 5, a menos que se haya recibido un comando STOP

Modos de captura según la frecuencia de trabajo

A continuación se presenta en una tabla un resumen de todos los modos de captura junto con sus respectivas frecuencias de trabajo, como así también las limitantes para cada caso.

Modo	F. mín	Limitante por abajo	F.MAX	Limitante por arriba
AQHI	1 Khz	tamaño de la memoria	fs	frecuencia del oscilador
AQME	-	-	6 Khz	velocidad de procesamiento del PIC
AQLO	-	-	4 Hz	transferencia por USB y procesamiento

Tabla 5.1 Modos de captura - frecuencias límite

La selección del modo captura depende de la frecuencia que se desea observar, y por consiguiente es responsabilidad de la aplicación que controla el osciloscopio solicitar el modo adecuado de captura para la frecuencia deseada.

Divisor horizontal (HDIV)

El funcionamiento de los modos de captura mencionados anteriormente puede ser afectado ligeramente utilizando el divisor horizontal. Este divisor es configurado a través del comando HDIV (ver Protocolo de comunicación) y es el que permite cubrir todo el rango de frecuencia.

El efecto del parámetro HDIV es el de retrasar el tiempo entre dos muestras. En todos los casos, HDIV=0 indica que no habrá ningun retraso y por lo tanto es la captura más rápida a ese modo de trabajo. Este se aplica de forma diferente según el modo de captura, a saber:

- en la captura a alta velocidad se utiliza para leer la memoria: la memoria es leída salteando HDIV posiciones entre cada lectura. Por ejemplo, para HDIV=5 se leerán (y transferirán) las muestras grabadas en las direcciones: 0, 5, 10, 15, etc.
- en la captura a media velocidad el parámetro HDIV es utilizado también par a leer la memoria, de análoga a la captura de alta velocidad
- en la captura a baja velocidad el parámetro HDIV es utilizado para generar una demora arbitraria entre dos transferencias consecutivas de datos, lo cual resulta en un muestreo a más baja velocidad.

Otros parámetros de captura

Además del parámetro HDIV existen otros parámetros que afectan las rutinas de adquisición de datos. Ellos son:

- CHAN - selecciona el canal a muestrear
- DUAL - habilita el muestreo de ambos canales
- CHOP - selecciona como serán muestreados ambos canales (solo para modo DUAL)
- VDV1 - selecciona la escala de voltaje a utilizar en la etapa de entrada del canal 1
- VDV2 - selecciona la escala de voltaje a utilizar en la etapa de entrada del canal 2

Driver

Vendor ID y Product ID

El estándar USB exige que todos los dispositivos, durante la etapa de negociación, se identifiquen con un *Vendor ID* y un *Product ID* (en adelante, VID y PID). Dicho par de valores sirve para conocer el fabricante del dispositivo (VID) y el modelo particular del producto que se ha conectado (PID). Por lo

tanto, modelos diferentes de un mismo producto generalmente tienen PIDs diferentes.

La utilidad principal de estos valores no es solamente la de identificar el dispositivo, sino la de encontrar y cargar los drivers apropiados para el mismo. Por consiguiente, cada driver que viene con Windows (o que bajamos de Internet) viene programado (al igual que el dispositivo) con uno o más PID y VID para los cuales sirve dicho driver. Esta es la forma que tiene Windows (o el sistema operativo en cuestión) de saber si el driver seleccionado es correcto.

En el caso de que el driver ya venga con el sistema operativo, el par VID/PID bastará para identificar el driver que es necesario cargar y por lo tanto cuando se conecta un dispositivo con VID/PID conocido el sistema lo detecta automáticamente e inmediatamente queda listo para usar. Sin embargo, en el caso de que el VID/PID no sea reconocido, el sistema operativo solicitará al usuario que suministre los drivers. Un ejemplo de esto es la conocida pantalla de Nuevo Hardware Detectado de Windows.

Driver para Windows 98/2000/XP

Como ya mencionamos anteriormente, el osciloscopio utiliza el estándar CDC para comunicarse con la PC por el puerto USB. Los sistemas operativos Windows, desde la versión 98SE, ya traen incluido un driver para control dispositivos CDC. Dicho driver es el archivo `usbser.sys` que, en Windows XP por ejemplo, se encuentra en `c:\windows\system32\drivers`.

Todos los dispositivos que utilizan dicho driver (por ejemplo, adaptadores Serie-USB) automáticamente agregan un puerto COM virtual al sistema, mientras están conectados, de forma que cualquier aplicación puede trabajar directamente con el puerto virtual como si se tratase de cualquier puerto serie real.

Nuestro osciloscopio, al utilizar el estándar CDC, hace uso de dicho driver y, por consiguiente, no requiere de un driver extra a partir de Windows 98SE en adelante.

Sin embargo, Windows a priori no sabe que driver debe asignar a nuestro osciloscopio porque no reconoce el par VID/PID con el cual éste se identifica. Por lo tanto, es necesario indicarle a Windows que driver debe utilizar para nuestro VID/PID. Esto se hace a través de un archivo .inf donde se especifica, entre otras cosas:

- PID/VID del dispositivo
- el driver a utilizar (en este caso `usbser.sys`)
- nombre con el que aparecerá el osciloscopio en la lista de dispositivos conectados (en este caso, dentro de Puertos de Comunicaciones)

Archivo INF

A continuación se muestra el archivo inf del osciloscopio:

```
; Archivo INF para la instalacion del osciloscopio USB utilizando
; el driver USB CDC ACM
;
; Proyecto de fin de carrera 2005 - Universidad ORT
;
; Pablo Hoffman - Martin Szmulewicz

[Version]
Signature="$Windows NT$"
Class=Ports
ClassGuid={4D36E978-E325-11CE-BFC1-08002BE10318}
Provider=%MCHP%
LayoutFile=layout.inf
DriverVer=08/17/2001,5.1.2600.0

[Manufacturer]
%MFGNAME%=DeviceList

[DestinationDirs]
DefaultDestDir=12

[SourceDisksFiles]

[SourceDisksNames]

[DeviceList]
%DESCRIPTION%=DriverInstall, USB\VID_05F9&PID_FFFF

;------------------------------------------------------------------------
---------
;  Windows 2000/XP Sections
;------------------------------------------------------------------------
---------

[DriverInstall.nt]
CopyFiles=DriverCopyFiles
AddReg=DriverInstall.nt.AddReg

[DriverCopyFiles]
usbser.sys,,,0x20

[DriverInstall.nt.AddReg]
HKR,,DevLoader,,*ntkern
HKR,,NTMPDriver,,usbser.sys
HKR,,EnumPropPages32,,"MsPorts.dll,SerialPortPropPageProvider"

[DriverInstall.nt.Services]
AddService=usbser, 0x00000002, DriverService

[DriverService]
DisplayName=%SERVICE%
ServiceType=1
StartType=3
ErrorControl=1
ServiceBinary=%12%\usbser.sys

;------------------------------------------------------------------------
---------
;  String Definitions
;------------------------------------------------------------------------
---------
```

```
[Strings]
MCHP="pablohoffman.com"
MFGNAME="pablohoffman.com"
DESCRIPTION="Osciloscopio USB"
SERVICE="USB RS-232 Emulation Driver"
```

En la línea:

```
%DESCRIPTION%=DriverInstall, USB\VID_05F9&PID_FFFF
```

Se puede observar la definición del VID y PID, en este caso **VID=05F9** y **PID=FFFF**. Dichos valores deben coincidir con lo valores del firmware (ubicados en oscusb/autofiles/usbdsc.c), para Windows reconozca el dispositivo y automáticamente le asigne el driver `usbser.sys`.

Driver para Linux

Linux, al igual que Windows, trae incluidos muchos drivers, entre ellos uno para dispositivos CDC. Por lo tanto, lo que hicimos fue utilizar un VID y PID conocidos para que linux lo reconociera y entonces asignara el driver apropiado al dispositivo. Esta es la razón por la cual elegimos los números VID=05F9 y PID=FFFF, no fue una decisión arbitraria.

Trabajando con esos valores, Linux automáticamente detecta el dispositivo (una vez conectado) y le asigna un archivo (`/dev/ttyACM0`) que puede ser utilizado para comunicación en forma serie como cualquier puerto serial estándar (`/dev/ttySxx`). Es importante tener en cuenta que el kernel de Linux debe estar compilado para soportar dispositivos CDC (cualquier kernel actual ya viene preparado).

De esta forma queda resuelta la comunicación USB desde Linux sin tener proporcionar un driver especial o siquiera un archivo INF (como en el caso de Windows).

Comunicación USB

Toda la comunicación a través del puerto USB se realiza utilizando un juego de funciones que provee el firmware CDC. Ellas son las siguientes:

Funciones de transferencia USB

Función	Descripción
putrsUSBUSART	Escribe un string terminado-nulo de la memoria de programa al puerto USB
putsUSBUSART	Escribe un string terminado-nulo de la memoria de datos al puerto USB
mUSBUSARTTxRom	Escribe un string de largo conocido de la memoria de programa al puerto USB
mUSBUSARTTxRam	Escribe un string de largo conocido de la memoria

	de datos al puerto USB
mUSBUSARTIsTxTrfReady	Devuelve TRUE si el driver está pronto para escribir datos en el puerto USB
getsUSBUSART	Lee un string del puerto USB
mCDCGetRxLength	Devuelve el largo del último string leído del puerto USB

Tabla 5.2 Funciones de transferencia USB

Para que la comunicación USB funcione correctamente y el dispositivo no pierda la conexión con el computador, el firmware CDC impone algunas restricciones que es necesario seguir estrictamente. Dichas restricciones se discuten a continuación:

Chequear si está disponible para enviar

Antes de enviar cualquier string por el puerto USB es necesario chequear que se encuentra listo para enviar más datos, utilizando la función mUSBUSARTIsTxTrfReady.

No utilizar código bloqueante

Dado que las funciones del puerto USB se ejecutan en la rutina `UsbTasks()` (cada vez que se entra al loop principal) se debe evitar a toda costa utilizar funciones bloqueantes que dependan del estado del puerto USB puesto que éste no se actualiza hasta que se vuelve a correr el loop, y por lo tanto resultaría en una situación de deadlock.

A modo de ejemplo, el siguiente código es incorrecto:

```
while(!mUSBUSARTIsTxTrfReady());
putrsUSBUSART("Hola mundo");
```

La forma correcta sería la siguiente:

```
if (mUSBUSARTIsTxTrfReady()) {
    putrsUSBUSART("Hola mundo");
}
```

Como todo el código se corre continuamente dentro de un loop infinito, en principio, el resultado de ambos códigos es el mismo, pero en caso de que el código sea más complicado puede requerir un análisis más profundo.

Cuidados a tener en cuenta al enviar datos

Las funciones de envío de datos (putsUSBUSART, putrsUSBUSART, mUSBUSARTTxRam y mUSBUSARTTxRom) no son funciones bloqueantes ni tampoco envían los datos inmediatamente, sino que habilitan determinados registros e inicializan una máquina de estados para la transferencia que se ejecutará *posteriormente*, una vez que se vuelva a iniciar el loop principal (dentro de la rutina `CDCTxService()` que se llama desde `UsbTasks()`).

Debido a esto, llamadas consecutivas a dichas funciones no funcionan pues la última llamada siempre sobreescribe a la anterior. La forma correcta de enviar varios strings consecutivamente es usando una máquina de estado o implementando rutinas que almacenen los datos en un buffer intermediarios. A continuación se muestran los dos casos:

Llamadas consecutivas (incorrecto)

```
if(mUSBUSARTIsTxTrfReady()) {
    putrsUSBUSART("Hola mundo");
    putrsUSBUSART("Hola de nuevo");
}
```

Usando máquina de estado (correcto)

```
byte state = 0;
if(state == 0) {
    if(mUSBUSARTIsTxTrfReady()) {
        putrsUSBUSART("Hola mundo");
        state++;
    }
} else if(state == 1) {
    if(mUSBUSARTIsTxTrfReady()) {
        putrsUSBUSART("Hola de nuevo");
        state++;
    }
}
```

Usando buffer intermedio (correcto)

```
if (mUSBUSARTIsTxTrfReady()) {
    putsUSBUSART(io_buffer);
    SendToBuffer("Hola mundo");
    SendToBuffer("Hola de nuevo");
}
```

En este caso la función `SendToBuffer()` concatenaría los strings en un buffer intermedio para ser enviando en la siguiente transferencia (a correrse en el próximo ciclo). Dicha función debe ser implementada por separado.

Este último fue la solución que adoptamos para manejar los envíos por el puerto USB, y dicha funcionalidad se encuentra implementada en `oscusb/osc/usbtx.c`.

Variables y manejo de memoria

Si bien el compilador C18 de Microchip se ajusta bastante bien al estándar ANSI '87, vale recalcar que tiene algunas particularidades que es necesario tener en cuenta a la hora de escribir código, como ser el manejo de las variables en memoria.

En concreto, las variables pueden estar almacenadas tanto en memoria RAM como en memoria ROM. El compilador C18 soporta un par de calificadores (no

contemplados en el estándar de C) `rom` y `ram` que permiten especificar donde se guardará la variable. Por ejemplo:

```
rom int version;
ram char[10] command;
```

El problema es que las funciones de manejo de string no permiten trabajar con variables ubicadas en distintas memorias. Por lo tanto, la función `strcmp()` no permite comparar una cadena almacenada en la ROM con otra cadena almacenada en la RAM.

Todas las variables definidas sin el calificador `rom/ram` quedan por defecto almacenadas en la RAM, mientras que las constantes como `"Hola mundo"` quedan almacenadas en la ROM.

Tomemos como ejemplo el siguiente código:

```
#include <string.h>

char cadena[10] = "hola";

bool comparo() {

    if (strcmp(cadena, "hola")) {
        return TRUE;
    } else {
        return FALSE;
    }
}
```

En principio, la función `comparo()` debería retornar TRUE. Sin embargo, debido a los problemas mencionados del almacenamiento en la RAM/ROM, retorna FALSE puesto que `strcmp()` no sabe comparar datos de la RAM (`cadena`) con datos de la ROM (`"hola"`). Por lo tanto es necesario reescribir el código de la siguiente manera para su correcto funcionamiento:

```
#include <string.h>

char cadena[10] = "hola";

bool comparo() {
    char hola[5] = "hola";

    if (strcmp(cadena, hola)) {
        return TRUE;
    } else {
        return FALSE;
    }
}
```

Y por lo tanto así es como está implementado (en osc.c) la comparación de los datos entrantes del puerto USB para detectar el comando que fue solicitado.

Trabajar de esta forma resulta bastante tedioso y perdimos bastante tiempo para darnos cuenta de donde estaba el problema. Incluso fue necesario el uso

del debugger por hardware. Además, nos parece que estos problemas deberían estar resueltos a nivel del compilador.

Estructura del código fuente

Esta es una descripción de los archivos y directorios del código fuente del firmware. Las entradas que terminan en / son directorios, mientras que el resto son archivos.

Archivo/directorio	Función que implementa
oscusb/autofiles/	Archivos descriptores del dispositivo USB
oscusb/inf/	Archivo INF que describe el dispositivo para windows (driver Windows 2000/XP)
oscusb/system/	Librerías del firmware CDC
oscusb/_output/	Archivo de salida del compilador y el enlazador
oscusb/main.c	Archivo principal desde donde arranca a correr el firmware
oscusb/oscusb.mcp	Archivo Project del MPLAB IDE
oscusb/oscusb.mcw	Archivo Workspace del MPLAB IDE
oscusb/oscusb.lkr	Archivo de comandos para el enlazador
oscusb/io_cfg.h	Definición básica de patas del PIC para el funcionamiento de la comunicación USB (definidas por el firmware CDC)
oscusb/osc/interrupt.c	Control y manejo de interrupciones
oscusb/osc/osc.c	Control y manejo del osciloscopio
oscusb/osc/usbtx.c	Control de transferencias por el puerto USB
oscusb/osc/osc_io.h	Definición de patas de E/S y macro del PIC para el osciloscopio

Tabla 5.3 Archivos del firmware

Configuration bits

Los **configuration bits** sirven para configurar el modo de funcionamiento del PIC (por ejemplo, la frecuencia del oscilador) y deben asignarse durante la programación. La configuration bits son seteados por el MPLAB durante la programación y se pueden asignar de 2 formas:

1. a través de la lista de configuration bits del MPLAB (en Configure -> Configuration bits)
2. a través de macros en el propio código, utilizando la declaración #pragma config

A continuación se muestra la pantalla de selección de los Configuration bits del MPLAB (opción 1).

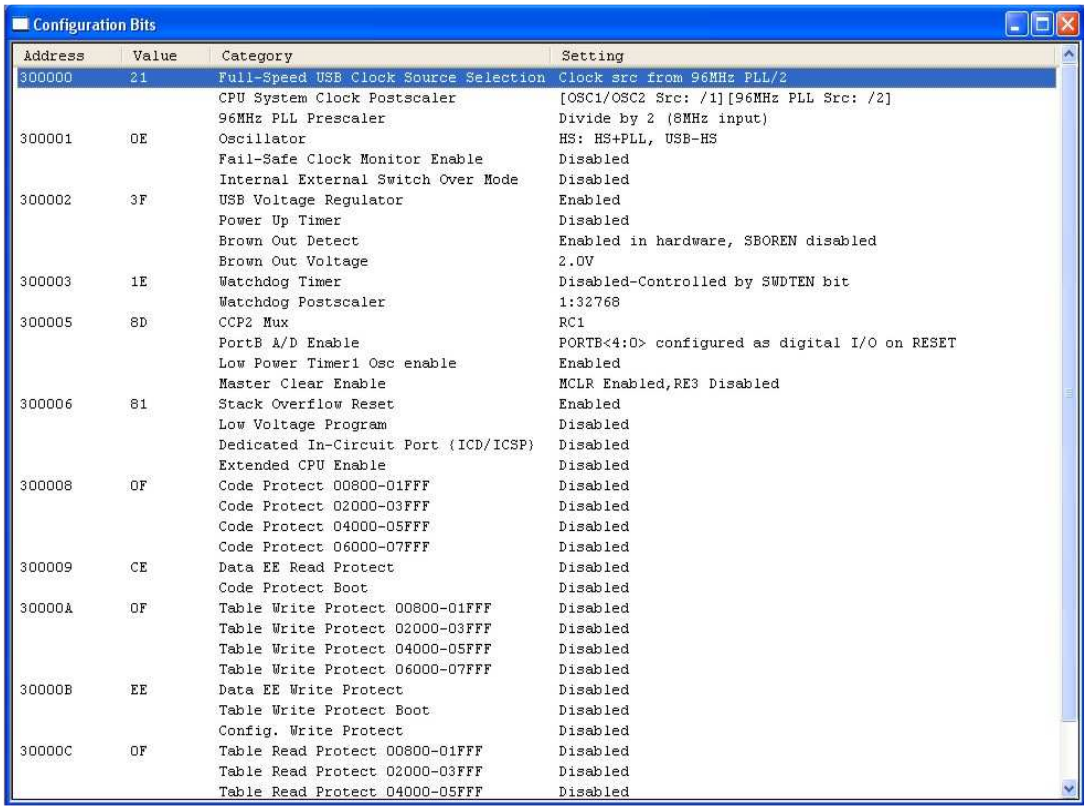


Fig. 5.2 Configuration bits del PIC 18F4550

Nosotros preferimos asignar los Configuration bits a través de definiciones en el propio código, para no depender de la configuración del entorno de trabajo del MPLAB. Dichas definiciones se encuentran en el archivo `oscusb/osc/confbits.h`.

Referencias

- Especificación de la clase de dispositivo CDC
  - http://www.usb.org/developers/devclass_docs/usbcdc11.pdf
- Microchip C18
  - http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en010014
- Informacion sobre configuration bits y #pragma config
  - http://ww1.microchip.com/downloads/en/devicedoc/39622f.pdf
- Como migrar aplicaciones de UART RS-232 a USB con un mínimo esfuerzo (application note de Microchip)
  - http://ww1.microchip.com/downloads/en/AppNotes/00956b.pdf

Capítulo 6. Software

Contenido de este capítulo

Selección de la plataforma

Concretamente, las herramientas a seleccionar son: a) el lenguaje de programación y b) el toolkit gráfico. El toolkit gráfico es la API utilizada para el manejo, control y creación de una aplicación con interfaz gráfica.

Para la decisión del lenguaje a utilizar para la elaboración del software gráfico que se instalará en el PC para el control del osciloscopio desarrollado se han tenido en cuenta los siguientes requisitos:

Requisitos tenidos en cuenta

1. Debe tener interfaz gráfica
2. Debe ser multi-plataforma (Windows, Linux, Mac OS X)
3. Debe ser de rápido desarrollo
4. Debe poder comunicarse con puertos virtuales de Windows (para hablar con el driver RS232-USB de CDC de Microchip) de forma sencilla

5. Debe ser un lenguaje gratuito que no requiera de licencias para compilar o distribuir el código compilado

Lenguaje de programación

El lenguaje que seleccionas para programar la aplicación fue python, ya que es un lenguaje robusto y extremadamente portable (multi-plataforma, *requisito 2*) puesto que existen interpretadores para todas las plataformas, dos de las cuales (Linux y Mac OS X) ya lo traen incluido dentro del propio sistema operativo. Además, es un lenguaje interpretado lo cual elimina los tediosos ciclos de edición-compilación-enlazado-ejecución y permite un desarrollo rápido y ágil (*requisito 3*). Por eso es comúnmente utilizado en el prototipado de aplicaciones, y lo hace ideal para nuestro caso.

Además python cuenta con una extensión llamada pySerial que permite el acceso transparente a puertos series (virtuales o reales) de Windows y, a la vez, a puertos series en linux, lo cual nos permite desentendernos del problema de la comunicación (*requisito 4*).

Otras aplicaciones como Java, si bien son portables, exigen más tiempo de diseño e implementación (pues el lenguaje es más estricto) y requieren tener instalada una máquina virtual Java (JRE o JDK) en la máquina del cliente donde se vaya a correr la aplicación. Al hacerlo en python se puede evitar esa restricción compilando la aplicación para que corra nativamente en cada arquitectura y sistema operativo.

La plataforma .Net de Microsoft no es portable para sistemas operativos fuera de windows y además su licencia tiene algunas restricciones respecto al código generado con ella, por lo cual fue descartada inmediatamente.

Toolkit gráfico

En cuanto al toolkit gráfico a decisión no fue tan sencilla.

