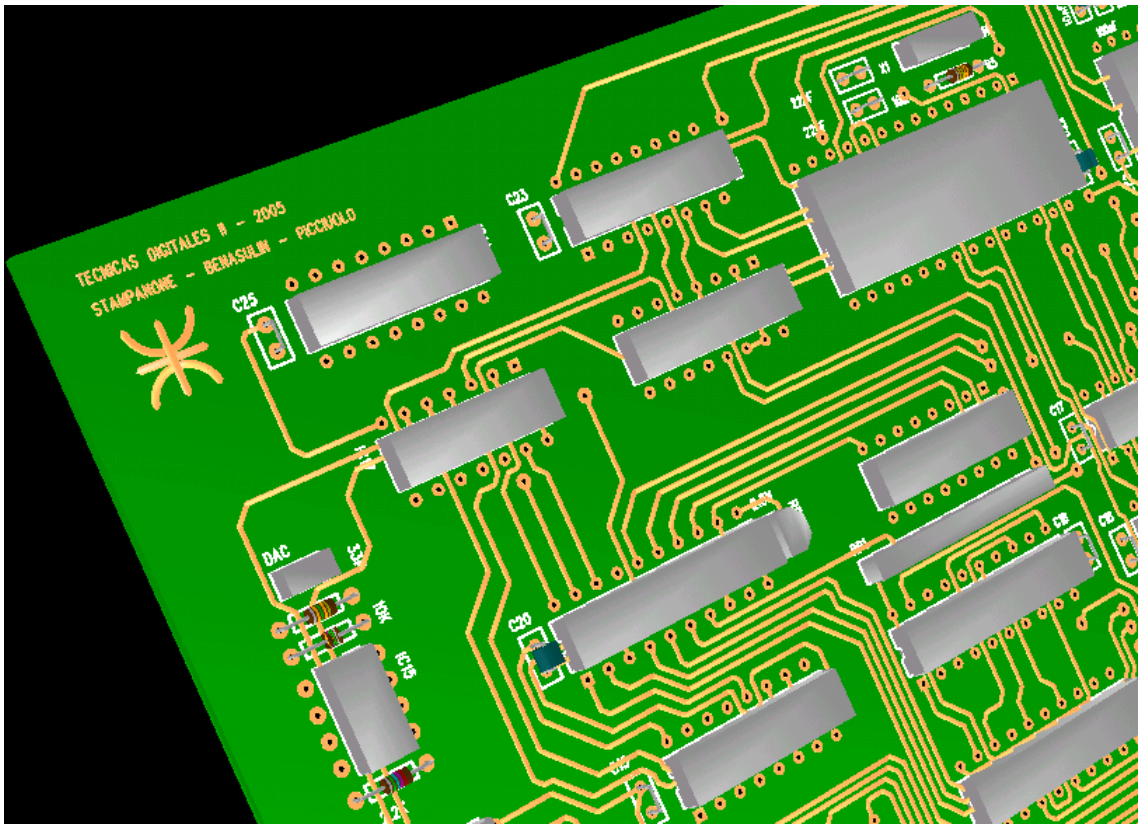
PLACA DE ADQUISICIÓN DE DATOS
12 BITS



UNIVERSIDAD TECNOLOGICA NACIONAL

FACULTAD REGIONAL CORDOBA

TECNICAS DIGITALES II

BENASULIN, DIMAS 45590
STAMPANONE, LEANDRO 44512
PICCIUOLO, FABRICIO 42426

AÑO 2005

Indice

1 Introducción:

A continuación se detalla cada módulo conformante de una placa de adquisición de datos a través del BUS ISA. Si bien es cierto que la Arquitectura ISA (Arquitectura Estándar de la Industria) tiende a desaparecer debido a su escasa velocidad de transferencia (Frecuencia de Clock en 8MHz), todavía sigue siendo útil para pequeños y medianos diseños debido a su sencillez de implementación.