Las bindings más populares disponibles para python son:

- tkInter - para trabajar con el toolkit **tk**
- GTK+ - utilizando los bindings PyGTK
- qt - utilizando los bindings PyQt
- wxWidgets - utilizando los bindings wxPython

tkinter

Tk es el toolkit nativo de python para trabajar con interfaces gráficas, pero la razón de esto es que era el mejorcito que existía en el momento que python fue diseñado, y de eso hace mucho tiempo. Tk tiene dos grandes desventajas: requiere mucho código para implementar cosas simples y ademas no es muy presentable. Por estas razones, tkinter fue descartado inmediatamente.

GTK+

A través de los bindings PyGTK se puede utilizar el popular toolkit GTK+ de la Free Software Foundation. Sin embargo lo descartamos por las siguientes razones:

- GTK no es muy lindo estéticamente en Windows
- GTK no tiene implementación para Mac OS X
- GTK no tiene un buen entorno rápido de desarrollo (RAD) para el diseño de ventanas

qt

Qt es muy portable (Windows, Linux, Mac) y estéticamente muy prolijo. Sin embargo, lo descartamos puesto que su licencia es ligeramente restrictiva para aplicaciones Windows (no así para Linux/Mac) y además la versión disponible actualmente (8 Feb 2006) de PyQt (los bindings de qt para python) no soporta la última versión de qt (4.x) sino que solo soporta hasta la 3.x, y además no soporta python 2.4 (hasta 2.3).

wxWidgets

wxPython usa el toolkit wxWidgets que es bastante prolijo y muy portable (Windows, Linux y Mac). Además tiene las siguientes ventajas:

- cuenta un muy buen diseñador rápido de ventanas (llamado Boa Constructor) que es gratuito y libre
- tiene una excelente documentación
- tiene una gran popularidad entre los toolkit GUI para python, lo cual significa que tiene un gran soporte comunitario (foros, etc)

Por estas razones es que decidimos adoptar wxPython como toolkit gráfico para la aplicación del osciloscopio.

Otras librerías

Otras librerías a utilizar son las siguientes:

pySerial

La librería pySerial permite comunicarse con los puertos de la PC de forma transparente, sencilla y (lo que es más importante) independiente de la plataforma.

pyWin32

La librería pyWin32 son bindings para poder acceder a la API de win32 desde python. La razón por la cual la usamos es que el pySerial la necesita para acceder a los puertos serie en Windows.

NumPy

La librería NumPy provee funciones para calculo de transformada discreta de Fourier (FFT) la cual es usada para mostrar el espectro de la señal en el software.

Entorno de desarrollo

El entorno de desarrollo utilizado para crear la aplicación gráfica fue el Boa Constructor. Entre sus características se encuentra el desarrollo rápido (drag and drop) de ventanas y sus componentes (botones, barras de estado, etc), como así también la posibilidad de depurar el programa gráfico utilizando breakpoints y las herramientas típicas de depuración.

Un dato curioso es que, además de ser un entorno de desarrollo para wxPython, el Boa Constructor también está escrito en python utilizando wxPython.

Compilador

En particular (para la plataforma Windows) se utilizó el compilador py2exe para compilar la aplicación a un archivo ejecutable (EXE) sin dependencia de ninguna otra librería lo cual le brinda al osciloscopio una gran portabilidad y facilita su distribución.

En Linux y Mac OS X no es necesario el uso de un compilador puesto que dichos sistemas operativos ya vienen con python instalados y puede correr automáticamente cualquier programa escrito en ese lenguaje.

Resumen

Este es un resumen del lenguaje, librerías y herramientas utilizadas para desarrollar el software:

Lenguaje	python
Librerías	pySerial NumPy pyWin32 (solo windows)
Entorno de desarrollo	Boa Constructor
Compilador	py2exe (solo Windows)

Tabla 6.1 Herramientas de desarrollo del software

Código fuente

Estructura

La estructura del código fuente es bastante simple y consta de los siguientes archivos:

- `oscapp.py` - este es el código que dispara toda la aplicación. Es el que llama a la ventana (frame) para controlar el programa
- `oscframe.py` - aquí está el código que define el comportamiento y la apariencia de la ventana del software. Aquí está todo el código de la aplicación gráfica.
- `oscctrl.py` - aquí está el código de comunicación con el osciloscopio. Este vendría a ser el driver
- `setup.py` - este archivo no es código propiamente dicho sino que son las directivas utilizadas para compilar el programa en Windows usando py2exe

Funcionamiento

Una programa gráfico en wxPython consta de una *Aplicación* y varios *Frames*, que pertenecen a ella. Estos frames son justamente las diferentes ventanas de la aplicación. Como en nuestro caso el software tiene una sola ventana, el mismo tiene un solo *Frame*.

Al disparar la aplicación (`oscusb.py`), ésta abre el frame por defecto (`oscframe.py`) que es la ventana que se ve cuando se ejecuta el programa.

El comunicación con el osciloscopio se realiza a través del driver que se encuentra implementado en el archivo `oscctrl.py` como una clase de python y es utilizado desde oscframe.py para enviar comando y recibir datos.

Driver

El driver (`oscctrl.py`) permite la comunicación con el osciloscopio y está implementado como una clase en python, la cual brinda las siguientes funciones, entre otras:

- `connect()` - conecta al osciloscopio devolviendo True o False si pudo conectarse
- `getVersion()` - devuelve la versión de firmware del osciloscopio al que se encuentra conectado
- `getPort()` - devuelve el puerto serie a través del cual se encuentra conectado
- `getData()` - devuelve los datos recibidos por el último comando (por ejemplo, un comando de captura)
- =send(comando, parámetros) - envía un comando junto con una lista opcional de parámetros y devuelve el código (numérico) de la respuesta
- =acquire(count, hdiv, vdiv, htrig) - ejecuta un comando de captura (ACQU) configurando el osciloscopio previamente con los parámetros especificados (hdiv, vdiv, htrig)
- `disconnect()` - desconecta el osciloscopio

Aplicación

Como ya se mencionó, la aplicación cuenta con una única ventana (llamada *Frame*) que está implementado en el archivo `oscframe.py`. Allí se encuentra todo el código de la aplicación, a saber:

1. creación y posicionamiento de los controles de la ventana (botones, barras de estado, etc)
2. comunicación con el osciloscopio a través del driver (`oscctrl.py`)
3. rutinas para graficar en pantalla la señal recibida por el osciloscopio en forma de muestras (secuencias de números)

1. Creación de la ventana

La creación de la ventana se realiza (al igual que en cualquier aplicación gráfica) creando un *Frame* y asignádole botones a dicho *Frame* en coordenadas específicas. Las coordenadas pueden darse en pixels o en proporciones, lo cual permite que la ventana puede mantener su aspecto al ser maximizada o cambiar su tamaño. A los diferentes controles (como por ejemplo, los botones) se les define un comportamiento a través de eventos que se disparan cuando se realiza una acción sobre los mismos (por ejemplo, pulsar un botón). Dichos eventos llaman a una función especificada, cuando son disparados.

2. Comunicación con el osciloscopio

La comunicación con el osciloscopio se realiza a través de la clase provista por el driver (`oscctrl.py`) al dispararse ciertos eventos.

Actualmente la aplicación sigue el siguiente mecanismo para conectarse con el osciloscopio:

1. Intenta conectarse al primer puerto serie disponible en el PC (COM1 en caso de Windows, /dev/ttyACM0 en caso de linux)
2. Si logra conectarse envía un comando VERS (ver Capítulo 7 - protocolo de comunicación), de lo contrario prueba con el siguiente puerto disponible que encuentra
3. Si obtiene respuesta al comando VERS, entonces considera que la comunicación con osciloscopio se ha realizado exitosamente y despliega la versión de firmware del mismo (la cual fue obtenida en la respuesta del comando VERS).

Este este mecanismo de barrido fue por un razón de comodidad ya que, debido a que osciloscopio genera dinámicamente un puerto serie virtual cada vez que se conecta, éste no siempre era el mismo, aún cuando se conectase el osciloscopio en el mismo puerto USB. Por lo tanto, el barrido nos ahorró el tiempo de estar buscando y configurando el puerto virtual del osciloscopio.

Sin embargo, a pesar de contar con dichas ventajas, reconocemos que el mecanismo de barrido debe ser sustituido en el producto final puesto que, al enviar información a todos los puertos, puede interferir con el funcionamiento de algún dispositivo conectado al PC por puerto serie.

Para este problema existen dos soluciones, una trivial y una prolija:

1. la solución es trivial consiste simplemente en colocar (en la aplicación) un selector del puerto serie a utilizar para el osciloscopio.
2. la solución prolija consiste en desarrollar un driver USB personalizado para el osciloscopio que no involucre puertos serie virtuales de por medio.

El único otro momento donde el programa se comunica con el osciloscopio es al pulsar el botón **Capturar** en el cual envía un comando de captura, precedido por los comandos de configuración de los parámetros de captura (HDIV, DUAL, etc).

3. Graficación de datos en pantalla

La graficación de los datos recibidos del osciloscopio se realiza a través del objeto wxDC del API wxWidgets lo cual, a diferencia de usar el API de win32 (por ejemplo) lo hace independiente de la plataforma.

A continuación se muestra la rutina de graficación:

```
size = dc.GetSize()
dc.SetPen(wx.GREEN_PEN)
top = min(len(data), size.x)
lastx, lasty = 0, 0
vdiv = vdivs[self.vdivch.GetSelection()]
for i in range(0, top):
    val = (data[i] - 128) / 255  / vdiv
    x = i
    y = int(size.y/2*val)
    if x > 0:
        dc.DrawLine(lastx, lasty, x, y)
    lastx = x
    lasty = y
dc.SetPen(wx.NullPen)
```

Uso de hilos (threads)

En particular, resulta de especial importancia comentar sobre el uso de hilos para la comunicación con el osciloscopio, ya que de lo contrario la aplicación resulta lenta y poco responsiva. Esto es porque, al activar la captura el programa está continuamente enviando comando ACQU al osciloscopio y recibiendo sus datos. Entonces, si esto se hace en primer plano (que es la forma más fácil e inmediata de hacerla) el programa queda colgado esperando los datos del osciloscopio hasta que estos llegan y los gráfica. El problema es que, como el tiempo de transferir los datos del osciloscopio al PC es bastante mayor comparado con el resto de los tiempos, el programa está continuamente esperando datos del osciloscopio y mientras esto sucede la ventana no responde lo cual resulta en una aplicación terriblemente molesta de usar.

Para solucionar esto es que existe la tecnología de los hilos en el cual, al presionar el botón de captura la aplicación dispara un sub-programa (llamado

hilo) de captura que trabaja en forma paralela al programa principal. Y una vez que le llegan una tanda de datos del osciloscopio (recordar que éste es el proceso problemático pues es lento) lo notifica a la aplicación principal transfiriéndole los datos y ésta lo único que tiene que hacer es graficarlos que es un proceso prácticamente instantáneo y por lo tanto la ventana no queda congelada.

Todo esto se realiza utilizando la clase Thread de python y extendiéndola, en la clase `AcquireThread` que se ve muestra continuación:

```
class AcquireThread(Thread):
    def __init__(self, notify_window):
        Thread.__init__(self)
        self._notify_window = notify_window

    def run(self):
        win = self._notify_window
        osc = win.osc
        hd = win.hdivch.GetSelection()
        size = 512

        if win.osc.acquire(count=size):
            chardata = win.osc.getData()
            data = []
            for c in chardata:
                data.append(ord(c))

            wx.PostEvent(self._notify_window, AcquireEvent(data))
        else:
            wx.PostEvent(self._notify_window, AcquireEvent(None))
```

De particular interés son las lineas `wx.PostEvent` donde se dispara el evento una vez terminada la captura de datos. En caso de haber algún error (segunda línea de `wx.PostEvent`) el hilo envía None en lugar de los datos lo cual notifica al programa principal que hubo un error al capturar los datos.

Para la notificación y comunicación de datos entre el hilo de captura y la aplicación se utiliza la clase wx.pyEvent de wxPython, extendiéndola para que permita enviar los datos dentro de si misma. El código de dicha clase se presenta a continuación:

```
class AcquireEvent(wx.PyEvent):
    def __init__(self, data):
        wx.PyEvent.__init__(self)
        self.SetEventType(EVT_RESULT_ID)
        self.data = data
```

Notar que aquí la única extensión que se le hizo a la clase base wx.PyEvent fue la de agregarle el campo data (usado para transportar los datos del osciloscopio).

Descargar código fuente

Descargar el software

La última versión del software se encuentra disponible para bajar en la siguiente página:

- http://pablohoffman.com/twiki/pub/Oscusb/OscusbSoftware

Referencias

- Lenguaje python
  - http://www.python.org
- Librería pySerial
  - http://pyserial.sourceforge.net
- Librería NumPy
  - http://numeric.scipy.org/
- Librería PyQt
  - http://www.pyqt.org
- Librería PyGTK
  - http://pygtk.org
- Toolkit qt
  - http://www.trolltech.com/download/qt/x11.html
- Toolkit GTK+
  - http://www.gtk.org/
- GNU
  - http://www.gnu.org/
- Boa Constructor (entorno de desarrollo)
  - http://boa-constructor.sourceforge.net/
- Librería wxPython
  - http://www.wxpython.org
- Librería wxWidgets
  - http://www.wxwidgets.org
- Librería pywin32
  - http://sourceforge.net/projects/pywin32/
- Compilador py2exe
  - http://www.py2exe.org/
- Instrucciones para empaquetado multiplataforma
  - http://yergler.net/talks/desktopapps_uk/

Capítulo 7. Protocolo de comunicación

Contenido de este capítulo

Objetivos

Al diseñar el protocolo de comunicación, se tuvieron en cuenta los siguientes puntos:

- **comunicación lineal (comando, respuesta)** - para que pueda ser fácilmente adaptado a una conexión serie
- **comandos y respuestas ASCII** - el formato de los comandos y respuestas debe ser tal que pueda ser controlado y depurado desde una terminal de texto ASCII (como Hyperterminal o similar)

- **posibilidad de transferencia binaria** - para acelerar las transferencias entre la PC y el osciloscopio
- **comandos y respuestas simples** - puesto que la capacidad de procesamiento del PIC es reducida, el protocolo debe ser fácilmente parseable.

Para cumplir con el cuarto requisito se decidió que los comandos constaran de un largo fijo de 4 caracteres.

Introducción

El protocolo es bien simple y consta de un único tipo de interacción: *comando y respuesta*. Cada comando puede tener uno o ningún parámetro. Los comandos son enviados desde el PC al osciloscopio y son seguidos obligatoriamente por una respuesta (del osciloscopio al PC). No se puede enviar un nuevo comando hasta no haber recibido la respuesta del anterior, salvo por el comando STOP que cancela el comando en curso.

A continuación se presenta un diagrama de dicha interacción:

Diagrama de la interacción

```
                comando
| PC |  ---------------------->  | Osciloscopio |

               respuesta
| PC |  <----------------------  | Osciloscopio |
```

Comandos

Los comandos tienen la finalidad de enviar una solicitud al osciloscopio para que realice una acción y devuelva una respuesta, ya sea para confirmar el comando recibido, o para devolver el dato solicitado en el comando.

Formato de comandos

El formato de los comandos es el siguiente:

```
| COMANDO | espacio | PARAMETRO   |    \n    |
  4 bytes   1 byte    0..n bytes    1 byte

CMD_ID    = identificador del comando en ASCII (4 caracteres)
PARAMETRO = parámetro del comando (un número en formato ASCII).
            si el comando no tiene parámetros este valor es ignorado.

espacio     = un espacio (caracter ASCII 32)
\n          = fin de linea (caracter ASCII 10)
```

Comandos de captura

Los siguientes comandos sirven para solicitar una captura de datos al osciloscopio. Por más información sobre los distintos tipos de captura consulte el Capítulo 5 - Firmware.

AQHI

Ejecuta una captura de alta velocidad con los parámetros pre-fijados (a través de los comandos STxx) y devuelve los valores.

Si el formato binario esta habilitado (BINA 1) los datos se devuelven como bytes adyacentes en formato binario. El valor de cada byte puede ser de 0-255 que equivale justamente a los 8 bits de resolución del osciloscopio.

Si el formato binario está deshabilitado (BINA 0) los datos se devuelven como números en formato ASCII separados por un espacio. Por ejemplo: 34 123 243. Todos esos números están entre 0 y 255.

Los parámetros de la captura pueden ser ajustados a través de los comandos HDIV, VTRI, VDIV, DUAL y CHOP.

- Parámetro: cantidad de muestras a devolver.
  - Valores posibles: 1 - 65535
  - Valor por defecto: 65535

AQME

Similar a AQHI, pero la captura se ejecuta a media velocidad (las escrituras a memoria son controladas por el PIC).

- Parámetro: cantidad de muestras a devolver.
  - Valores posibles: 1 - 65535
  - Valor por defecto: 65535

AQLO

Similar a AQHI y AQME, pero la captura se realiza en tiempo real, transfiriendo las muestras a medida que se van capturando. Este comando soporta el valor especial 0 como parámetro lo cual significa seguir capturando sin detenerse, o hasta recibir un comando STOP, en lugar de transferir una cantidad pre-definida de valores.

- Parámetro: cantidad de muestras a devolver o selección del modo de captura
  - Valores posibles:
    - 1 - 65535 - cantidad muestras a devolver
    - 0 = captura continua (sigue capturando hasta recibir un comando STOP)
  - Valor por defecto: 0

Comandos de configuración

Los comandos de configuración permiten configurar diversos aspectos del funcionamiento de la captura de datos y deben ser enviados antes del comando de captura.

CHAN

Selecciona el canal a usar para capturar los datos. Solo tiene validez cuando el osciloscopio funciona en modo simple canal (DUAL=0).

- Valores posibles del parámetro:
  - 1 - canal 1 (valor por defecto)
  - 2 - canal 2

ADDR

Especifica la dirección a partir de la cual se leerá el contenido de la memoria en los comandos de captura, y también en el comando DUMP. Este comando está pensado para fines depurativos.

- Valores posibles del parámetro: 1 - 65535
  - Valores por defecto: 0

HDIV

Especifica la división horizontal a usar. Sirve para enlentecer el comando de captura y así obtener una serie de muestras más espaciadas en el tiempo.

- Valores posibles del parámetro: 0 - 65535
  - Valor por defecto: 0

VDV1

Selecciona la escala de voltaje a usar en el canal 1.

- Valores posibles del parámetro:
  - 0 - ±5V (valor por defecto)
  - 1 - ±10V
  - 2 - ±20V
  - 3 - ±40V

VDV2

Selecciona la escala de voltaje a usar en el canal 2.

- Valores posibles del parámetro:
  - 0 - ±5V (valor por defecto)
  - 1 - ±10V
  - 2 - ±20V
  - 3 - ±40V

BINA

Configura el modo binario de transferencia de los comandos de captura (AQHI/AQME/AQLO). Los formatos disponibles son: binario y ASCII. El formato binario es más eficiente en cuanto a velocidad. El formato ASCII es legible en una terminal de texto como Hyperterminal.

- Valores posibles del parámetro:
  - 0 - deshabilitar modo binario
  - 1 - habilitar modo binario (valor por defecto)

DUAL

Configura el modo dual de captura.

- Valores posibles del parámetro:
  - 0 - habilita captura de un único canal (valor por defecto)
  - 1 - habilita captura de ambos canales

CHOP

Configura la forma en que serán recibidas las muestras del osciloscopio. Si está habilitado, se recibirá una muestra por cada canal alternadamente. Por el contrario, si está deshabilitado, se recibirán primero todas las muestras del canal 1 y luego todas las muestras del canal 2.

Esta opción tiene validez únicamente cuando está habilitado el modo Dual.

- Valores posibles del parámetro:
  - 0 - deshabilita el modo chop (valor por defecto)
  - 1 - habilita el modo chop

Comandos de control

Los siguientes comandos sirven para controlar el estado del osciloscopio.

STOP

Detiene el comando de captura en curso. Pensado para utilizar principalmente con el modo continuo del comando AQLO.

- Parámetro: no tiene

RSET

Resetea el osciloscopio, volviendo todas los parámetros de configuración a su valor por defecto.

- Parámetro: no tiene

Comandos de diagnóstico

Los siguientes comandos sirven para monitorear el estado del osciloscopio y obtener información sobre el mismo.

PING

Devuelve OK si el osciloscopio está activo.

- Parámetro: no tiene

VERS

Devuelve la versión de firmware del osciloscopio, en formato ASCII 8-bit.

- Parámetro: no tiene

Comandos de depuración

Los siguientes comandos sirven para depurar el osciloscopio y están pensados para ser usados únicamente para testear el correcto funcionamiento del mismo. No tienen ninguna utilidad para la aplicación que interactuará con el usuario final.

WRLO

Escribe una señal cuadrada en la memoria a baja velocidad (controlada por el PIC). El parámetro pasado es la cantidad de muestras a escribir. Esta función existe únicamente para fines depurativos.

- Valores posibles del parámetro: 0 - 65535
    - Valor por defecto: 65535

WRHI

Escribe un mismo valor en todas las posiciones de la memoria a alta velocidad (controlado por el contador). El parámetro pasado es el valor a escribir. Esta función existe únicamente para findes depurativos.

- Valores posibles del parámetro: 0 - 255

DUMP

Vuelca el contenido de la memoria, para el canal seleccionado con el comando CHAN y a partir de la dirección especificada con el comando ADDR. El parámetro pasado es la cantidad de muestras a volcar.

- Valores posibles del parámetro: 1 - 65535

Respuestas

La respuesta es la reacción del osciloscopio a un comando. Todos los comandos devuelven una respuesta. Para enviar un nuevo comando se debe esperar a recibir la respuesta del último comando enviado, salvo por el comando STOP que cancela el comando actual.

Existen dos tipos de respuestas:

- respuestas con datos - son aquellas que devuelven valores
- respuestas sin datos - son aquellas que no devuelven valores

Formato de respuestas

El formato de las respuestas es el siguiente:

```
Respuestas con datos:

|  CODIGO  | espacio |   NOMBRE    | espacio |   LARGO   |   \n   |
DATOS    |   \n   |
| 4 bytes  |  1 byte |   n bytes   |  1 byte |  n bytes  | 1 byte |  n
bytes   | 1 byte |

Respuestas sin datos:

|  CODIGO  | espacio |    NOMBRE   |   \n   |
| 4 bytes  |  1 byte |   n bytes   | 1 byte |

CODIGO      = código de respuesta (número en formato ASCII)
NOMBRE      = un nombre descriptivo de la respuesta (en formato ASCII)
LARGO       = cantidad de bytes de la respuesta incluyendo el último
\n (en formato ASCII)
              solo para comandos que devuelven datos.
DATOS       = datos de la respuesta (en formato ASCII o binario según
corresponda)
              solo para comandos que devuelven datos.

espacio     = un espacio (caracter ASCII 32)
\n          = fin de linea (caracter ASCII 10)
```

Códigos de respuesta

Los códigos de error disponibles (en la versión de firmware 1.00) son los siguientes:

Código	Nombre	Significado
0	OK	comando aceptado
1	UNKNOWN	comando desconocido
2	OUT-OF-RANGE	valor fuera de rango
3	BUSY	el osciloscopio está ocupado ejecutando otro comando

Tabla 7.1 Códigos de respuesta

Ejemplos de sesión (comandos y respuestas)

```
PING
0 OK
VERS
0 OK 5
1.00
HDIV 128
0 OK
AQHI 5
0 OK 6
%!_P^
BINA 0
0 OK
AQHI 5
0 OK 18
24 123 203 129 56
CAPTURAR 343
1 UNKNOWN
```

Capítulo 8. Fabricación y puesta en marcha

Contenido de este capítulo

Fabricación de la placa

Alternativas de fabricación

Llegada la etapa de la construcción de la placa en donde montar los componentes, nos planteamos tres posibilidades para llevar a cabo dicha tarea:

- Protoboard
- Circuito impreso (PCB)
- Placa universal

Protoboard

El Protoboard, o tableta experimental, es una herramienta que nos permite interconectar elementos electrónicos, ya sean resistencias, condensadores, semiconductores, etc, sin la necesidad de soldar los componentes.

El protoboard esta lleno de orificios metalizados -con contactos de presión- en los cuales se insertan los componentes del circuito a ensamblar.

Se conocen en español como "placas de prototipos" y son esencialmente unas placas agujereadas con conexiones internas dispuestas en hileras, de modo que forman una matriz de taladros a los que podemos directamente "pinchar" componentes y formar el circuito deseado. Como el nombre indica, se trata de montar prototipos, de forma eventual, nunca permanente, por lo que probamos y volvemos a desmontar los componentes, quedando la protoboard lista para el próximo experimento.

Cada agujero de inserción está a una distancia normalizada de los demás, lo que quiere decir que un circuito integrado encajará perfectamente.

Tienen la ventaja de ser de rápida ejecución, sin necesidad de soldador ni herramientas, pero los circuitos que montemos deberán ser más bien sencillos, pues de otro modo se complica en exceso y las conexiones pueden dar lugar a fallos, porque la fiabilidad de las mismas decrece rápidamente según aumenta el número de éstas.

Fig 8.1 Protoboard

Esta opción si bien puede ser simple y rápida de llevar a cabo, consideramos que es poco prolija y puede traer complicaciones de falso contacto, interferencia, entre otros problemas.

Circuito impreso (PCB)

El circuito impreso esta constituido por una placa aislante, en una o en sus dos caras, de conductores planos metalizados cuyo objeto es asegurar las conexiones eléctricas entre el conjunto de los componentes electrónicos dispuestos en su superficie.

El término normalizado que designa a este componente es placa impresa, pero en uso común se emplea circuito impreso. Igualmente, en inglés el término Printed Circuit Board es de uso corriente, mientras que printed circuit se emplea prácticamente solo para referirse a la técnica de la fabricación de una placa impresa.

Existen distrintos tipos de circuitos impresos, de simple capa (llamado de una sola cara), de doble capa, multicapa, flexible, rígido, flexorígido, de agujeros metalizados, etc.

El circuito impreso puede ser la mejor opción si ya se conoce y se tiene probado el circuito, donde no se permiten cambios fácilmente, pero todas las conexiones son seguras, y mas aptas para trabajar a alta velocidad.

Sin embargo esta opción la descartamos ya que si bien tiene muchas ventajas, no nos permite hacer modificaciones fácilmente, y creemos que para una etapa de prueba no es la mejor solución.

Además, esta solución implica un costo mayor. Más aún, en los principios del diseño del PCB (que luego hemos postergado), hemos observado una alta complejidad en la distribución de las pistas, necesitándose una placa doble faz, con puentes y jumpers, pero aún así, no estamos seguros de que quepa todo el circuito en una placa doblefaz.

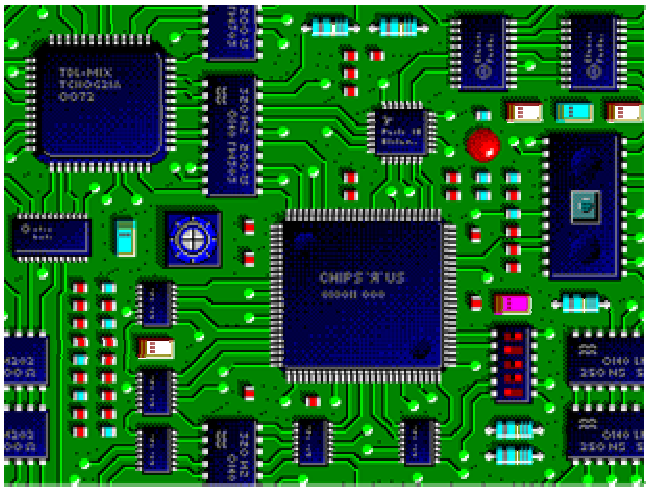


Fig 8.2 Circuito Impreso

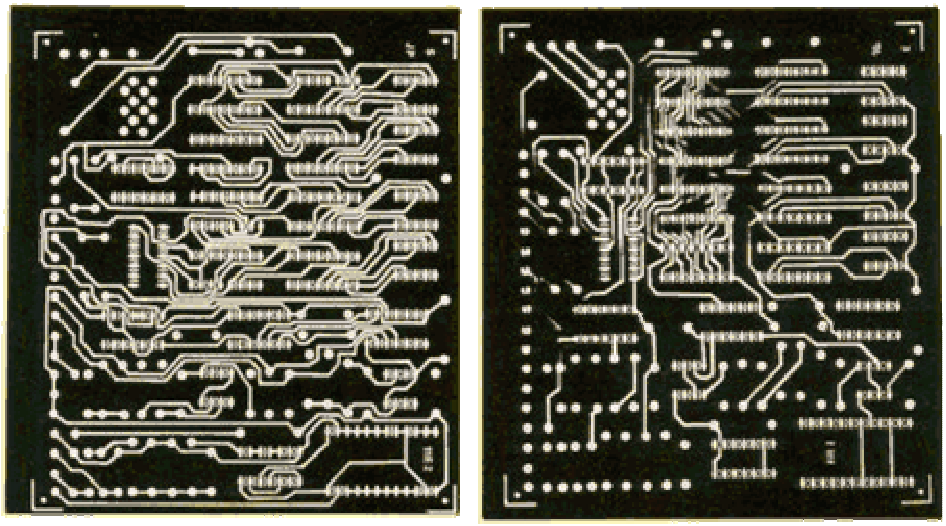


Fig 8.3 PCB

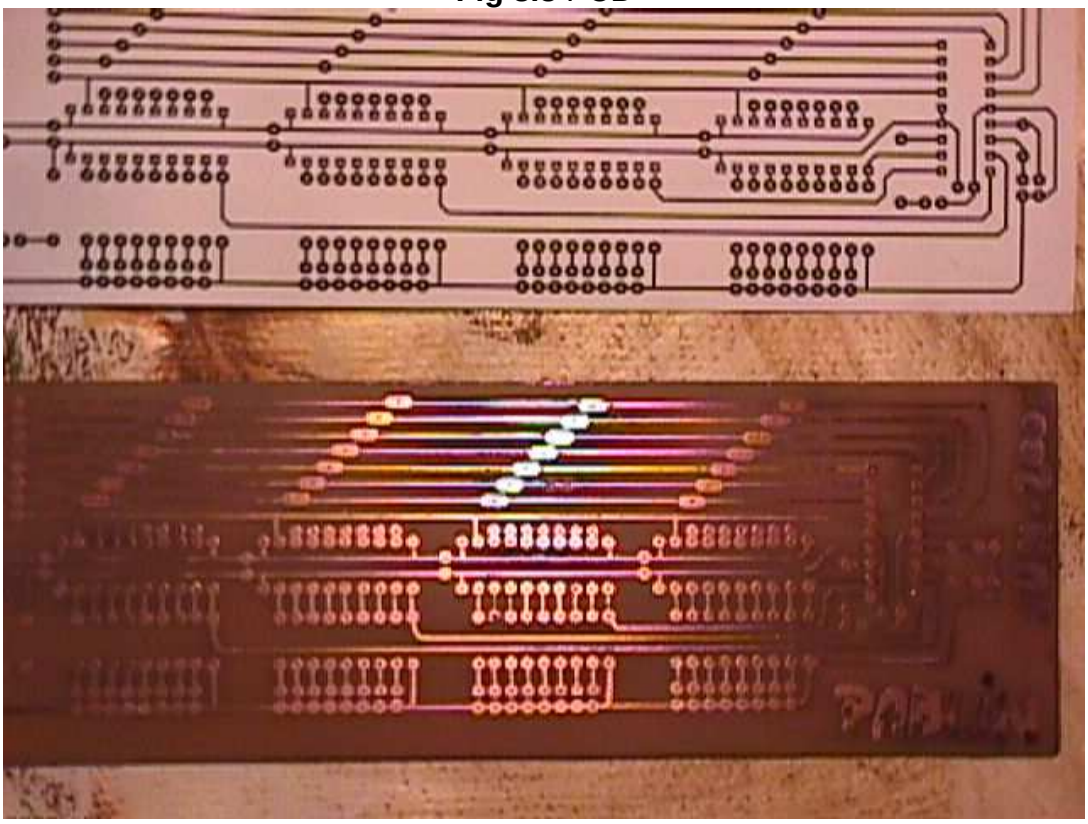


Fig 8.4 Placa Impresa

Tal como se puede observar en la imagen, un circuito impreso no es mas que una placa plástica (que puede ser de fenólico o pertinax), sobre la cual se dibujan "pistas" e "islas" de cobre las cuales formaran el trazado de dicho circuito, partiendo de un dibujo en papel, ya sea creado por una persona o por un ordenador.

Para empezar se debe decidir el material que se va a utilizar. Si se trata de un circuito donde hayan señales de radio o de muy alta frecuencia tendremos que usar una placa virgen de pertinax, que es un material poco alterable por la

humedad. De lo contrario, para la mayoría de las aplicaciones, una placa de fenólico es suficiente.

Cada trazo o línea se denomina pista, la cual puede ser vista como un cable que une dos o mas puntos del circuito. Cada círculo o cuadrado con un orificio central donde el terminal de un componente será insertado y soldado se denomina isla.

Cuando se compra una placa de circuito impreso virgen ésta se encuentra recubierta completamente con una lámina de cobre, por lo que, para formar las pistas e islas del circuito habrá que eliminar las partes de cobre sobrantes.

Además de pistas e islas sobre un circuito impreso se pueden escribir leyendas o hacer dibujos. Esto es útil, por ejemplo, para señalar cual terminal es el positivo, hacia dónde se inserta un determinado componente o incluso como marca de referencia del fabricante.

Para que las partes de cobre sobrantes sean eliminadas de la superficie de la placa se utiliza un ácido, el Percloruro de Hierro o Percloruro Férrico. Este ácido produce una rápida oxidación sobre metal haciéndolo desaparecer pero no produce efecto alguno sobre plástico. Utilizando un marcador de tinta permanente o plantillas Logotyp podemos dibujar sobre la cara de cobre virgen el circuito tal como queremos que quede y luego de pasarlo por el ácido obtendremos una placa de circuito impreso con las pistas que se pretendían.

Placa universal

El circuito impreso universal para prototipos, también conocido como _UPCB (Universal Printed Circuit Board)_, es un circuito impreso de uso general diseñado a partir de la estructura básica del protoboard, esta placa facilita el montaje de aplicaciones electrónicas sin requerir la etapa de diseño y fabricación de un circuito impreso especifico.

El circuito impreso universal para prototipos esta fabricado con importantes recursos tecnológicos que garantizan aplicaciones de mejor desempeño y presentación, entre las principales características técnicas se encuentran:

- Capa de blindaje en cobre para evitar interferencias.
- Pads recubiertos de estaño-plomo, evita oxidación y garantiza una soldadura de máxima calidad.
- Película de antisoder verde que recubre el cobre en las áreas donde no se debe soldar protege de la oxidación y de cortocircuitos.
- Hueco de 3mm para tornillo permite un eficaz anclaje al chasis o caja contenedora.
- Circuito impreso de fibra de vidrio provee máxima resistencia al impacto y a torsiones.

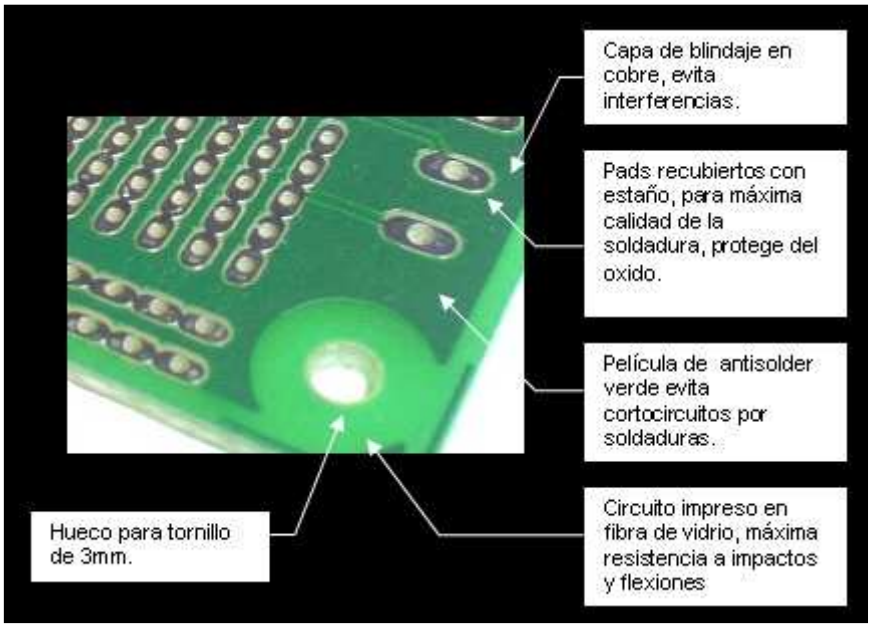


Fig 8.5 Placa Universal en detalle

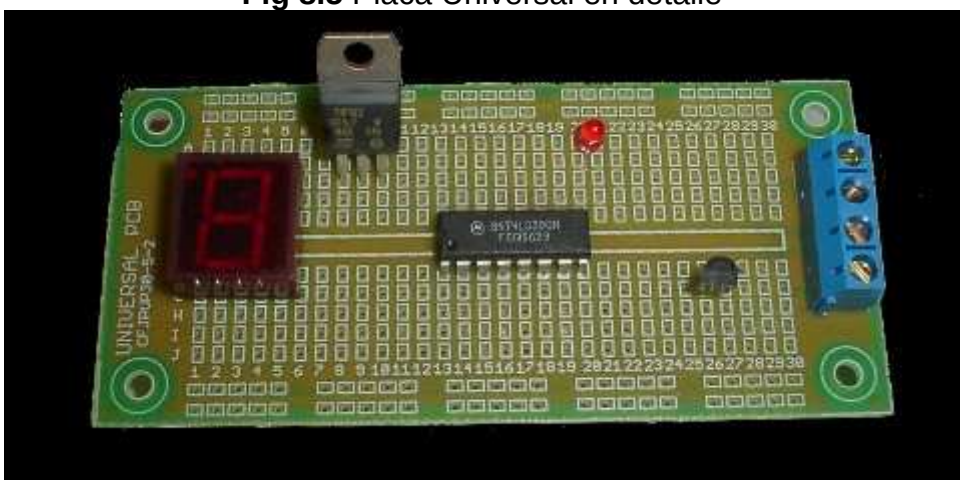


Fig 8.6 Placa Universal

La placa universal tiene las ventajas de un protoboard, ya que es versátil y permite cambios fácilmente, y las de un circuito impreso, con la facilidad de la colocación de los componentes y pistas de cobre para soldar. Esta placa es muy permisiva en cuanto a las conexiones, ya que al tratarse de cables, éstos pueden hacer cruces entre sí que no se podrían realizar en un circuito impreso.

También nos permitió realizar cambios en cuanto a conexiones cuando encontrábamos un funcionamiento erróneo, o por ejemplo cuando habíamos elegido una salida del microcontrolador que no permitía el tipo uso que se precisaba.

Proceso de fabricación

Habiendo ya decidido que la opción sería construir sobre una placa universal, a continuación detallamos nuestro proceso de construcción.

- Materiales
- Disposición de componentes

- Realización, prueba y experiencia

Materiales

Básicamente los materiales que se precisan en esta etapa son:

- UPCB (placa universal)
- Zócalos
- Cables y buses

Hemos utilizado las placas universales que se consiguen hoy en plaza, ya que son de buena calidad y es lo estándar para estos casos. Nos han dado muy buen resultado, con excepcion de alguna pista que se corta o se suelta por la presión de los cables al torcerlos, o al desoldar y volver a soldar en el mismo pad (isla).

Los zócalos los creímos necesarios ya que permiten soldar y dejar previsto el lugar de los mismos, sin que el mismo componente se encuentre en el lugar. Más aún, esto nos permite no tener que soldar el componente en si, sino que éste se coloca a presión en el zócalo. Además, el hecho de utilizar zócalos tiene como ventaja fundamental la facilidad de colocación y extracción del componente, como también su reemplazo.

Existe un tipo de zócalo diferente y especial que es el *ZIF* (Zero Insertion Force). Estos zócalos no requieren de fuerza para la inserción del componente, sino que se coloca dentro del zócalo y luego se presionan las patas del mismo moviendo una palanca que tiene en uno de los extremos.

Esto permite de forma simple y sin riesgos colocar o extraer el componente del zócalo. En nuestro caso, lo vimos sumamente útil para el microcontrolador, el cual es uno de los componentes más delicados y con frecuencia se saca y se vuelve a colocar para su programación.

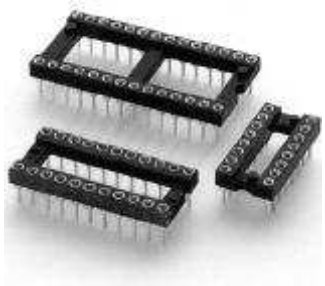


Fig 8.7 Zócalo

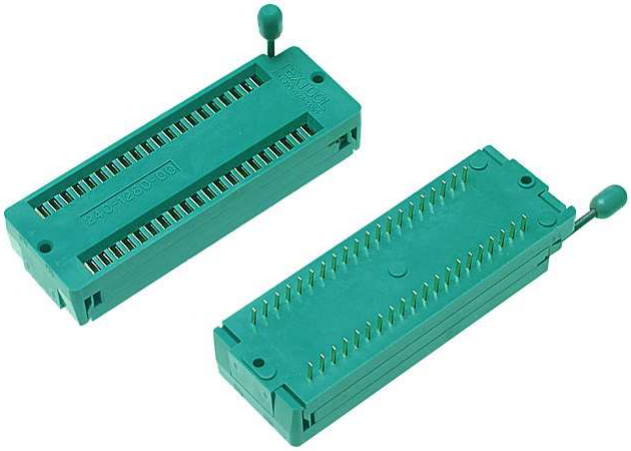


Fig 8.8 Zócalo ZIF

Para realizar las conexiones entre los componentes hemos utilizado cables obtenidos del UTP Categoría 5e. Este tipo de cable esta muy probado en la industria y es muy flexible en cuanto a su torsión y a su facilidad de soldado.

Los buses los hemos cableado con cables del tipo *flat*, los cuales son específicos para este tipo de conexiones.

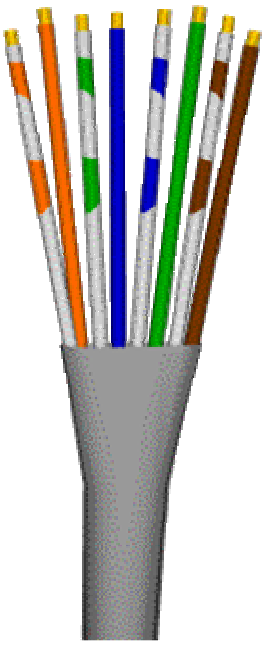


Fig 8.9 Cable UTP Cat5e

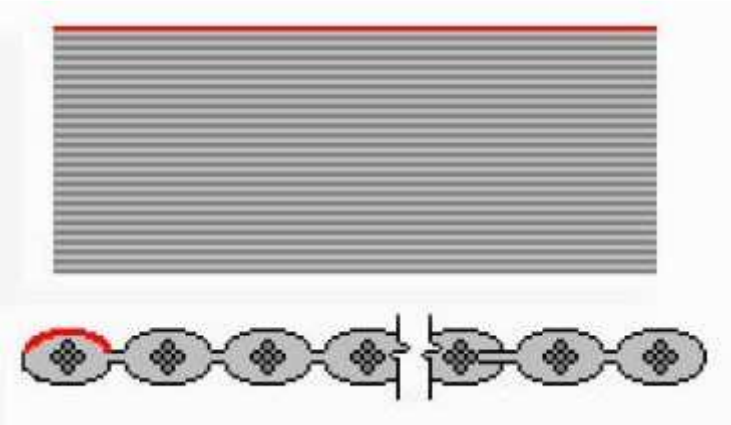


Fig 8.9 Cable Flat

Disposición de componentes

La distribución de los componentes en la placa tuvo que ser pensada basándose en dos características:

- velocidad a la que trabajan los componentes
- facilidad de conexión entre sí

Esto quiere decir que los componentes que trabajan a alta velocidad, como ser la memoria y el conversor, deberían estar lo mas cerca posible.

Así mismo, los buses son de más fácil conexión cuando los componentes tienen una distribución de patas similar y se encuentran juntos.

Dado que la cantidad de componentes no nos permitía colocarlos todos en una misma placa, hemos tenido que separarlo en dos placas. Esto implica tener que decidir qué bloques o componentes colocar juntos en una misma placa y cuales no.

Como mencionamos antes, una de las pautas para esta decisión es la velocidad de trabajo de cada bloque.

Basándonos en esto, decidimos que toda la parte de control del sistema estaría en una placa y la parte de alta velocidad en la otra.

La distribución entonces fue la siguiente:

- Placa de captura y alta velocidad:
  - Buffers de entrada
  - Conversor AD
  - Memorias
  - Lógica adicional
- Placa de control:
  - PIC
  - Contador
  - Buffer bidireccional
  - Lógica adicional

En cuanto a la primer placa, ésta se dedica exclusivamente a la captura y almacenamiento de los datos, con la sola excepcion de los contadores. En ella también se encuentra una mínima lógica adicional, como ser algunas compuertas, qué deben de estar en la misma placa ya que de lo contrario éstas señales deberían ir a la otra placa y luego volver, aumentando el recorrido y las posibilidades de interferencias.