Actualmente todos los PC que se comercializan cuentan con dos o mas ranuras de expansión PCI, los cuales cuentan con una frecuencia de Clock de 60MHz y capacidad de Plug & Play, lo cual exige la presencia de un "controlador dedicado" con capacidad de almacenar datos del fabricante y opciones de configuración, este último no se puede diseñar con microcontroladores ya que prácticamente ningún microcontrolador trabaja a una frecuencia de Bus interno de 60MHz, por lo cual se requiere de dispositivos mas veloces como FPGA´s o bien un controlador de BUS PCI(Interfaz de Componentes Periféricos) específico.

1.1 Características Generales:

La Placa de Adquisición Desarrollada integra los siguientes componentes:
Interfaz Periferica Programable tipo 82C55, con capacidad de 24 puertos I/O configurables.
Conversor Digital – Analogico con 8 bits de resolución
Módulo Conversor Analogico – Digital 12 bits de resolución, Multiplexor de 8 canales.
Módulo Selector de Dirección Base.
Selector de Interrupción.

1.2 El BUS ISA:

En 1981 IBM lanza al mercado su primera computadora personal que contaba con un transporte de datos de 8 bits conocido como XT, que funcionaba a la misma velocidad que el procesador. Estos equipos estaban potenciados con procesadores de la familia 8086/8088, que utilizaba e l transporte de datos XT con un canal de 8 bits para datos a una frecuencia de 4.77 MHz.

El ancho de banda de este bus con el procesador 8088 formaba un equipo perfecto, pero la ampliación del canal de datos en el 8086 a 16 bits se vio limitada por la capacidad de este transporte , ya que por primera vez en la historia de la computación moderna, aparecieron los famosos cuellos de botella. Dada la evolución de los procesadores, para ese entonces, el bus de la computadora no era la solución para una comunicación fluida con el exterior del procesador. En definitiva no podía hablarse de un alto desempeño en un PC cuando esta sólo tenía un ancho de ruta de datos de 8 bits y el procesador trabajando a 16 .

En 1984, IBM lanza al mercado el PC/AT con el procesador 80286 para el que empleó el bus AT, que era una marca registrada de IBM.

El transporte AT, como originalmente fue llamado por IBM, fue documentado por primera vez en una publicación llamada *PC-AT Technical Reference*. Esta referencia técnica incluía esquemas y listados del BIOS, lo que hacía fácil para otras empresas, como Compaq, producir clones compatibles con IBM, mismos que no podían utilizar el término Bus AT, por ser una marca registrada de IBM, razón por la cual comenzó a llamarse Arquitectura Estándar de la Industria , mejor conocido por las siglas de su nombre en inglés: Industry Standard Arquitecture o ISA , ampliando el bus XT a 16 bits y 8 MHz, logrando una capacidad de transferencia de 16 MB/s. En esa primera versión de la especificación no se incluyó información esencial como tiempos, reglas y otros requerimientos que pudieron hacerlo una buena especificación, lo cual llevó a que eventualmente se aparecieran diversas especificaciones del bus ISA, que usualmente no eran compatibles entre sí.

Leandro Stampanone-44512
Benasulin, Dimas -45590
Picciuolo Fabricio -42426

La Ranura de Expansión del BUS ISA esta compuesta de la siguiente manera:

Bus de Direcciones completo demultiplexado (A19-A0) para el sistema 8088 de 1MB de capacidad de direccionamiento

El Bus de Datos (D7-D0) de 8 bits.

Las cuatro señales de control $\overline{MEMR}$, $\overline{MEMW}$, $\overline{IOR}$ **e** $\overline{IOW}$ para el control de la E/S y de cualquier memoria que podría estar ubicada en la tarjeta de circuito impreso.

IRQ2-IRQ7: Línea de **Solicitud de Interrupción.**

DRQ1-DRQ3: Entradas de **Solicitud de DMA.**

$\overline{DACK0}$ - $\overline{DACK3}$: Salidas de **Reconocimiento de DMA.**

Tensiones de Alimentación: 5V, 12V, -5V, 12V.