La ubicación de los contadores no nos dio opción, y tuvieron que ser colocados en la segunda placa.

Dado que el diseño esta localizado en que el microcontrolador solamente realice tareas de control, ninguna de las señales que éste controla son de tan alta velocidad como las que se manejan en la etapa de captura.

Realización, prueba y experiencia

Esta fue una de las etapas más tediosas y lentas del proceso, ya que hubo que soldar muchos cables y cada uno de ellos tuvo que ser hecho a medida.

Hubo que prestar mucha atención en que las soldaduras sean buenas, que no hayan corto-circuitos, como así también, tener precaución con el soldador de no quemar el recubrimiento aislante de los cables, ya que así también se podrían producir falsas conexiones.

Una vez planeada la ubicación de los componentes dentro de la placa, se colocan primero los zócalos de los integrados, y luego a partir de ellos se comienza con el soldado de los cables para interconectar como corresponde a cada uno de los componentes. Aquí hay que tener especial cuidado en no confundir los pines de los circuitos integrados, ya que eso puede provocar un mal funcionamiento, un corto circuito, o incluso la ruptura de algún componente.

Luego hay que interconectar las dos placas, ya sea con buses (cable flat para buses de datos) o con cables individuales, los que suelen ser de control, a excepción de la señal de reloj y las alimentaciones.

Una vez finalizada la construcción de las placas, hay que hacer una prueba exhaustiva de las conexiones. Este punto es necesario e imprescindible, ya que como mencionamos, una mala conexión puede provocar un mal funcionamiento, un corto circuito, o incluso la ruptura de un circuito integrado, siendo este último caso el peor.

El precio de cada componente y especialmente la disponibilidad de los mismos nos obligaba a estar seguros de su correcta ubicación y conexión antes de colocarlos y ponerlos en funcionamiento.

Como es normal, en la etapa de prueba hemos encontrado errores y cortos. Esto era de esperarse, ya que con tantos cables y circuitos integrados, resulta poco probable cometer un error. Vale aclarar que esto no hubiese pasado si hubiésemos optado por la opción de utilizar una placa impresa (PCB), ya que la

misma tiene todas las conexiones hechas en pistas de cobre, aunque no nos hubiese permitido realizar los cambios que hemos hecho sobre la marcha.

La conclusión que podemos sacar de haber trabajado de la forma en que lo hemos hecho es que la elección del tipo de placa fue la correcta, ya que es mucho más seguro y confiable que trabajar en un protoboard, y a la vez no es tan restrictiva como una placa impresa.

También como critica constructiva podemos decir que se podría haber comenzado con la construcción antes de lo que lo hemos hecho. Una vez que se tienen definidos los componentes que se van a usar, ya se está en condiciones de comenzar a colocar los zócalos en las placas. Luego a medida que se van definiendo las conexiones, se van soldando los cables. Nosotros hemos comenzado con la construcción una vez que teníamos definidas todas las conexiones. Aún así, y como hemos comentado previamente, tuvimos que realizar cambios de conexiones sobre la marcha, y aquí es donde demuestra la ventaja la placa universal sobre el PCB.

A continuación presentamos unas fotos del equipo. Aclaramos que se puede encontrar cierta *desprolijidad* en el cableado, especialmente en las conexiones entre las dos placas. Destacamos una vez mas que este punto será solucionado con la construcción de un circuito impreso.

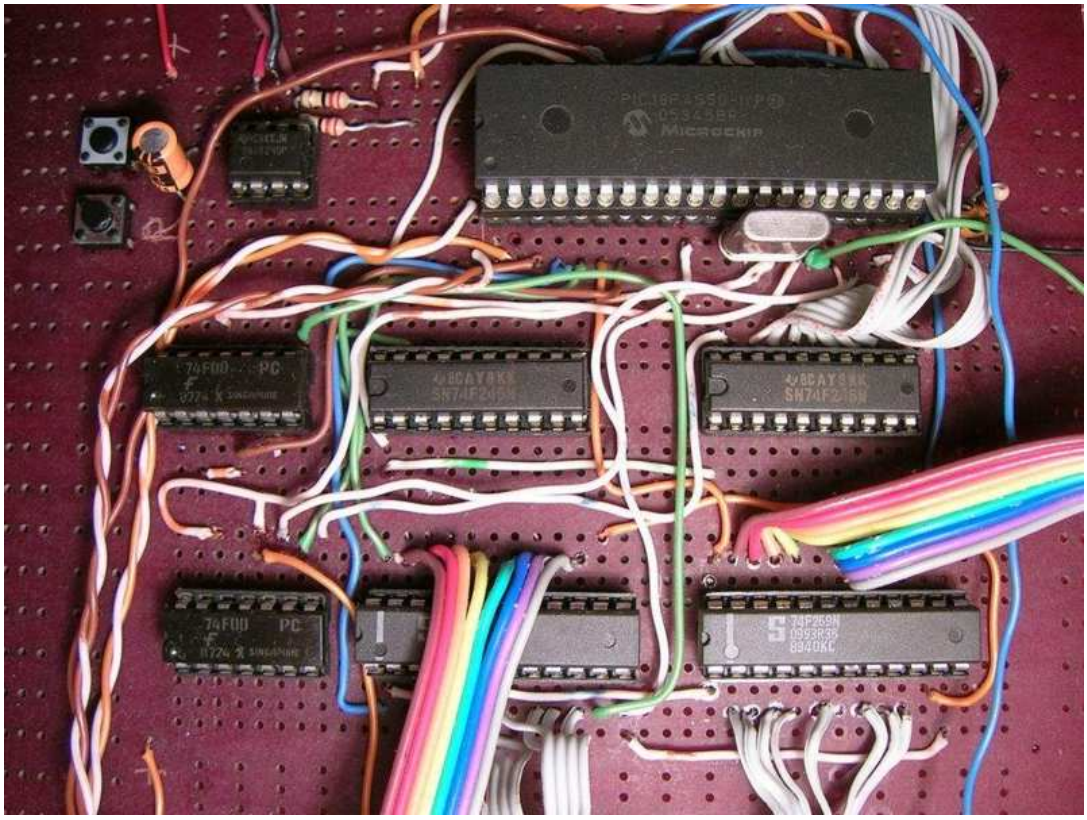


Fig 8.10 Placa de control: pic - buffers - contadores - compuertas

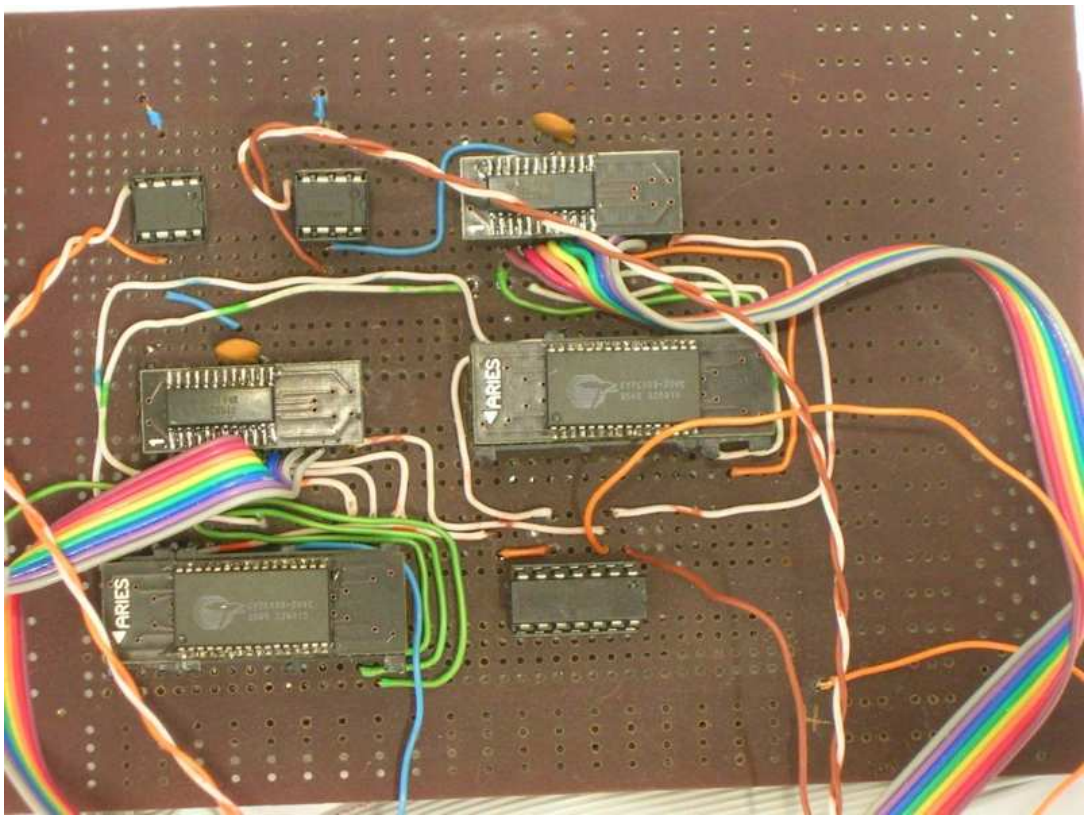


Fig 8.11 Placa de adquisición: conversores - memorias - entrada - compuertas

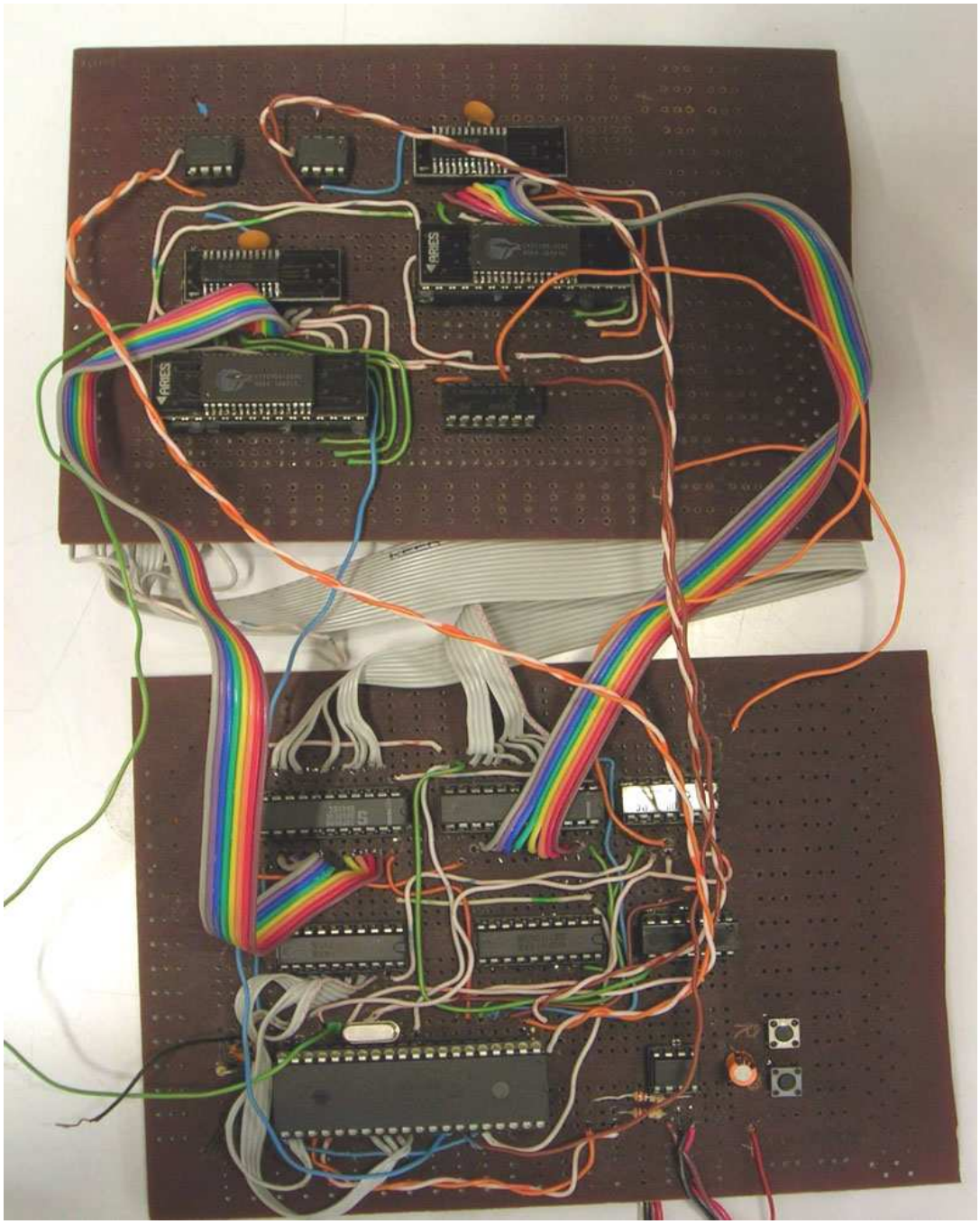


Fig 8.12 Equipo prototico completo

Carcaza

La compra de la carcaza fue una de las tareas realizadas al final del proceso de fabricación, ya que ésta solo aporta una mejora estética (y no funcional) al prototipo

En consecuencia, tuvimos que decidir rápidamente entras las alternativas disponibles en plaza. Optamos por una carcaza de plástico por su flexibilidad a la hora de realizar perforaciones para colocar los controles.

Fig 8.13 Carcaza del osciloscopio

Fig 8.14 Conectores BNC del osciloscopio

Compra de componentes

La compra de los componentes no es una etapa en si, sino que es una tarea que no tiene un momento definido. Una vez que se decide por el uso del cierto componente, el acto que le sigue es el de la averiguación de dónde se puede comprar y a qué precio. Aquí es donde uno se encuentra con las siguientes dificultades:

- Disponibilidad
- Precio

En nuestro diseño usamos muchos componentes que no se encuentran en plaza en el mercado uruguayo. Si bien unos cuantos los hemos conseguido en Uruguay, la haber disponibilidad de los mismos en otros países y a mejor precio, nos hemos planteado la posibilidad de comprarlos en el exterior por una cuestión de costos.

Una vez que habíamos decidido la utilización del PIC como controlador principal, compramos uno para poder comenzar con las pruebas.

En cuanto a los componentes que no se conseguían en Uruguay, consideramos Argentina y Estados Unidos como posibles proveedores de los mismos, dado que contamos con facilidades en estos dos países para comprar lo que precisemos y luego traerlo.

Aprovechando la ocasión de que teníamos que comprar ciertos componentes en Estados Unidos por la falta de disponibilidad en Uruguay y Argentina, hemos decidido también comprar ciertos componentes que se conseguían en Uruguay, pero en el exterior tenían un costo menor, como por ejemplo el PIC.

En el caso del PIC, el primero que compramos lo compramos en Uruguay, pero luego hemos comprado otros 2 mas en Estados Unidos, aprovechando que ya teníamos que hacer una compra de otros componentes.

Esto nos da cierta ventaja en cuanto a la facilidad para conseguir componentes que de otra forma serían casi imposibles o muy costosos. Sin embargo esto trae aparejado un problema, que es la demora.

En cuanto a las compras en Argentina, nos resultaba más fácil que con las de Estados Unidos, ya que contábamos con una persona que viajaba constantemente, con una frecuencia de una vez cada dos semanas en promedio, lo que nos permitía traer componentes con poco tiempo de demora.

De todas maneras, hubieron compras que eran necesarias hacerlas en Estados Unidos, y aquí es cuando se vio demorado el proyecto. No siempre era posible contar con una persona que viaje cuando el proyecto lo precisaba, sino que había que coordinar el proyecto con estos viajes, y además, no siempre esta persona podía traer estos componentes, con lo cual tuvimos demoras de hasta casi dos meses en la espera de éstos.

Sin embargo, la forma en la que hemos procedido la creemos correcta. Siempre se tiene la posibilidad de recurrir a un *Courier* de entrega rápida y

directa, pero es claro que esto insume un costo mayor, el cual tratamos de evitar. Como lección podemos decir que hubiese sido mejor poder prever los componentes con mas anticipación, y así realizar las compras y traslados con tiempo y evitar que el proyecto se demore.

Puesta en funcionamiento

La puesta en funcionamiento del equipo fue progresiva. Consideramos fundamental colocar los componentes de a uno por vez e ir probando las conexiones y su funcionamiento. Hay que prever que puedan existir errores en la placa aún teniendo en cuenta las etapas de fabricación y prueba. Esta forma de proceder tiene como objetivo la preservación de los componentes, la mayor facilidad en la detección de errores, y la aislación de problemas.

Una vez que se tienen todas las conexiones hechas, y el circuito encargado de generar la señal de reloj esta listo, procedimos primero a colocar el microcontrolador. Luego de ver que éste se encontraba funcionando y teníamos comunicación con él, comenzamos a colocar el resto de los componentes.

Primero fueron las compuertas discretas. Esto ha sido así, debido a que para poder colocar el resto de los componentes, se precisan listas las señales de control. Luego colocamos el buffer bidireccional. A través de él, podíamos medir y comprobar si efectivamente el bus de entrada/salida se encontraba funcionando correctamente, como así también las señales de control involucradas.

Para evitar interferencias y especialmente para hacer que los cables que conectaban las dos placas no actuasen de antenas, decidimos colocar los amplificadores de entrada, pero no alimentarlos. Es decir, las señales analógicas de entrada que también se encontraban conectadas al microcontrolador se encontraban flotantes y sobre un recorrido largo de cable. Al colocar los amplificadores pero no alimentarlos, realmente no se estaban utilizando estos circuitos, pero si servían para reducir la interferencia posible. Las conexiones de tierra estaban hechas entre las dos placas desde el principio, y así también para todos los componentes de cada placa.

El siguiente paso fue colocar los contadores. Estos contadores son controlados por el PIC, y su señal de reloj puede provenir tanto desde el cristal como desde una señal de control del PIC, la cual tiene como objetivo tener una señal de reloj mas lenta y configurable.

Una vez que comprobamos el buen funcionamiento del contador, probamos la función de *presetear* el contador, es decir, darle un valor de contador desde el cual éste debe comenzar a contar. Aquí se probó por completo el funcionamiento del bus bidireccional, ya que a través de éste se realiza la operación de carga del contador. Aquí nos hemos encontrado con un problema, que es que la carga del *preset* del contador se realiza sincrónicamente, es decir, cuando recibe un pulso de reloj. Para el contador de los bits menos significativos esto no sería un problema ya que la señal de reloj es controlada.

Pero para el segundo contador, el de los bits más significativos, esto era un problema ya que en realidad la señal de reloj que éste recibe proviene del contador que lo precede (contadores en cascada). Este problema y su solución están explicados en la siguiente sección: Problemas.

A continuación procedimos a colocar las memorias. Este paso fue crucial, ya que las memorias eran los primeros componentes que se colocaban en la placa de adquisición. Hasta este momento se había trabajado con una sola placa, y este era el momento de probar la segunda placa. Aquí había que comprobar la alimentación, si existían corto circuitos, caídas de tensión, interferencias, etc. Por suerte tuvimos un resultado muy positivo. Salvo algún problema de falso contacto (más bien producido por los zócalos, ver: Problemas con los zócalos), hemos tenido un excelente resultado. Así hemos logrado conectarnos satisfactoriamente con la segunda placa, la de alta velocidad, la de captura y adquisición. Cabe aclarar que para realizar este paso fue necesario conectar la alimentación de la placa en cuestión, ya que si bien la conexión de tierra estaba hecha, no así la de tensión.

Al conectar la alimentación de la placa de adquisición automáticamente el amplificador de entrada comenzó a funcionar. Esto no es problema ninguno, y sólo optamos por no dejar *flotando* la entrada del amplificador y conectarla a tierra. Como siguiente paso, conectamos los conversores. Este momento marca un punto especial en el proyecto, y la prueba del hardware. Aquí es donde se va a apreciar si todos los componentes logran trabajar en armonía y logran comunicarse con facilidad. Al igual que en algún otro caso anterior, hemos tenido problemas con las conexiones, pero como adicional, hemos verificado también el correcto funcionamiento de la memoria, ya que ahora podíamos comprobar con mayor confianza los datos que se guardaban y que se leían. Aquí detectamos un error en las señales de control de la memoria, ya que no se estaba realizando correctamente el procedimiento para su lectura (Ver la siguiente sección).

Finalmente una vez solucionados los problemas detectados, a través de la interfaz de consola que nos provee el osciloscopio, obtuvimos muestras que coincidían con lo esperado. Es decir, que cuando la entrada se colocaba a tierra, se obtenían valores todos cercanos a cero (posibles ruidos, etc), y cuando utilizamos una señal cuadrada de un generador, obteníamos valores coherentes, o sea, una serie de valores altos, luego otra serie de valores bajos, y así sucesivamente.

Consideramos muy satisfactorio el proceso de fabricación de la placa, en el cual obtuvimos pocos errores y fallas, y consumió poco tiempo llevarlo al funcionamiento deseado.

Para estas pruebas utilizamos un reloj de 8 megaciclos, el cuál es el mínimo posible para que funcionen todos los componentes, pero la velocidad esta lejos del objetivo final. Es natural suponer que a menor velocidad las cosas funcionen mejor, y en caso de que existan problemas, sean mas fácil detectarlos, o mismo al comprobar el funcionamiento de los componentes. La intención es que una vez que se logre hacer funcionar el equipo correctamente

a esta velocidad, se irá incrementando la velocidad del reloj y así cada vez acercarnos más a la velocidad de trabajo propuesta como objetivo (40 Mhz).

Depuración por hardware

La posibilidad de depurar por hardware es una herramienta fundamental en la etapa de implementación.

Esta herramienta nos permite verificar y ejecutar *paso a paso* el programa cargado en el microcontrolador. Nos permite también monitorear diferentes variables, contenidos de memoria, o estados de las patas del PIC.

La conexión del ICD2 se realiza a través de un cable de 6 hilos. Para facilitar la conexión, utilizamos un cable con conectores RJ12 en sus extremos, y zócalos en las placas.

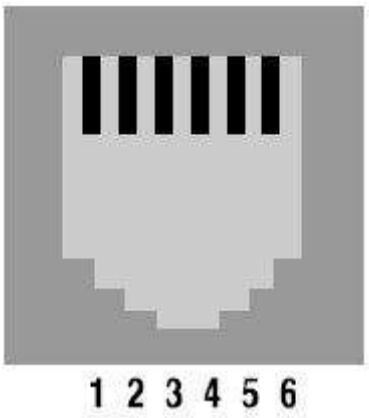


Fig 8.15 Pinout del conector RJ12 del ICD2

Señal	Zócalo ICD2	Zócalo placa
Vpp	6	1
Vdd	5	2
GND	4	3
PGD	3	4
PGC	2	5
NC	1	6

Tabla 8.1 Pines del conector RJ12 del ICD2

Descripción de los pines:

- **GND** - Voltaje de referencia.
- **Vdd** - Tensión de alimentación positiva. Esta alimentación es la que energiza al microcontrolador.
- **Vpp** - Tensión de programación. Debe conectarse al puerto MCLR del PIC. Este voltaje debe ser superior a Vdd para que el microcontrolador ingrese al modo de programación.
- **PGC** - Señal de reloj de la transmisión de datos serie.
- **PGD** - Línea de datos. Esta conexión es bidireccional, y permite la comunicación de dos vías entre el PIC y la interfaz de programación

Para poder entrar en el modo *debug* es necesario realizarlo desde el programa, el MPLAB. Allí, seleccionando las opciones correspondientes, se programa al PIC con una versión "modificada" del programa original. Esta modificación, hecha automáticamente por el software, contiene las rutinas necesarias para poder realizar el depurado *on-line*, ya que de tratarse de la operación normal del PIC, éste ejecutaría el programa de forma normal, sin interrupciones ni posibilidad de monitoreo. Una vez finalizado el proceso de depurado, se debe volver a programar el microcontrolador con la versión *normal* del programa.

Hemos incluido estas conexiones en nuestro equipo, permitiendo el depurado _en linea_ del mismo. Nos da la posibilidad de una conexión simultánea del ICD2 y el ordenador con la placa.

La ventaja fundamental es que podemos observar el comportamiento del sistema cuando este se encuentra conectado al PC y ejecutando las operaciones solicitadas. Nos ha facilitado la detección de errores tanto de programación como de hardware.

Corriendo paso a paso las instrucciones, pudimos verificar si el puerto de control del PIC efectivamente estaba realizando las operaciones solicitadas, y si los componentes conectados a él también lo hacían.

Problemas

NAND no oscila

En una de las primeras etapas del circuito, la etapa osciladora va directamente conectada hacia el microcontrolador, pero a su vez va conectada en paralelo a una NAND (funcionando como NOT). En algún momento durante la construcción y prueba de la placa, esta compuerta tenía como salida la señal de reloj, obviamente invertida. Luego, más avanzado el proceso de colocación de componentes y prueba, observamos que ya su salida no oscilaba, sino que se mantenía constante con valor cero. Buscando los posibles orígenes de este problema, vimos que su entrada estaba funcionando perfectamente, y que todos los circuitos así también lo hacían. Sin embargo, habría que seguir investigando, ya que el problema no radicaba en un componente defectuoso, sino como producto de cambios que podríamos haber hecho en la placa en la etapa de prueba y detección de errores.

Observamos como la amplitud de la señal de reloj podía ser mayor o menor, pero aún no encontrábamos el parámetro que hacía que esto cambie. Al medir las tensiones de la amplitud de dicha señal y comparándolas con las especificaciones de la compuerta, vimos que efectivamente se trataba de un tema de amplitudes. Ésta señal no tenía la amplitud suficiente para que la compuerta detecte un cambio de estado en su entrada.

Más específicamente, las mediciones daban valores de *1.5v a 4v* para la amplitud de la señal de reloj como entrada de la compuerta. Luego, en la hoja de datos de ésta, se tiene como especificación que _V.high (min) = 2.0v y un V.low (max) = 0.8v_. Dado que la tensión del oscilador nunca era menor que

1.5v, queda claro que la compuerta entonces nunca detectaría un cambio de estado en su entrada.

Realizando modificaciones en los bits de configuración del PIC logramos hacer que estas amplitudes cambien, o mejor dicho, sean más cercanas a cero. Sin embargo no llegamos a lograr que bajen de 1v. Por otra parte, si desconectábamos la NAND como carga del circuito oscilador, éste tenia amplitudes entre *0.6v y 3.8v*. Esto implica que las amplitudes tienen relación directa con la carga y las impedancias que dicho circuito tiene.

Hasta hemos probado quitarle una resistencia de 1M ohm que el oscilador tenía, la cual en ciertos documentos hemos visto, pero en otros no estaba incluida. Al quitarle esta resistencia, y sin tener la carga adicional de la compuerta, la amplitud tenia una variación entre *0.3v y 3.4v*. De todos modos, al volver a conectar la compuerta, estas amplitudes volvían al mismo estado detectado al principio, sin lograr que la compuerta funcione.

Finalmente, detectado que se trataba de un problema de impedancias y no de configuración, decidimos probar de obtener la señal y cargar al oscilador en la otra pata que este tiene. Es decir, el circuito oscilador tiene dos bornes de conexión, entre los cuales se encuentra el cristal y componentes adicionales. Estas dos patas van conectadas al PIC en sus puertos OSC1 y OSC2. Durante toda esta etapa estuvimos obteniendo la señal de reloj del puerto OSC1, pero llegado este punto, optamos por probar conectar en el puerto OSC2.

Aquí se resuelve el problema. Evidentemente el puerto OSC1 no esta preparado para tener cargas en paralelo y el puerto OSC2 sí.

Las amplitudes obtenidas ahora son entre *-0.2v a 4.8v*. Esto es más que suficiente para que la NAND opere correctamente.

Además, ya casi no se observan diferencias cuando se conecta la NAND, es decir, que efectivamente se trataba de un tema de cargas, y de las características de cada puerto.

El siguiente problema que detectamos es que la salida de la NAND, la cual sería una señal cuadrada (se trata de un componente digital, por mas que su entrada sea analógica), pero ésta se encontraba considerablemente distorsionada. Dicha señal, si bien tenia la misma frecuencia y ostentaba ser lo esperado, era muy curvada y no parecía ser tanto una señal digital cuadrada.

No sabemos si es un tema de velocidad, si es que la gran amplitud de la señal de entrada genera este comportamiento, o si es que se debe utilizar una compuerta del tipo *schmidt-trigger*.

A continuación se puede observar el comportamiento al cual hacemos referencia. Esta imagen es una foto tomada del osciloscopio con el cuál hemos trabajado.

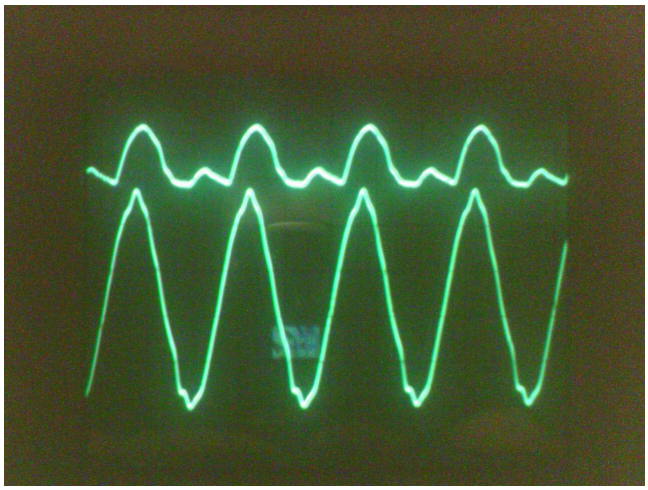


Fig 8.16 abajo el oscilador, arriba la salida de la NOT (NAND)

Para entender un poco mejor la razón por la que al conectarse a una de las dos terminales de oscilación el circuito no funcionaba pero con el otro terminal sí, hemos hecho una búsqueda un poco mas exhaustiva en la documentación del PIC. Allí encontramos el siguiente diagrama:

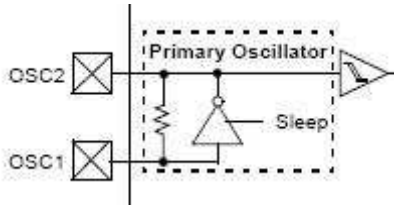


Fig 8.17 Bloque de entrada del oscilador

Esto podría ser la explicación al problema, ya que el puerto OSC1 esta configurado como la entrada de un buffer, y por lo tanto con alta impedancia, mientras que el terminal OSC2 es una salida, pudiendo provocar incompatibilidades al intentar conectarlo en paralelo con otros componentes.

Este comportamiento, y en especial la forma de onda no tan perfecta (cuadrada) no nos ha generado problemas desde este punto en adelante, sin embargo luego hemos visto con el osciloscopio que esta señal habría mejorado, y creemos que esto es debido al avance que ha habido en la conexión de los componentes. Es decir, hasta que no hemos logrado hacer funcionar correctamente esta compuerta nos hemos visto imposibilitados de avanzar con la conexión y prueba del resto de los componentes. Al haber solucionado este tema, es que hemos continuado conectando el resto del circuito. Es así que avanzada un poco más la implementación del circuito hemos observado que el comportamiento mencionado había mejorado aún más.

De todos modos consideramos importante estudiar las cargas que puede tener esta compuerta (NAND), para poder corroborar si es que se trataba de un tema de sobrecarga. Esta compuerta estaba cargada con los siguientes componentes:

- 2 entradas de NAND (igual compuerta, una en control de memoria y otra en adquisición)
- 2 entradas de reloj de los conversores A/D

Las salidas de las NAND tienen las siguientes características: Fan Out: I(OH) = -1mA ; I(OL) = 20mA

Las entradas de las NAND: Fan Out: I(IH) = 20uA ; I(IL) = -0.6uA Características eléctricas en continua: I(IH) = 5uA ; I(IL) = -0.6mA

Y finalmente, las entradas de los conversores A/D: I(IH) = 5uA ; I(IL) = 5uA

Este estudio demuestra que las salidas de la compuerta NAND son ampliamente capaces de poder manejar los 4 componentes que tienen en su salida.

Problemas de oscilación

El oscilador también ha presentado otro problema. Cuando el circuito se encontraba conectado a la placa ICD2 (programador y depurador), oscilaba perfectamente. Sin embargo, al conectar el equipo al puerto USB del PC, encontramos un funcionamiento errático, donde en ciertas ocasiones oscilaba y en otras no. Hemos buscado las razones de este comportamiento, pero realmente nos ha costado. Hemos pensado en que podía ser un tema de interferencias, consumo de corriente del puerto USB, malas conexiones, etc.

Finalmente la razón de este comportamiento no radicaba en un problema de hardware, sino de los bits de configuración del PIC. Específicamente provenía de la configuración del oscilador en estos bits. Aquí se debe especificar con qué oscilador se trabaja, el tipo, la velocidad, etc. Al configurar correctamente estos bits, no ocurrió nuevamente este comportamiento.

Retardo de contador lleno

Durante el proceso de prueba de funcionamiento de los contadores, observamos que estos no se detenían cuando llegaban a su máxima cuenta.

En el diseño se contempla que cuando ambos contadores llegan a su cuenta máxima, un pulso es generado en la pata FULL (contador lleno) del contador mayor, y este pulso es detectado por el microcontrolador generando una interrupción de alta prioridad. Esta interrupción, entre otras tareas, detiene el contador. Al tratarse de una interrupción, se podría decir que se obtiene uno de los tiempos de respuesta más bajos posibles. Sin embargo, esta respuesta no es inmediata, sino que consume un cierto tiempo de procesamiento.

Las mediciones dieron los siguientes resultados:

- Corriendo el contador a 8Mhz y el PIC a 24Mhz, el retardo desde que el contador se llena hasta que es detenido es de 16 cuentas (16 ciclos de 8Mhz).

- Corriendo el contador a 8Mhz y el PIC a 48Mhz, el retardo baja a la mitad, retardo de 8 ciclos de 8Mhz.

8Mhz -> 125 ns 24Mhz -> 41.6 ns 48Mhz -> 20.8 ns

16 ciclos a 8Mhz -> 2us 8 ciclos a 8Mhz -> 1us

2us a 48Mhz -> 48 ciclos 1us a 24Mhz -> 48 ciclos

Haciendo algunas cuentas, se llega a que el retardo que tiene el microcontrolador en atender la interrupción y detener el contador es de 48 ciclos.

De acuerdo a la velocidad a la que corra el contador, y a la que se ejecuten las instrucciones del PIC, este tiempo cambia, y se puede observar en las mediciones obtenidas.

Es evidente que sólo se pueden mejorar estos tiempos vía software (o firmware), pero en principio esto no implica un problema.

El único efecto que esto produce es que se sobreescriban las *n* muestras (16, 8, o bien, la cuenta que resulte de los 48 ciclos del PIC) obtenidas y guardadas en memoria. Teniendo una memoria de gran tamaño, y siendo tantas las muestras que se toman de la señal de entrada, creemos que la citada cantidad no es relevante y puede despreciarse. El único cuidado que hay que tener es el de no tenerlas en cuenta y considerarlas como muestras no validas, descartándolas y suponiendo el resultado los valores guardados a partir de la muestra número *n*.

Zócalos

Los circuitos integrados seleccionados para componer el sistema completo fueron en su mayoría de encapsulado DIP, lo que facilita enormemente su uso, especialmente en placas universales (ver: Fabricación). Sin embargo los conversores y las memorias no las pudimos obtener en este tipo de encapsulado.

Los conversores TLC5540 de Texas Instruments tienen un encapsulado SOIC-24. Para este componente debimos comprar un zócalo adaptador de SOIC-24 a DIP-24. Afortunadamente este tipo de encapsulado (frente al SOJ) es de mas fácil soldadura, y el zócalo adaptador calzó perfecto en los zócalos DIP estándar, con los cuales construimos la plaqueta.

Las memorias Cypress CY7C-109B tiene encapsulado SOJ-32 y debimos comprar zócalos adaptadores de SOJ-32 a DIP-32. Sin embargo en este caso nos encontramos con un gran problema. El zócalo adaptador (en su parte DIP) tenía una distancia entre sus lineas paralelas de 400-mil (1 mil = 1/1000" -> 400-mil = 0.4" = 10.16mm), mientras que la separación standard es de 300-mil (7.62mm). O sea, en la fabricación de la plaqueta, hemos dispuesto a todos los componentes es zócalos DIP standard, y luego a medida que llegaba el

momento de colocarlos, el zócalo ya se encontraba soldado. Ahora, si tenemos ya soldado e instalado un zócalo de 300-mil y tenemos que colocar en él un componente con separación de 400-mil, está claro que no va a ser posible. Dado que consideramos más complicado y con posibilidad de error el cambiar el zócalo de la placa, ya que este ya se encontraba soldado (con todas las conexiones adyacentes también soldadas) optamos por construir (en primera instancia) un adaptador "casero" de DIP-32@400-mil a DIP-32@300-mil.

Esto al principio provocó fallas intermitentes en la conexión desde el integrado hacia las pistas de cobre, pero se solucionó ajustándolo mejor y haciendo mas presión sobre los componentes. Resulta evidente que estos zócalos deben de reemplazarse por sus correspondientes, pero la solución optada ha sido por cuestiones de tiempos y facilidad.

Preseteo de contadores

Los contadores que hemos seleccionado tienen la funcionalidad de *presetearlos*, es decir, cargarle un valor predefinido al contador y luego éste comenzará a contar a partir de dicho valor. El procedimiento de carga de dicho valor es sincrónico, es decir, que una vez que se pone el valor deseado en las patas correspondientes y la señal de carga esta seteada, se debe dar un pulso en la señal de reloj para que dicha carga tenga acción. A diferencia de los componentes asincrónicos, los cuales una vez que con la pata de control de indica la carga del valor de preseteo, no requieren de un pulso de reloj para llevar la acción a cabo.

Esto no sería un problema para el contador de los bits menos significativos, ya que la señal de reloj de este componente esta controlada por el PIC. Sin embargo, el diseño de contadores en cascada, lo que permite aumentar la cantidad de bits del contador, hace que la señal de reloj del contador más significativo sea controlada por el contador de los bits menos significativos.

Este proceso hace que cada vez que el contador mas chico llega a la última cuenta (el último numero antes de volver a cero), dé un pulso en su pata de *FULL* (contador lleno), y esta es la que hace que el contador mayor aumente en 1 su cuenta.

Como vimos, el contador mayor esta controlado en cuanto a su sincronismo y señal de reloj por el contador menor, y esto complica el proceso de preseteo, ya que no tenemos forma directa de controlar la señal de reloj de dicho contador (recordar que son sincrónicos).

Entonces la solución que hemos encontrado para resolver este problema es haciendo que cada ciclo de preseteo de contadores cargue en el contador menor el mayor valor posible, luego cargarle al contador mayor el valor deseado (esto no ocurrirá hasta un nuevo pulso de reloj), y posteriormente enviar un pulso de reloj al contador mayor para que este avance en su cuenta. Dado que lo habíamos cargado con el máximo valor posible en su cuenta, un nuevo pulso de reloj hará que este vuelva a cero y de un pulso en su señal de *contador lleno*. Esta señal es la que controla el reloj del contador mayor, y por

consiguiente aquí se producirá la carga efectiva de los datos en este contador, ya que como hemos dicho, este contador es sincrónico y realiza las operaciones en cada pulso de reloj.

Finalmente se carga al contador menor con el valor deseado y se termina con el proceso de preseteo.

Si bien esta solución no sería la ideal a primera vista, consideramos que una solución que se pueda hacer por software, en vez de realizar cambios en el hardware, es mejor y más fácil de llevar a cabo.

Control de lectura de memoria

La memoria tiene varias formas de ser operada y controlada, diferentes señales proveen combinaciones diferentes qué hacen que el componente accione como se espera. Por ejemplo, podríamos decir que puede ser controlada por la pata de lectura, o por la de escritura, o por la de habilitación. De acuerdo con qué configuración se esté utilizando, entonces dichas señales deben de tener cierto valor para el funcionamiento correcto.

Habiendo tales diferencias, hemos cometido un error de diseño sobre cómo debería ser la señal de lectura.

Mientras probamos el funcionamiento de la placa observamos un funcionamiento en este componente que no coincidía con el esperado. Al depurar en tiempo real qué es lo que estaba ocurriendo, detectamos que el problema tenía origen exactamente en el comportamiento de la memoria frente a lo que las señales de control suponían. Aquí entonces revisando más a fondo las formas de control detectamos que la señal de lectura estaba invertida, y por lo tanto, la solución inmediata fue la de eliminar una de las compuertas NOT (hecha con NAND) del camino de control de dicha señal.

Una vez realizado este cambio, la memoria funcionó tal como se esperaba.

Cambio de elección de patas del PIC

Mientras se comprobaba el funcionamiento de todos y cada uno de los componentes, detectamos que alguna de las veces que un componente no actuaba como se esperaba el problema provenía del microcontrolador. El PIC no realizaba lo que se le solicitaba. Ciertas patas del PIC que debían cambiar de valor no lo hacían, incluso al depurar en tiempo real por hardware.

Algunos de estos comportamientos anómalos se debían a una mala definición o programación de los puertos de entrada/salida del microcontrolador. Sin embargo otros casos se debían a la mala elección por nuestra parte, donde por ejemplo ciertos puertos utilizados para interrupciones no podían ser utilizados como patas de control standard.

En estos casos, debimos recablear las señales buscando algún otro puerto disponible en el PIC. Esto implica tener que desoldar un cable, extraerlo, y

volver a colocar otro de las medidas necesarias y entre el viejo componente y el nuevo puerto del PIC. Esta operación hay que realizarla con sumo cuidado ya que los procesos de desoldar y volver a soldar pueden provocar cortes de pistas, corto circuitos, y otros problemas.

Una vez que realizamos los cambios correspondientes, y se ha reprogramado el microcontrolador con la nueva configuración, el funcionamiento fue óptimo.

Aquí es donde vuelven a aparecer las ventajas de utilizar una placa universal en vez de un circuito impreso. La acción de hacer un cambio de último momento en la selección de puertos implica que la conexión entre un componente y otro cambie de recorrido, tenga un destino diferente, que el camino sea otro. Esto en una placa impresa podría ser imposible de llevar a cabo, a menos que se utilicen cables para reemplazar la pista de cobre. Nuevamente consideramos como buena la elección de utilizar una placa universal.

Conexión simultánea al ICD2 y al puerto USB

La mayor ventaja del ICD2 es poder depurar lo que está pasando en la placa directamente sobre el hardware, lo cual, en contraste con una simulación, es un excelente recurso para encontrar problemas en el funcionamiento.

Al programar la placa, ésta queda por defecto en modo reset (el MPLAB la deja configurada así). Por lo tanto una vez programada la placa se debe sacar del modo reset yendo a **Programmer -> Release from reset**. Esto hace que la placa corra simultáneamente conectada al puerto USB y al ICD2.

Para poder depurar el funcionamiento es necesario seleccionar el modo Debugger en lugar de Programmer, y luego ejecutar el programa yendo a Debugger -> Run (también pulsando la tecla F9).

Problemas de programación del ICD2

En repetidas oportunidades encontramos dificultades al programar el PIC, esto era en la etapa de verificación de la programación.

Es decir, el proceso de programación consta de la verificación del dispositivo y la conexión, luego se programa por completo el dispositivo, y finalmente se hace una verificación del contenido.

Cuando existe una diferencia entre lo programado y lo verificado, puede ser o bien que haya sido mal programado, o que el proceso de verificación no se haya completado correctamente. Dado que después de una programación no satisfactoria el dispositivo no funcionaba, es claro que la falla estaba en la etapa de programación.

Varias veces hemos tratado de encontrar la raíz del problema, pero sin poder llegar a ninguna conclusión.

De hecho, se podría decir que habíamos encontrado una forma de hacer que vuelva a su funcionamiento normal, y era reiniciando el PC y recolectando todos los equipos. Esto nos hizo suponer que se trataba de un problema en el PC, o mejor dicho, de algún inconveniente con el puerto USB.

Sin embargo, mas adelante descubrimos que el problema radicaba en el regulador de tensión del ICD2. Luego de un largo periodo de operación este componente recalentaba y producía el mal funcionamiento. Creemos que una de las posibles razones de que el regulador aumente su temperatura es la calidad de la fuente de alimentación que utilizamos. Luego de detectado el problema se resolvió añadiéndole un disipador al regulador de tensión, o incluso colocando un ventilador disipador próximo a él.

Referencias

- Tipos de placas
  - http://webdiee.cem.itesm.mx/web/servicios/archivo/tutoriales/protoboard/index.htm
  - http://www2.ing.puc.cl/~dmery/arqui/el_protoboard.pdf
  - http://personales.ya.com/lcardaba/projects/placas/placas.htm
  - http://olmo.cnice.mecd.es/~adog0004/siete.htm
  - http://www.pablin.com.ar/electron/cursos/pcb/
  - http://www.electronica2000.com/tabletas/tabletas.htm
  - http://perso.wanadoo.es/chyryes/glosario/PCB.htm
  - http://www.hostear.com/google/enciclopedia-virtual-informatica/circuito-impreso.php
  - http://www.pcbhome.com/glosario.htm
  - http://personales.ya.com/lcardaba/projects/placas/placas.htm

- ICD2
  - http://www.embedinc.com/picprg/icsp.htm
  - http://microcontrollershop.com/product_info.php?products_id=367
  - http://www.embedinc.com/easyprog/
  - http://www.piclist.com/techref/piclist/freeicd/index.htm
  - http://www.embedinc.com/products/index.htm

- Microchip MPLAB (programador y depurador del PIC18F4550)
  - http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en019469&part=SW007002
- Microchip MPLAB C18 (compilador C para el PIC18F4550)
  - http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en010014

Capítulo 9. Manual de usuario

Contenido de este capítulo

Introducción

En este capítulo se presenta el manual de usuario del osciloscopio, el cual incluye una descripción de sus características y la guía de como utilizar el software.

La última versión del software se puede bajar de la siguiente página:

http://pablohoffman.com/oscusb/software/

Información sobre el producto

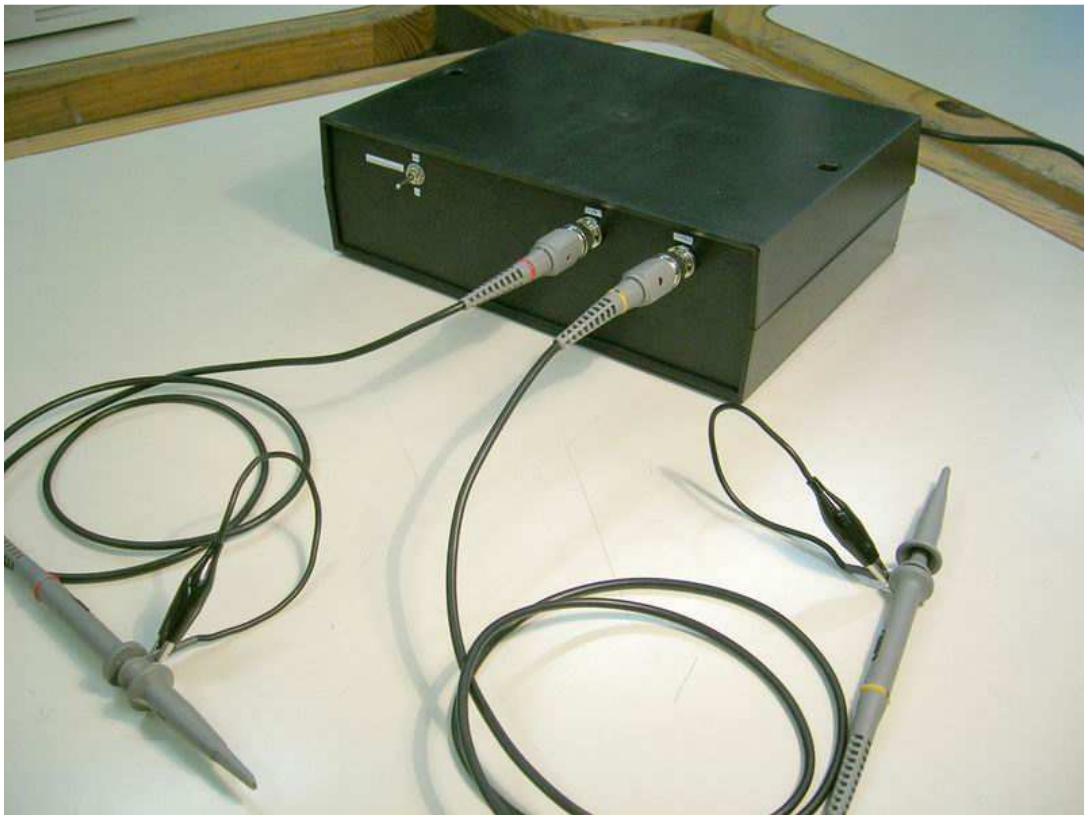


Fig 9.1 Foto frontal del osciloscopio

Requisitos mínimos del sistema

Para que funcione el osciloscopio se necesita un computador con las siguientes características mínimas:

Procesador	Pentium o equivalente
Memoria	32 Mb.
Espacio en disco	5 Mb.
Sistema operativo	Microsoft Windows 98 SE, ME, 2000, XP o superior Linux 2.6 o superior Mac OS X 10.2 o superior (no probado)
Puertos	Puerto compatible con USB 1.1 como mínimo

Tabla 9.1 Requisitos del sistema

Conectores y controles del osciloscopio

El osciloscopio posee varios controles y conectores que se detallan a continuación.

Panel frontal

Fig 9.2 Panel frontal del osciloscopio

Descripción de los controles del panel frontal (de izquierda a derecha):

Etiqueta	Tipo	Descripción
CHAN1	Conector BNC	para conectar la punta de osciloscopio del canal 1
CHAN2	Conector BNC	para conectar la punta de osciloscopio del canal 2
ETAPA DE ENTRADA	Llave	para habilitar/deshabilitar la etapa de entrada (también es necesario habilitar/deshabilitar la desde el programa)

Tabla 9.2 Controles del panel frontal

Panel trasero

Fig 9.3 Panel frontal del osciloscopio

Descripción de los controles del panel trasero (de izquierda a derecha):

Etiqueta	Tipo	Descripción
USB	Conector USB tipo B utilizado para conectar el osciloscopio a la PC con un cable USB estándar	
DEBUG	Conector RJ12	para depurar el funcionamiento del osciloscopio a través del

		programador/depurador ICD2
RESET	Botón	para resetear el osciloscopio
ON/OFF	Llave	para encender/apagar el osciloscopio
9V DC	Conector de corrriente	para conectar la alimentación al osciloscopio

Tabla 9.3 Control del panel trasero

Especificaciones

Ver Apéndice V: Especificaciones

Instalación del osciloscopio

A continuación se detallan los pasos de instalación a seguir para instalar el osciloscopio en su sistema operativo de preferencia.

Windows

1. Baje el software

1. Bajar la última versión del software de http://pablohoffman.com/twiki/pub/Oscusb/Oscusb
2. Descomprimir el archivo oscusb-sw-rX.zip (donde X es el número de revisión) en un algún directorio, por ejemplo c:\oscusb

2. Conecte el osciloscopio e instale el driver

Al conectar el osciloscopio por primera vez aparecerá el Asistente para instalación de hardware de Windows indicándole que se ha conectado un nuevo dispositivo a su PC. Siga los siguientes pasos para instalar el driver:

1. Seleccionar **Instalar desde una lista o ubicación específica (avanzado)** y haga clic en **Siguiente**
2. Hacer clic en Examinar y seleccione la carpeta c:\oscusb\driver\WIN98-2K-XP
3. Hacer clic en Siguiente
4. Windows dirá que se ha reconocido el nuevo dispositivo **Osciloscopio USB**
5. Hacer clic en Finalizar

Linux

1. Baje la última versión del software de http://pablohoffman.com/oscusb/software/
2. Descomprima el archivo oscusb-sw-rX.zip (donde X es el número de revisión) en algún directorio, por ejemplo: /usr/local/oscusb

Uso del osciloscopio

Tanto en Windows como en Linux el programa tiene la siguiente interfaz:

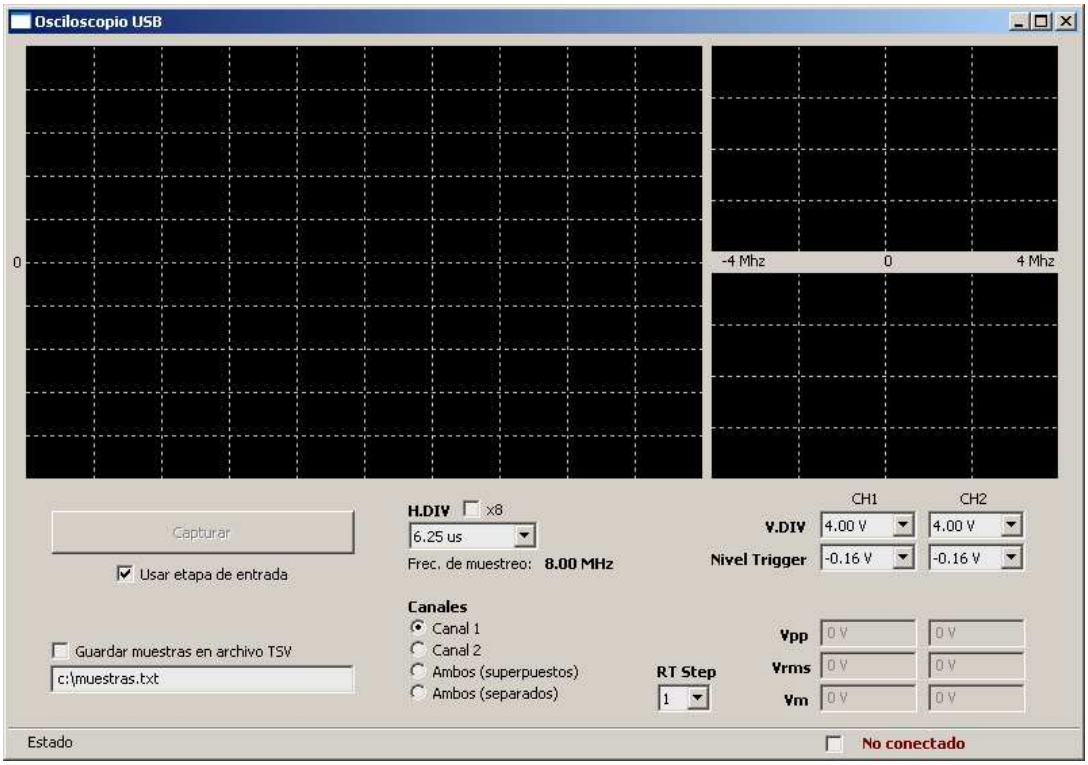


Fig. 9.4 Interfaz del software del osciloscopio

Como se puede observar la interfaz cuenta con 3 displays y varios controles. Los cuales se explican a continuación:

Display en el tiempo

El display más grande de la izquierda es donde se podrán observar las señales capturadas en el tiempo, ya sea de uno o ambos canales (separados o superpuestos). El mismo está dividido por un grilla de 10x10 y su escala depende del valor de los parámetros DIV y DIV.

La posición del cero en el display de la señal en el tiempo depende de ciertos factores, como ser:

- cantidad de canales
- etapa de entrada habilitada
- etc

Y cambia automáticamente al cambiar esas opciones con los controles.

Análisis espectral

Los displays pequeños de la derecha muestran el análisis espectral de las señales, respectivamente: canal 1 (arriba), canal 2 (abajo).

La frecuencia máxima de los displays de análisis espectral cambia automáticamente cuando se cambia la división horizontal a través del control DIV. El cero en dichos display siempre está fijo en el medio.

Controles

Los diversos controles de la aplicación (ubicados en la parte inferior de la interfaz) permiten configurar el modo de funcionamiento del osciloscopio, de forma análoga a como se haría en un osciloscopio analógico.

Botón capturar

El botón Capturar sirve para dar comienzo al proceso de captura. Es un botón de doble estado: para detener la captura hay que presionarlo nuevamente.

Al presionar el botón Capturar aparecerán las señales en el display principal y sus respectivos análisis espectral respondiendo a la configuración de los controles. Ambos continuarán actualizándose continuamente hasta que la captura sea detenida.

Como es de esperar, los controles pueden ser cambiados durante el proceso de captura. No es necesario detener la captura y volver a largar.

H.DIV

El control H.DIV permite configurar la división horizontal del osciloscopio, es decir, el valor en el tiempo que ocupa una división del display.

Debajo del control H.DIV se indica la frecuencia de muestreo a la cual está trabajando el osciloscopio para lograr la división horizontal seleccionada. Dicho valor es actualizado automáticamente al cambiar el H.DIV, al igual que la frecuencia máxima de los análisis espectrales.

El selector x8 permite hacer un zoom digital en el tiempo de la señal.

Canales

El selector de canales permite seleccionar los canales a ver en el display. Las opciones son:

- **Canal 1** - muestra solo el canal 1
- **Canal 2** - muestra solo el canal 2
- **Ambos (superpuestos)** - muestra ambos canales en el display superpuestos uno arriba del otro
- **Ambos (separados)** - muestra ambos canales en el display, pero separados: canal 1 (arriba), canal 2 (abajo)

Al cambiar esta opción se cambia la posición del cero en el display principal y también habilita (o deshabilita) los respectivos display de análisis espectral.

V.DIV

Los controles V.DIV permiten configurar configurar la división vertical del osciloscopio, es decir, el valor de voltaje que ocupa una división del display. En otras palabras, permite seleccionar la escala de voltaje a usar en cada canal. La misma puede seleccionarse de forma independiente por cada canal.

Nivel Trigger

Los controles de Nivel de Trigger permiten seleccionar el valor del trigger por nivel del osciloscopio. Es decir, el valor a partir del cual se comenzará a mostrar la señal en el display.

Este trigger es un disparador por software que, una vez recibidas las muestras del osciloscopio, decide a partir de qué muestra se visualizará en el display.

RT Step

El control RT Step permite cambiar el intervalo (step) de actualización del display en las capturas de tiempo real (es decir, para H.DIV = 1s, 2s ó 4s). Por defecto su valor es 1 y las señales son dibujadas pixel por pixel. Sin embargo, para PCs más lentas su velocidad puede ser insuficiente para observar un display continúo en tiempo real. Para dichos casos basta con aumentar el RT Step.

Parámetros de la señal

El osciloscopio presenta un cuadro con información sobre diversos parámetros típicos de la señal en la sección inferior derecha de la ventana. Ellos son:

- **Vpp** - voltaje pico-pico
- **Vrms** - voltaje eficaz
- **Vm** - voltaje medio

Estos valores son actualizados de forma continua, junto con la señal en el tiempo y su espectro.

Guardar muestras en archivo TSV

El programa permite guardar las muestras recibidas del osciloscopio en un archivo para su posterior análisis, por ejemplo en algún programa de cálculo numérico.

Para ello basta con tildar la opción **Guardar muestras en un archivo TSV** e ingresar debajo el nombre del archivo donde se desean guardar las muestras. Esto debe hacerse antes de pulsar el botón Capturar, puesto que esta opción no puede ser cambiada una vez que el osciloscopio está capturando.

El formato del archivo generado es bien simple: cada línea contiene dos valores separados por un tabulador. El primer valor es el voltaje del canal 1, y el segundo valor es el voltaje del canal 2.

Barra de estado

La barra de estado ubicada en la parte inferior izquierda de la ventana brinda información sobre el estado de funcionamiento del osciloscopio, por ejemplo, la cantidad de muestras capturadas y si hubo algún error al recibir algún dato.

Estado de conexión

La barra de estado de conexión es la que se encuentra en la esquina inferior derecha de la ventana y nos brinda información sobre el estado de conexión con el osciloscopio y su versión de firmware.

En caso de estar conectado aparecerá un indicador verde señalando que se encuentra conectado y la versión de firmware del osciloscopio. En caso de no estar conectado aparecerá el mismo en color rojo indicando **No conectado**.

Es posible desconectarse y volverse a conectar al osciloscopio tildando el cuadro que se encuentra a la izquierda del estado de conexión.

Capítulo 10. Temas pendientes y mejoras posibles

Contenido de este capítulo

Como ocurre en todo proyecto, tuvimos que dar prioridad prioridad a algunas tareas sobre otras debido (principalmente) a la falta de tiempo. Aquí presentamos una serie de temas que quedaron pendientes para implementar en una etapa posterior del proyecto, junto con la importancia de cada una.

Mejorar la etapa de entrada y aislación

Actualmente la etapa de entrada ha sido diseñada solamente para poder demostrar que éste control puede ser hecho, y que tanto el hardware, como el firmware y el software, están preparados para realizarlo.

Sin embargo no se ha realizado un exhaustivo diseño de esta etapa, y por lo tanto su funcionamiento puede considerarse muy elemental.

Los amplificadores de alta ganancia generan oscilaciones, y los de baja ganancia no amplifican lo suficiente para poder medir el rango de señales que se desea. Además, estos componentes tienen que cumplir con el requisito de ancho de banda necesario para no degradar el funcionamiento del equipo. A esta etapa se le debe dedicar una revisión y rediseño para cumplir con los requisitos necesarios.

La seguridad y la aislación de la señal de entrada hacia el equipo no ha sido realizada. Dado que hemos trabajado constantemente en un ambiente controlado, este tópico ha sido relegado. Queda pendiente entonces la realización de una aislación y adaptación de impedancias para preservar y proteger la seguridad del equipo y del operador.

Aislar el equipo del PC en el canal USB

La comunicación entre el PC y el osciloscopio se realiza de forma directa, con conexión directa entre uno y otro sin existir en el camino una protección o aislación, de modo de proteger a ambos equipos. Creemos este tema tan importante como el anterior, ya que no solo se trata de la seguridad, sino también de la posibilidad de que un equipo externo al PC lo dañe.

Mejorar el ancho de banda

Una de las primeras mejoras posible sería aumentar la frecuencia máxima de trabajo del osciloscopio para que fuese capaz de medir frecuencias mayores, de forma llegar a los 20 Mhz o incluso más. Este punto no concierne a la parte de la lógica por lo que no es necesario modificarla. Por la forma en que está diseñado la lógica es perfectamente escalable a mayores velocidades ya que, al no usarse el PIC directamente para el proceso de captura, la velocidad del mismo está solo limitada por los propios componentes involucrados (ADC, memoria y contador). Vale la pena recalcar este punto ya que pusimos especial cuidado en realizar un diseño que fuera escalable en cuanto a ancho de banda. Fue por esto que descartamos alternativas como la de usar un único chip con todo integrado. Por otro lado, tanto el ADC como la memoria seleccionados soportan velocidades de hasta 40 Mhz. Por lo tanto, lo único que sería necesario para lograr un prototipo que trabaje a mayores velocidades es utilizar un cristal de mayor frecuencia y mejorar la interferencia electromagnética para eliminar los problemas de ruido e interferencia que ocurren a dichas frecuencias.

Compatibilidad con puntas de atenuación

Es muy corriente que las puntas de medición para osciloscopios tengan incluida una función extra, la posibilidad de atenuar grandes señales de entrada. Esta característica esta diseñada para poder medir señales de gran amplitud, de este modo la misma punta de medición atenúa en 10 veces a la señal, para que ésta pueda cumplir con los rangos de entrada del osciloscopio. Esta es una práctica muy común y la mayoría de los osciloscopios y puntas de medición ya tienen esta capacidad. Queda por lo tanto como pendiente agregar en nuestro osciloscopio esta funcionalidad. En nuestro caso resulta simple, ya que se trata solamente de una modificación en el software, en donde se debe agregar la opción de seleccionar si se trata de una punta X1 o X10, y realizar las operaciones matemáticas correspondientes sobre los datos capturados. Estudiando varias hojas de datos de diferentes puntas de medición, observamos que todas las características y requisitos se establecen sobre una impedancia vista de entrada hacia el osciloscopio de 1 Mohm. Actualmente nuestro osciloscopio no tiene realizada completamente la etapa de entrada, por lo que este requisito no se cumple, sin embargo la intención es que la entrada sea 100% compatible y tenga características similares a la entrada de un osciloscopio comercial.

Mecanismo de conexión con el osciloscopio

Actualmente se utiliza el mecanismo de barrido de puertos serie para conectarse con el osciloscopio. Este mecanismo no es inocuo para el PC y por lo tanto debe cambiarse en el producto final, ya sea colocando un selector de puerto a utilizar (solución sencilla) o desarrollando un driver USB propio para el osciloscopio que no involucre el uso de puertos virtuales.

Interferencia electromagnética

Debido a la naturaleza de nuestro proyecto (componentes de alta velocidad, buses de datos, alta tasas de transferencias) se hace indispensable un delicado estudio pormenorizado de la interferencia electromagnética generada y recibida por los componentes de forma de poder diseñar una distribución óptima para minimizar dichos problemas.

Para atacar este problema de forma efectiva habría que comenzar por fabricar un circuito impreso de la placa teniendo en cuenta, entre otros, los siguientes puntos:

- evitar al máximo posible los loops en las pistas de alta frecuencia
- colocar componentes de alta frecuencia lo más cerca posible
- utilizar planos de tierra

Debido a que la fabricación de un circuito impreso para la placa no estaba dentro de los objetivos previstos, este fue uno de los tópicos que quedó pendiente para una futura mejora del prototipo.

Detección de transitorios por software

La detección de transitorios es una tarea muy importante, incluso más que el análisis en frecuencia, pero bastante más compleja también, puesto que involucra tareas de procesamiento inteligente, como ser correlación de los datos con secuencias de valores conocidos para detectar picos de voltaje y otros tipos de irregularidades. Esta tarea también fue relegada por no estar relacionada directamente con el diseño electrónico del aparato o su comunicación con el PC, sin embargo siempre tenida en cuenta a lo largo del desarrollo para dejarla prevista. Por eso escogimos una memoria grande de 32K para poder almacenar una buena cantidad de muestras, lo cual es un requisito necesario para este tipo de procesamiento de la señal. Dada su gran utilidad, consideramos que esta es una de las primeras características a agregar en el software.

Fabricar circuito impreso

Hemos estudiado las ventajas de utilizar un circuito impreso en vez de una placa universal. En el capítulo Fabricación de la placa analizamos las características de cada tipo de placa (Protoboard, PCB, Placa universal), y hemos planteado las ventajas y desventajas de cada una.

Una vez que se cuenta con el diseño final del equipo, es conveniente realizar un circuito impreso. De esta forma se podrá obtener mejores desempeños en cuanto a velocidad y ruido, y también menor probabilidad de fallas, malas conexiones, etc.

Driver USB propio

Escribir un driver de comunicación USB propio, lo cual permitiría obtener transferencia más rápida de los datos entre el osciloscopio y el PC llegando a la velocidad de transferencia de USB 1.1 (1.4 Mbytes/seg). Hoy en día la velocidad máxima de transferencia está limitada a 80 Kbytes/seg debido al firmware CDC utilizado para la comunicación.

Sin embargo, el tener un driver USB propio también complicaría la portabilidad, puesto que habría que escribir un driver para cada sistema operativo donde se quiera correr el osciloscopio, en lugar de aprovechar los drivers para la comunicación USB-CDC que ya vienen disponibles en todo sistema operativo moderno.

Utilizar un Vendor ID y Product ID propio

Si bien los valores del Product ID (PID) y Vendor ID (VID) puede ser modificados arbitrariamente por nosotros, dichos códigos son un tema muy delicado. La USB-IF (USB Implemmenters Forum, http://www.usb.org) tiene la autoridad (y responsabilidad) absoluta sobre la asignación de códigos VID únicos de 16 bit para cada fabricante que quiere comercializar sus productos USB. Esos códigos son obtenidos mediante una licencia y el pago de un impuesto (por una única vez) que oscila en los U$S 1500.

Una vez que al fabricante se la ha asignado un código VID tiene a disposición 65.536 código PID codes (de 16-bit) para identificar únicamente distintos modelos de su creación. En caso de querer comercializar el Osciloscopio USB sería necesario conectar a la USB-IF para solicitar nuestro propio VID.

Reducir el tamaño del firmware

Hoy en día el firmware utiliza funciones de la librería estándar de C stdio.h, concretamente `sprintf()`. Dichas funciones son extremadamente grandes y hacen que el tamaño final del firmware prácticamente se duplique. Como en nuestro caso el código entra sin problemas en el PIC no lo consideramos una prioridad pero se podría eliminar dichas funciones (que son muy genéricas) implementando manualmente rutinas que realicen la funcionalidad específica que nosotros precisamos.

Convolución con sinc

Actualmente el software interpola linealmente las muestras recibidas del osciloscopio para graficar la señal obtenida. Esto ocasiona que a frecuencias

muy cercanas a la máxima (4 Mhz) no sea posible vislumbrar una señal senoidal, más aún, solo se obtiene una banda continua de puntos, ya que en dicho caso se están interpolando únicamente dos puntos por período. Para solucionar este problema y lograr graficar señales senoidales de hasta 4 Mhz deberíamos entonces interpolar la señal con un sinc que es aquella señal cuya transformada en frecuencia es un escalón (es decir, un pasabajo), lo cual se asemeja más al filtro aplicado en el muestreo de la señal, y por lo tanto nos permite obtener un resultado más fiel al original.

Vale notar que al convolucionar un sinc con la señal más rápida (la cual tiene 2 muestras por período aprox.) se obtiene una señal senoidal perfecta lo cual es coherente con lo esperado puesto que una senoidal de 4 Mhz es la señal más rápida que el osciloscopio es capaz de medir.

Para realizar esta convolución con sinc se usaría la misma librería utilizada para la FFT (NumPy) y, debido a ello, sería una tarea bastante trivial que, a la vez aportaría una gran ventaja de usabilidad al osciloscopio.

LEDs de estado

Una característica importante de uso que dejamos de lado fue colocar LEDs de estado al osciloscopio, en particular:

- LED de encendido
- LED de conexión USB
- LED de captura de datos en curso

La primera (LED de encendido) iría conectada directamente a la salida del regulador 7805. La segunda (LED de conexión USB) iría conectada al pin Vcc del bus USB, mientras que la última (LED de captura de datos en curso) iría conectada a un pin del PIC y sería controlada por éste.

Capítulo 11. Autoevaluación y conclusiones

Contenido de este capítulo

Autoevaluación

Este capítulo esta sujeto a una redacción sumamente subjetiva sobre el camino recorrido. Una autoevaluación de cómo hemos procedido en estos 12 meses de trabajo, las lecciones aprendidas, conclusiones, e ideas de mejoras.

Si bien como primer etapa fue realizar un cronograma de trabajo sobre el cual basarse, no es lo mismo trabajar una semana con un cronograma de 7 días que trabajar 12 meses. Creemos que la realización y el seguimiento de un cronograma es *fundamental* para cumplir exitosamente el objetivo planteado. El beneficio en la organización y seguimiento del proyecto es sustancial, creemos que de plantearse objetivos intermedios , a corto plazo, es realmente necesario. No se puede plantear un objetivo de aquí a 12 meses sin proponer objetivos y plazos intermedios.

Por otra parte, la división y el cumplimiento de las tareas es imprescindible. Es necesario definir el encargado de cada tarea y el plazo que se tiene para ésta. Se trata de un trabajo de equipo, y esta claro que todas las partes del mismo deben funcionar en coordinación y de forma aceitada tal lo hacen los engranajes de un sistema mecánico. Si una de las partes no cumple con el objetivo o el plazo, todo el equipo se ve demorado y perjudicado. Nuevamente el mismo ejemplo, si un engranaje deja de funcionar, hasta que éste no cumpla su función, el sistema no podrá operar. La división de las tareas es un punto importante a tener en cuenta, ya que no solo una sola parte no podría realizar las tareas de todos, sino también que cada parte es más idónea en ciertas áreas que el resto del equipo puede no serlo. De esta forma, queda claro que es más productivo y eficaz cuando cada parte realiza tareas con las cuales se siente más cómodo, conoce más, y hasta puede realizarlas más rápido.

Este análisis es en cuanto al trabajo en equipo, sin embargo debemos destacar ciertos agentes externos que han hecho que se haya trabajado cómodamente. La necesidad de tener acceso a Internet creemos que hoy en día es sumamente necesaria. Se cuentan con muchas herramientas que facilitan el trabajo, como ser los diferentes buscadores y toda la documentación disponible en la red. Además, la herramienta de colaboración en linea que hemos utilizado (TWiki) hizo también posible que cada integrante pueda trabajar desde donde y cuando pueda, ya que siempre esta disponible toda la información, y ésta es actualizada a cada momento.

En cuanto al proyecto en sí, hemos aprendido muchas lecciones. Las dificultades que se tienen al trabajar en altas frecuencias las hemos comprobado. No es que no hayamos creído en ellas, todo lo contrario, sino que las hemos tenido en cuenta y sufrido. Pero no es lo mismo tomar precauciones para evitar estos inconvenientes sin haberlos sufrido nunca, que realmente se den en nuestra realidad. Un factor importante a tener en cuenta es el hecho de haber trabajado sobre una placa universal y no sobre un circuito impreso (PCB). El PCB se diseña para minimizar las posibles interferencias y ruidos, y realizar las conexiones más apropiadas entre componentes. Hay que tener recaudo de *"enjaular"* a todo el circuito para protegerlo aún más de las interferencias. Afortunadamente el circuito funciona correctamente a 8Mhz, pero esta claro que el objetivo es aumentar la frecuencia de trabajo para cumplir con las características técnicas propuestas. Creemos que al llegar a esta instancia será necesaria la fabricación de un circuito impreso justamente por el inconveniente de las altas frecuencias citado.

Otro ítem importante a tener en cuenta en la realización de este tipo de proyecto es la necesidad del equipamiento, y la disponibilidad del mismo. Creemos conveniente elaborar una lista de los instrumentos y herramientas necesarias para la ejecución del proyecto.

Un aprendizaje que nos ha dejado este trabajo, y que creemos muy importante, es el de prever con anticipación las necesidades. La disponibilidad de los componentes nos ha retrasado en el comienzo de la etapa de implementación y prueba. De haber tenido conciencia de esto, ordenar los componentes con anticipación nos hubiese dado una ventaja en los tiempos.

Estudio de rentabilidad

Finalmente, concluiremos con un análisis de los costos de fabricación y desarrollo para estudiar si el producto es rentable comercialmente. Para ello calcularemos el costo total de producir un osciloscopio con la siguiente fórmula:

Costo total = Costo materiales + Costo desarrollo

Donde costo total es la suma del costo de todos los componentes y el Costo de desarrollo viene dado por la siguiente fórmula:

Costo desarrollo = Costo hora hombre * Horas hombre dedicadas / Unidades a vender

Dado que nuestro mercado objetivo es el ámbito académico, estimaremos las unidades a vender basados en los potenciales clientes. A esto le sumaremos un estimativo de unidades que se podrán vender a entusiastas, empresas, personas que trabajen en el ramo, e incluso la posibilidad de vender unidades en el exterior (categorizados como *otros*).

Cliente	**Unidades a vender**
Universidad de la república	50

Universidad ORT	15
UTU	50
IEME	10
Universidad Católica	15
IADE	10
otros	30
Total	**180**

Tabla 11.1 Estimación de unidades a vender

Suponiendo el costo de la hora hombre de desarrollo en U$S 15, y estimando un total de 700 horas hombre dedicadas (2 * 350 hs) nos queda:

Costo desarrollo = 700 * 15 / 180 = **U$S 58**.

El costo de los componentes se desglosa a continuación. No se incluyen adaptadores SOIC-DIP y similares puesto que éstos solo serán necesarios para el prototipo de desarrollo. Además no se incluyen precios de resistencias y capacitores pero éstos a su vez son compensados por el abaratamiento de los componentes al comprarlos en grandes volúmenes y empaquetados más económicos.

Cantidad	Componente	Modelo	Precio (U$S)	Total
1	PIC	PIC18F4550	6.72	6.72
1	Cristal 8 Mhz		1	1
2	Memoria SRAM	CY7C109B-25VC	2.55	5.1
2	Contador 8-bit	74F269SPC	1.80	3.6
2	ADC	TLC5540	5.13	10.26
2	Buffer 3-state	74F245	2.3	4.6
2	Amplificador de entrada	MAX477	6.36	12.72
2	Conector BNC	-	1.55	3.1
1	Protector USB	SN65240P	0.99	0.99
1	Conector USB-B	ED90003-ND	1.23	1.23
			Total:	**U$S 49.32**

Tabla 11.2 Costo de los componentes

Finalmente, el costo total de fabricación del osciloscopio nos queda:

Costo total = Costo materiales + Costo desarrollo = U$S 49 + U$S 58 = **U$S 107**

Esto nos daría el margen suficiente como para comercializar el osciloscopio a **U$S 200**, lo cual nos parece un precio razonable para centros de estudio.

El margen de ganancia bruta que se obtiene quedaría disuelto en las siguientes categorías:

- reinversión (investigación, mejoras, etc).

- distribución.
- marketing, publicidad.
- ganancia neta.

Los porcentajes de cada ítem serán evaluados en el momento oportuno, y siempre que este proyecto llegue a ser una realidad comercial.

Es por las razones expuestas que podemos concluir que el proyecto es rentable.

Conclusión final

El proyecto que hemos desarrollado consiste en la construcción de un equipo que realiza funciones similares a un osciloscopio convencional y muestra los datos obtenidos en un PC. Tanto los controles de Time/Div, Volt/Div, etc., como los datos obtenidos, son transmitidos desde y al PC por medio de un puerto USB.

El equipo consta de dos canales y debería ser capaz de medir señales de hasta 20MHz.

Para ello desarrollamos un prototipo basado en un microcontrolador PIC 18F4550 el cual controla todo el funcionamiento del equipo. Si bien el prototipo fue diseñado para funcionar de acuerdo a las especificaciones, los ruidos generados con un oscilador de 48 MHz impidieron seguir trabajando a esta frecuencia y actualmente funciona con un oscilador de 8 MHz. Todo esto se encuentra más detallado en la sección de implementación (Ver: Problemas).

Hemos también desarrollado un software multiplataforma para mostrar gráficamente los datos obtenidos en un PC y controlar el equipo de acuerdo a las medidas que se deseen obtener.

Si bien las especificaciones completas del proyecto no podrán ser alcanzadas en esta instancia, creemos que el trabajo realizado exigió de las habilidades y conocimientos propios de un grupo de ingenieros, y logramos alcanzar un nivel muy aceptable permitiendo que en un breve periodo se logren los objetivos finales.

Por lo expuesto creemos que este proyecto ha sido satisfactorio en todo su desarrollo.

Así mismo, creemos en la capacidad de poder lograr los objetivos finales en una etapa posterior, y con vistas a su aplicación, utilización, y comercialización, en el mundo real.

Bibliografía

- ALAN DIX. 1999. Three DSP Tips. Disponible en Internet: <http://www.hiraeth.com/alan/papers/DSP99/DSP99.pdf>
- ARTUR KRUKOWSKI, IZZET KALE, RICHARD C. S. MORLING. 1996. Application of polyphase filters for bandpass sigma-delta analog-to-digital conversion. Disponible en Intenet: <http://users.cscs.wmin.ac.uk/~krukowa/pdf/Paper12.pdf>
- BITSCOPE DESIGNS. 2002. Bandwidth vs Sample Rate. Disponible en Internet: <http://www.bitscope.com/design/hardware/convertor/?p=3#sub>
- BONNIE C. BAKER. 2004. Predict the Repeatability of Your ADC to the BIT. Disponible en Internet: <http://ww1.microchip.com/downloads/en/devicedoc/adn010.pdf>
- BRAD BRANNON. 1995. Aperture Uncertainty and ADC System Performance. Disponible en Internet: <http://www.analog.com/UploadedFiles/Application_Notes/535694092954737395652273066884897736516373 4AN501.pdf>
- DALLAS SEMICONDUCTOR. 2002. How Quantization and Thermal Noise Determine an ADC's Effective Noise Figure. Disponible en Internet: <http://www.maxim-ic.com/appnotes.cfm/appnote_number/1197>
- DALLAS SEMICONDUCTOR. 2004. Coherent Sampling Calculator (CSC). Disponible en Internet: <http://www.maxim-ic.com/appnotes.cfm/appnote_number/3190>
- DAN STRASSBERG, 2003. Eyeing jitter: shaking out why signals shake. Disponible en Internet: <http://www.edn.com/article/CA293235.html>
- HEWLETT-PACKARD. 1998. Oscilloscopes Sampling: How fast and which way? Disponible en Internet: <http://tritium.fis.unb.br/Fis3Exp/www.tmo.hp.com/tmo/pia/BasicInstrument/TUTnBRIEF/English/BI-T-Sampling-010.html>
- MICROCHIP. 2001. Analog-to-Digital Converters. Disponible en Internet: <http://ww1.microchip.com/downloads/en/devicedoc/adc.pdf>
- P. P. VAIDYANATHAN. 2001. Generalizations of the sampling theorem: seven decades after nyquist, Disponible en Intenet: <http://www.systems.caltech.edu/dsp/students/bojan/journ/Nyquist7decades.pdf>
- PENTEK, INC. 2004. Putting Undersampling to Work. Disponible en Internet: <http://www.pentek.com/deliver/TechDoc.cfm/PutUndersamp.pdf?Filename=PutUndersamp.pdf>
- WIKIPEDIA. 2006. Nyquist–Shannon sampling theorem. Disponible en Internet: <http://www.answers.com/topic/nyquist-shannon-sampling-theorem>
- WIKIPEDIA. 2006. Quantization Noise. Disponible en Internet: <http://today.answers.com/topic/quantization-error>
- TEXAS INSTRUMENTS. 1995. Understanding Data Converters. Disponible en Internet: <http://focus.ti.com/lit/an/slaa013/slaa013.pdf>

- USB IMPLEMENTERS FORUM. 2000. USB 2.0 Specification. Disponible en Internet: <http://www.usb.org/developers/docs/>

Apéndice I: Esquemáticos

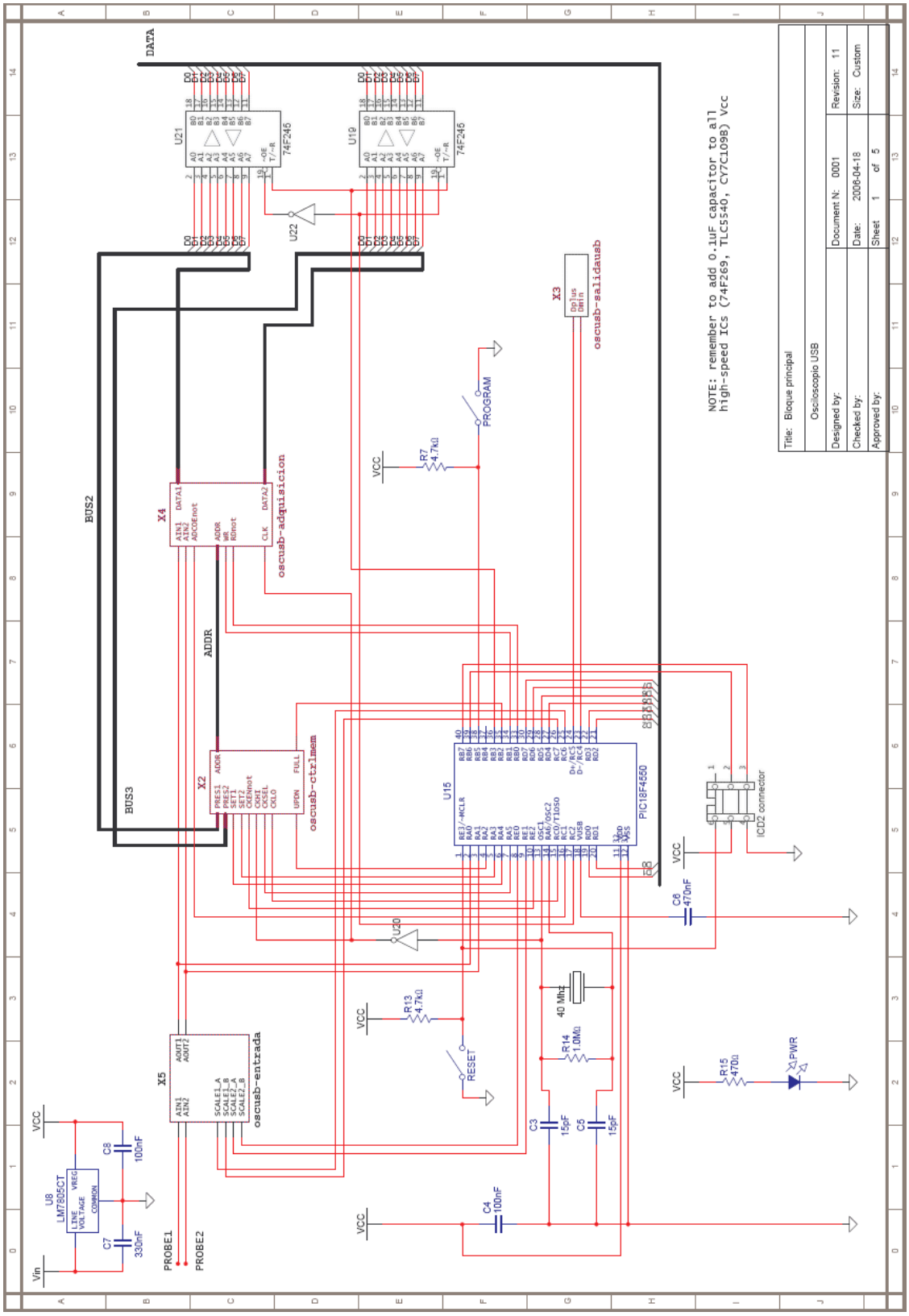
A continuación se publican los esquemáticos finales del osciloscopio, divididos por bloques funcionales.

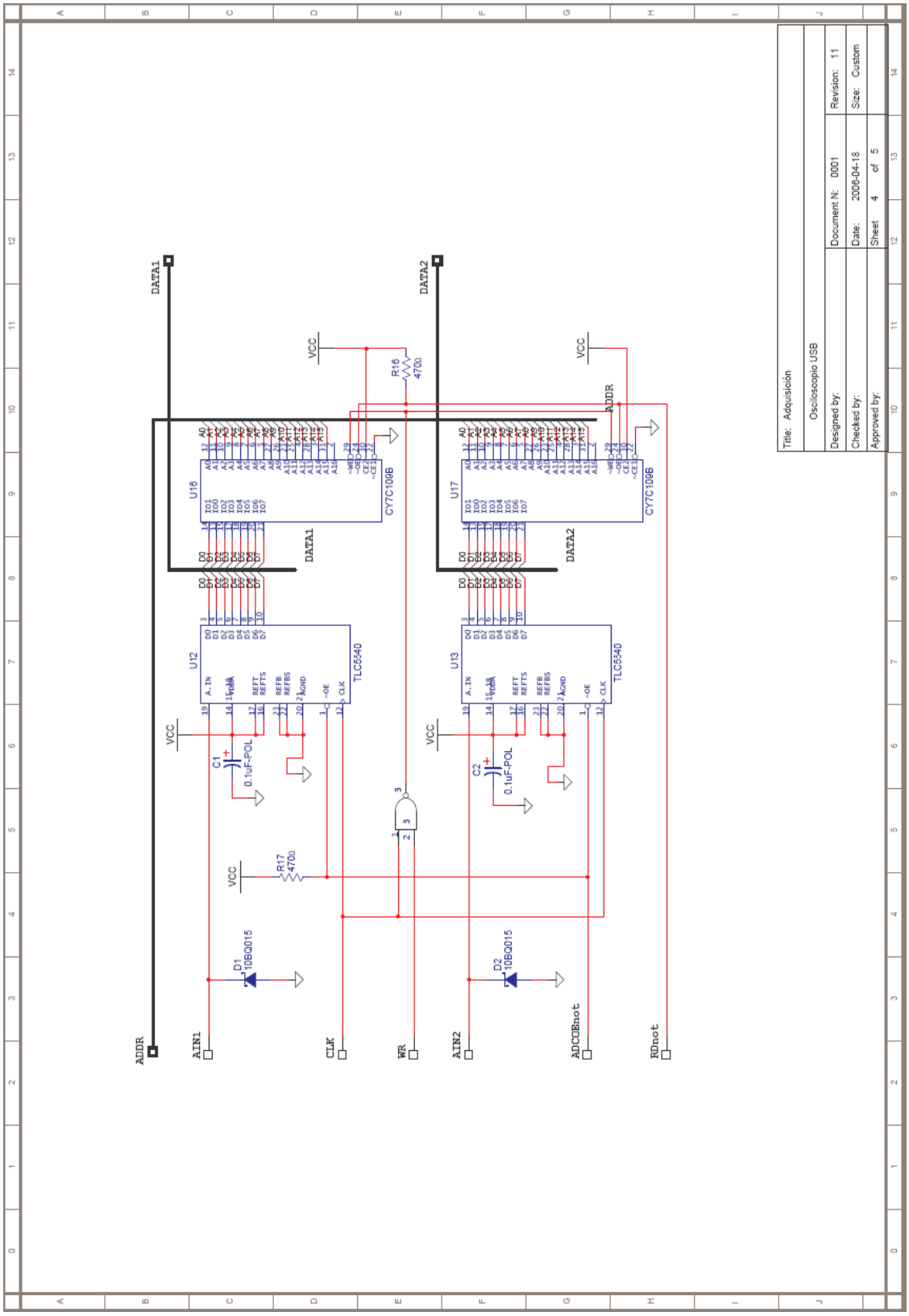
En total hubo 11 revisiones de los esquemáticos. Todas ellas se encuentran disponibles en la siguiente página:

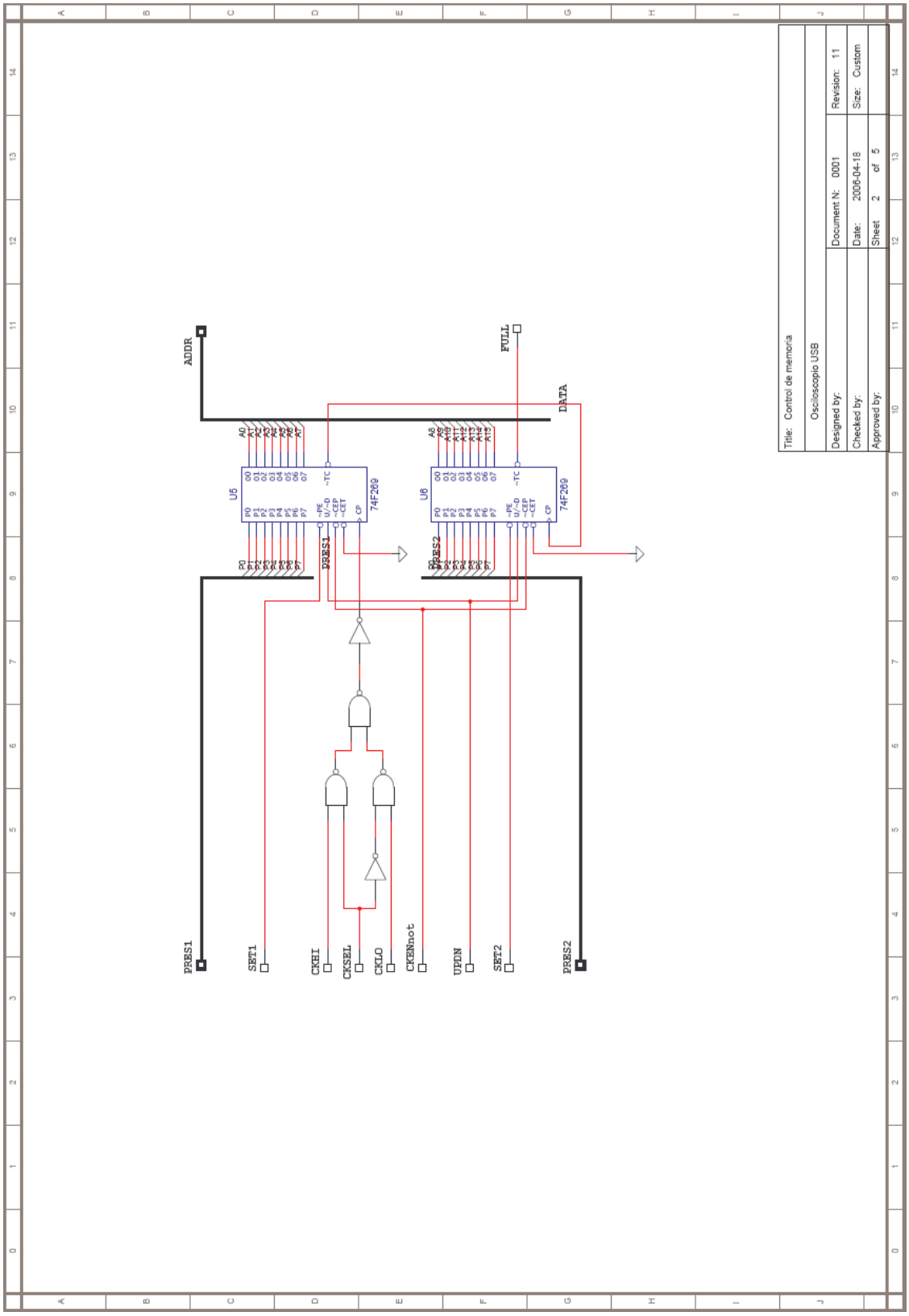
http://pablohoffman.com/oscusb/esquematicos/

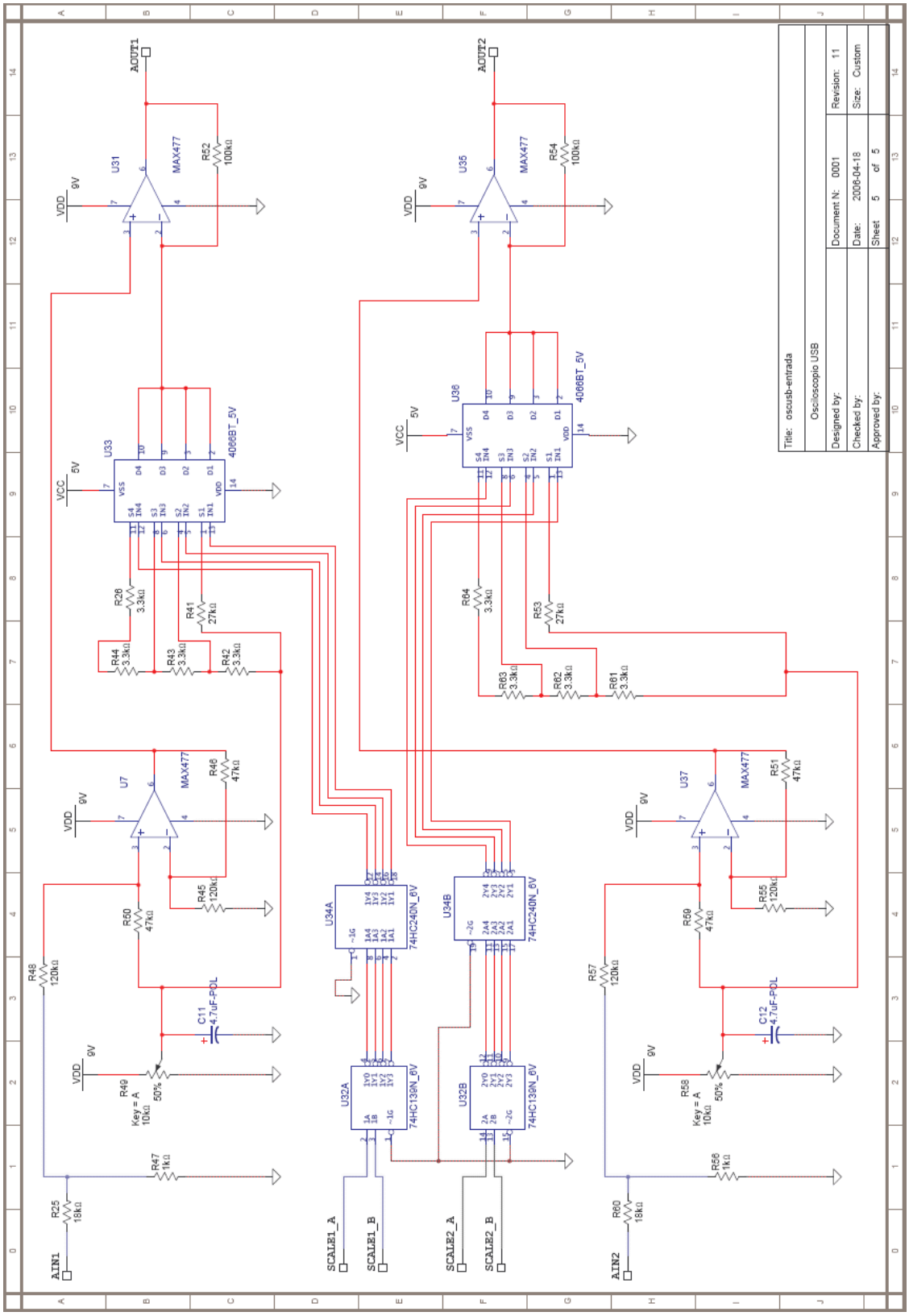
Allí se puede observar el progreso de los mismos, puesto que cada revisión tiene anotaciones de los cambios con respecto a la versión anterior.

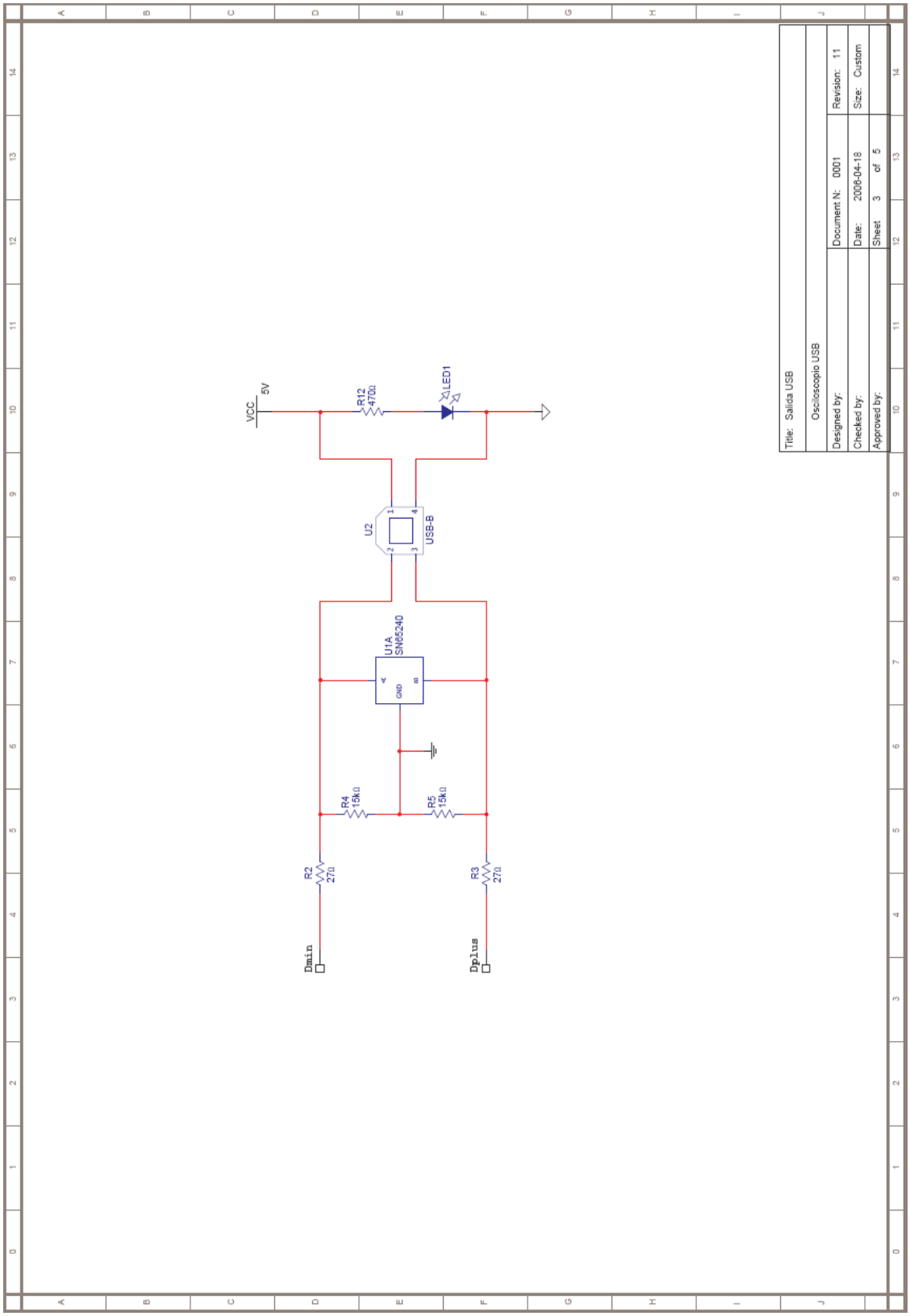
A continuación se publica la última revisión de los mismos.


DATA
U21
74F245
U19
74F245
U22
X3
oscuab-salidauab
NOTE: remember to add 0.1uF capacitor to all
high-speed ICs (74F269, TLC5540, CY7C1098) Vcc
Title: Bloque principal
Oscilloscopio USB
Designed by:
Checked by:
Approved by:
Document N: 0001
Revision: 11
Date: 2006-04-18
Size: Custom
Sheet 1 of 5
BUS2
X4
oscuab-adquisicion
PROGRAM
R7
4.7kΩ
VCC
ADDR
BUS3
X2
oscuab-ctrlmem
U15
PIC18F4550
ICD2 connector
C6
470nF
U20
40 MHz
R13
4.7kΩ
RESET
R14
1.0MΩ
C3
15pF
C5
15pF
R15
470Ω
PWR
X5
oscuab-entrada
C4
100nF
U8
LM7805CT
C8
100nF
C7
330nF
Vin
PROBE1
PROBE2


Title: Adquisición
Osciloscopio USB
Designed by:
Checked by:
Approved by:
Document N: 0001
Date: 2006-04-18
Sheet 4 of 5
Revision: 11
Size: Custom
DATA1
DATA2
ADDR
VCC
R16
470Ω
R17
470Ω
U16
U17
U12
U13
CY7C109B
TLC5540
C1
C2
0.1uF-POL
D1
D2
10BQ015
AIN1
AIN2
CLK
WR
ADCOEnot
RDnot


ADDR
FULL
DATA
U5
U6
74F200
PRBS1
PRBS2
SET1
CKEI
CKSEL
CKLO
CKENnot
UPDN
SET2
PRBS2
Title: Control de memoria
Osciloscopio USB
Designed by:
Checked by:
Approved by:
Document N: 0001
Date: 2006-04-18
Sheet 2 of 5
Revision: 11
Size: Custom


Title: oscusb-entrada
Osciloscopio USB
Designed by:
Checked by:
Approved by:
Document N: 0001
Date: 2006-04-18
Sheet 5 of 5
Revision: 11
Size: Custom
AOUT1
AOUT2
AIN1
AIN2
VDD
VCC
MAX477
4066BT_5V
74HC240N_6V
74HC138N_6V
SCALE1_A
SCALE1_B
SCALE2_A
SCALE2_B


VCC
5V
R12
470Ω
LED1
U2
USB-B
U1A
SN65240
GND
R4
15kΩ
R5
15kΩ
R2
27Ω
R3
27Ω
Dmin
Dplus
Title: Salida USB
Oscilloscopio USB
Designed by:
Checked by:
Approved by:
Document N: 0001
Date: 2006-04-18
Sheet 3 of 5
Revision: 11
Size: Custom

Apéndice II: Lista de materiales (BOM)

BOM (Bill of materials)

Esta es la lista de todos los componentes necesarios, con el código de referencia en el que aparecen en los esquemáticos. Fue generada con el Multisim a partir de la última revisión de los esquemáticos.

Qty.	Description	RefDes	Package
1	74HC_6V, 74HC139N_6V	U32	Generic\NO16
1	74hc240, 74HC240N_6V	U34	Generic\NO20
2	CAP_ELECTROLIT, 4.7uF-POL	C11, C12	Generic\ELKO5R5
4	Opamps, MAX477	U7, U31, U35, U37	Generic\DIP-8
2	CMOS_5V, 4066BT_5V	U33, U36	Generic\SO-14
2	ADC_DAC, TLC5540	U12, U13	Generic\DIP-24
2	SCHOTTKY_DIODE, 10BQ015	D1, D2	Generic\SMB
2	CAP_ELECTROLIT, 0.1uF-POL	C1, C2	Generic\ELKO5R5
2	Memory, CY7C109B	U16, U17	Ultiboard\DIP32
4	RESISTOR, 470Ohm_5%	R16, R17, R12, R15	Generic\RES0.25
2	RESISTOR, 15kOhm_5%	R4, R5	Generic\RES0.25
1	Transient, SN65240	U1	Generic\DIP-8
2	LED, LED_green	LED1, PWR	Generic\LED1
2	RESISTOR, 27Ohm_5%	R2, R3	Generic\RES0.25
1	Connectors, USB-B	U2	Ultiboard\SO4
2	Counters, 74F269	U5, U6	Generic\PDIP-24
2	RESISTOR, 4.7kOhm_5%	R7, R13	Generic\RES0.25
2	CAPACITOR, 100nF	C4, C8	Generic\CAP4
1	Microprocessors, PIC18F4550	U15	Generic\PDIP-40
2	SWITCH, DIPSW1	J1, J2	Ultiboard\DIPSW1H
1	RESISTOR, 1.0MOhm_5%	R14	Generic\RES0.25

```
2       CAPACITOR, 15pF                 C3, C5
Generic\CAP1
1       CAPACITOR, 470nF                C6
Generic\CAP4
2       Buffers, 74F245                 U19, U21
Generic\DIP-20
1       SOCKETS, DIP6                   ICSP
Generic\DIP-6
1       VOLTAGE_REGULATOR, LM7805CT     U8
Generic\TO-220
1       CAPACITOR, 330nF                C7
Generic\CAP4
--------------------------------------------------------------------------
-----------------
```

Apéndice III: Hojas de datos

Contenido de este apéndice

A continuación se publican los links a las hojas de datos utilizadas por los componenetes mencionados en la documentación, ya se que se utilizen en el osciloscopio o no. La lista se separa en secciones segun la categoría del componente.

Procesadores

- PIC18F4550
  - http://ww1.microchip.com/downloads/en/DeviceDoc/39632b.pdf
- ETRAX 100LX
  - http://developer.axis.com/products/etrax100lx/18354_etrax_lx.pdf

Contadores

- Fairchild 74F269
  - http://fairchildsemi.com/ds/74/74F269.pdf

Memorias

- Texas BQ4011
  - http://www-s.ti.com/sc/ds/bq4011.pdf
- Cypress CY7C199
  - http://www.cypress.com/portal/server.pt/gateway/PTARGS_0_2_1524_209_259_43/http%3B/sjapp20.mis.cypress.com%3B7001/publishedcontent/publish/design_resources/datasheets/contents/cy7c199_5.pdf
- Cypress CY7C109B
  - http://www.cypress.com/portal/server.pt/gateway/PTARGS_0_2_1524_209_259_43/http%3B/sjapp20.mis.cypress.com%3B7001/publishedcontent/publish/design_resources/datasheets/contents/cy7c109b_5.pdf
- ALSC AS7C256A
  - http://www.alsc.com/pdf/sram.pdf/fa/as7c256av.1.2.pdf

ADCs

- Texas ADS831
  - http://www-s.ti.com/sc/ds/ads831.pdf
- Texas TLC5540
  - http://www-s.ti.com/sc/ds/tlc5540.pdf
- Texas THS0842
  - http://www-s.ti.com/sc/ds/ths0842.pdf
- Analog Devices AD9057
  - http://www.analog.com/en/prod/0%2C%2CAD9057%2C00.html
- Analog Devices AD9059
  - http://www.analog.com/en/prod/0%2C%2CAD9059%2C00.html

Buffers 8-bit

- Texas SN74AHCT541
  - http://www-s.ti.com/sc/ds/sn74ahct541.pdf
- Fairchild 74F245
  - http://fairchildsemi.com/ds/74/74F245.pdf

Protectores USB

- Texas SN65220
  - http://www-s.ti.com/sc/ds/sn65220.pdf

Amplificadores

- Maxim MAX477
  - http://pdfserv.maxim-ic.com/en/ds/MAX477.pdf
- Texas OPA695
  - http://www-s.ti.com/sc/ds/opa695.pdf
- Texas OPA691
  - http://www-s.ti.com/sc/ds/opa691.pdf
- Texas OPA830
  - http://www-s.ti.com/sc/ds/opa830.pdf

Osciladores programables

- Linear Technology LTC6905
  - http://www.jlab.org/~segal/junk/LTC6905.pdf

Apéndice IV: Glosario

Glosario

A continuación se presentan los términos, abreviaciones y siglas utilizadas comunmente en la jerga del proyecto.

- AB - Ancho de Banda
- ADC - Analogue to Digital Converter
- API - Application Progam Interface. Conjunto de funciones provistas por una librería para extender la funcionalidad de un lenguaje de programación para una funcionalidad particular.
- Bitscope - osciloscopio para PC de origen australiano con esquemáticos del hardware publicados en la web.
- CRO - Cathode Ray Oscilloscope
- DFT - Discrete Fourier Transform
- DSO - Digital Storage Oscilloscope. También es el nombre del software para controlar el Bitscope.
- IC - Integrated Controller.
- DMA - Direct Memory Access. Sistema utilizado por dispositivos para acceder directamente a la memoria, sin pasar por los puertos de entrada/salida del procesador. Permite transferencias de alta velocidad.
- Endpoint - entidad lógica que usan los dispositivos para comunicarse por el protocolo USB
- Flash - tecnología utilizada por algunos osciloscopios digitales para digitalizar la señal.
- FPGA - Field-programmable gate array. Más información en http://en.wikipedia.org/wiki/FPGA
- HDL - Hardware Description Language
- PCB - Printer Circuit Board
- PGA - Programmable Gain Amplifier
- Pipe - En USB, es el canal de comunicación de conecta al host controlador con el endpoint.
- Pipeline - tecnología utilizada por algunos osciloscopios digitales para digitalizar la señal. Ver TecnologiasADC
- SAR - Successive Aproximation Register. Un arquitectura muy común para de ADC de mediana a alta frecuencia.
- Schottky (diodo) - El diodo Schottky es un diodo con bajo voltaje de umbral y alta velocidad de conmutación. Mas info en http://en.wikipedia.org/wiki/Schottky_diode
- SMD - Surface Mount Devices. Ver SMT.
- SMT - Surface Mount Technology. La tecnología de ensamblar integrados "sin agujeros", es decir soldado directamente a la placa. Mas info en http://en.wikipedia.org/wiki/SMD
- TLC - Linea de convertidores Analógico-digitales de Texas Instruments.
- TLC5540 - Convertidor ADC que usa el Bitscope. Ver InformacionBitscope.
- Toolkit gráfico - API utilzada para el control, manejo y creación de una aplicación gráfica.

Apéndice V: Especificaciones

A continuación se presenta una tabla con el resumen de las especificaciones del osciloscopio, según el prototipo final armado. Se adjunta información sobre el factor o componente limitante de cada valor.

Especificación	Valor	Componente que limita/define
Resolución vertical	8 bits	ADCs y memorias
Conexión a la PC	USB 1.1 (USB 2.0 compatible)	El PIC solo trabaja hasta USB 1.1
Ancho de banda analógico	40 Mhz	Amplificadores de entrada (MAX477)
Velocidad de muestreo máxima	Canal único: 4 MS/s Canal doble: 4 MS/s	Cristal 8 Mhz
Entradas	2 canales con entradas BNC	
Rangos de tensión	±40 V	Etapa de entrada
Precisión de tensión	3%	Error del ADC y etapa de entrada (estimado)
Tamaño del buffer	64K	Memorias
Impedancia de entrada	1 MΩ	Amplificadores de entrada (MAX477)
Alimentación eléctrica	9V DC	Regulador 7805
Consumo de potencia	7W	Componentes activos (ver cálculo en capítulo 4)
Dimensiones	200mm x 150mm x 80mm	Carcaza
Software	Windows 98/2000/XP Linux Mac (no probado)	Lenguaje python

Apéndice VI: Lista de figuras

Apéndice VII: Lista de tablas