Clock de 14.3118 MHz y 8 MHz.

LADO DE COMPONENTES	
GND	$\overline{\text{IO CHK}}$
RESET	D7
+5	D6
IRQ9	D5
-5	D4
DRQ2	D3
-12	D2
$\overline{\text{CARD SLCTD}}$	D1
+12	D0
GND	$\overline{\text{IO RDY}}$
$\overline{\text{MEMW}}$	AEN
$\overline{\text{MEMR}}$	A19
$\overline{\text{IOW}}$	A18
$\overline{\text{IOR}}$	A17
$\overline{\text{DACK3}}$	A16
DRQ3	A15
$\overline{\text{DACK1}}$	A14
DRQ1	A13
$\overline{\text{DACK0}}$	A12
CLOCK	A11
IRQ7	A10
IRQ6	A9
IRQ5	A8
IRQ4	A7
IRQ3	A6
$\overline{\text{DACK2}}$	A5
TC	A4
ALE	A3
+5	A2
14.3118MHz	A1
GND	A0

BUS DE DATOS

BUS DE DIRECCIONES

1.3 El BUS ISA de 16 bits:

Con la aparición del 8086 con su BUS de datos de 16 Bits el BUS ISA había quedado chico, aparece entonces un segundo conector en la ranura de expansión que incorpora entre otras señales las 8 líneas de datos para completar un Bus de Datos de 16 Bits.

A continuación se detalla el conexionado del mismo:

$\overline{\text{MCS16}}$	SBHE
$\overline{\text{IOCS16}}$	A23
IRQ10	A22
IRQ11	A21
IRQ12	A20
IRQ15	A19
IRQ14	A18
$\overline{\text{DACK0}}$	A17
DRQ0	$\overline{\text{MEMR}}$
$\overline{\text{DACK5}}$	$\overline{\text{MEMW}}$
DRQ5	D8
$\overline{\text{DACK6}}$	D9
DRQ6	D10
$\overline{\text{DACK7}}$	D11
DRQ7	D12
+5	D13
$\overline{\text{MASTER}}$	D14
GND	D15

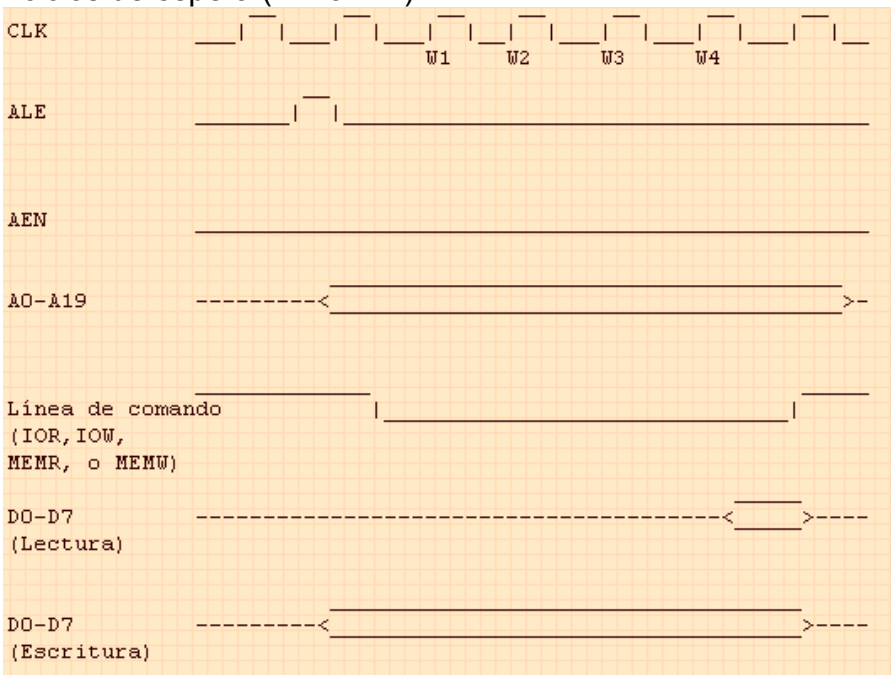
1.4 Descripción de los terminales:

SEÑAL	**DESCRIPCION**
A0-A19	Bits de dirección 0-19, permiten direccionar 1Mb de memoria y 64K de puertos de e/s.
A17-A23	Bits de dirección 17-23, permiten direccionar desde 256Kb de memoria a 16Mb.
AEN	Address Enable; Cuando está activa el controlador DMA posee el control de las líneas de dirección y del BUS de datos, conforme se indique en MEMR/MEMW. Cuando está inactiva la CPU tiene el control de estas líneas.
ALE	Address Latch Enable (salida); se emplea para que la CPU esté aislada de las líneas de dirección (triestado). Es forzado activado durante los ciclos DMA.
CARD SLCTD	Card Selected; indica que una tarjeta ha sido activada en el slot XT de 8 bits.
CLK	Señal de reloj del sistema (actual velocidad del BUS).
D0-D7	Bits de datos 0-7 para e/s a memoria o puertos de e/s.
DACK0-DACK3	Reconocimiento DMA para los canales 0 al 3; empleada por el controlador para reconocer una petición DMA (validación de acceso DMA). DACK0 es empleada para el refresco de memoria (MREF).
DRQ0-	Petición DMA 0-3; empleada por periféricos que desean los servicios del controlador

DRQ3	DMA; Se mantiene activa hasta que la correspondiente señal DACKx se hace activa.
I/O CH CHK	I/O Channel Check; Genera una interrupción no enmascarable.
I/O CH RDY	I/O Channel Ready; es puesta inactiva por memoria o dispositivos de e/s para retardar el acceso a memoria o los ciclos de e/s. Normalmente es empleada por dispositivos lentos para añadir estados de espera. No debe ser inactiva durante más de 17 ciclos.
I/O CS16	I/O Chip Select 16 Bit; indica ciclo de e/s de 16 bits
IOR	I/O Read; indica a un dispositivo de e/s que coloque su dato en el BUS del sistema.
IOW	I/O Write; indica a un dispositivo de e/s a leer un dato del BUS del sistema.
IRQ2-IRQ7	Petición de interrupción 2-7; indica a la CPU que un dispositivo de e/s necesita servicio.
MASTER	Empleado por DRQ para ganar el control del sistema.
MEM CS16	Memory Chip Select 16 bit; indica ciclo de memoria de 16 bits.
MEMR	Memory Read; esta señal es producida por la CPU o el controlador DMA e indica a la memoria que debe introducir el dato direccionado en el BUS del sistema. Presente tanto en el BUS PC como en la extensión AT.
MEMW	Memory Write; esta señal es producida por la CPU o el controlador DMA e indica a la memoria que debe leer y almacenar el dato presente en el BUS. Presente tanto en el BUS PC como en la extensión AT.
OSC	Oscilador; Señal de reloj de 14.31818 MHZ (periodo de 70ns); 50% del ciclo de servicio.
RESET DRV	Reset Drive; empleada para resetear la lógica del sistema.
SBHE	System BUS High Enable; activa los bits de datos 8-15 de la extensión AT del BUS.
TC	Terminal Count; produce un impulso cuando la cuenta final de un canal DMA es alcanzado.

1.5 Diagrama de tiempos de acceso a memoria o e/s de 8 bits:

Se muestran 4 ciclos de espera (W1 a W4):



ALE se pone a nivel lógico alto (1) y la dirección aparece en A0 a A19. El dispositivo esclavo debe leer la dirección durante el flanco de bajada de ALE, y la dirección en A0 a A19 permanece válida hasta el final del ciclo de transferencia. Notar que AEN permanece a nivel bajo durante todo el ciclo de transferencia.

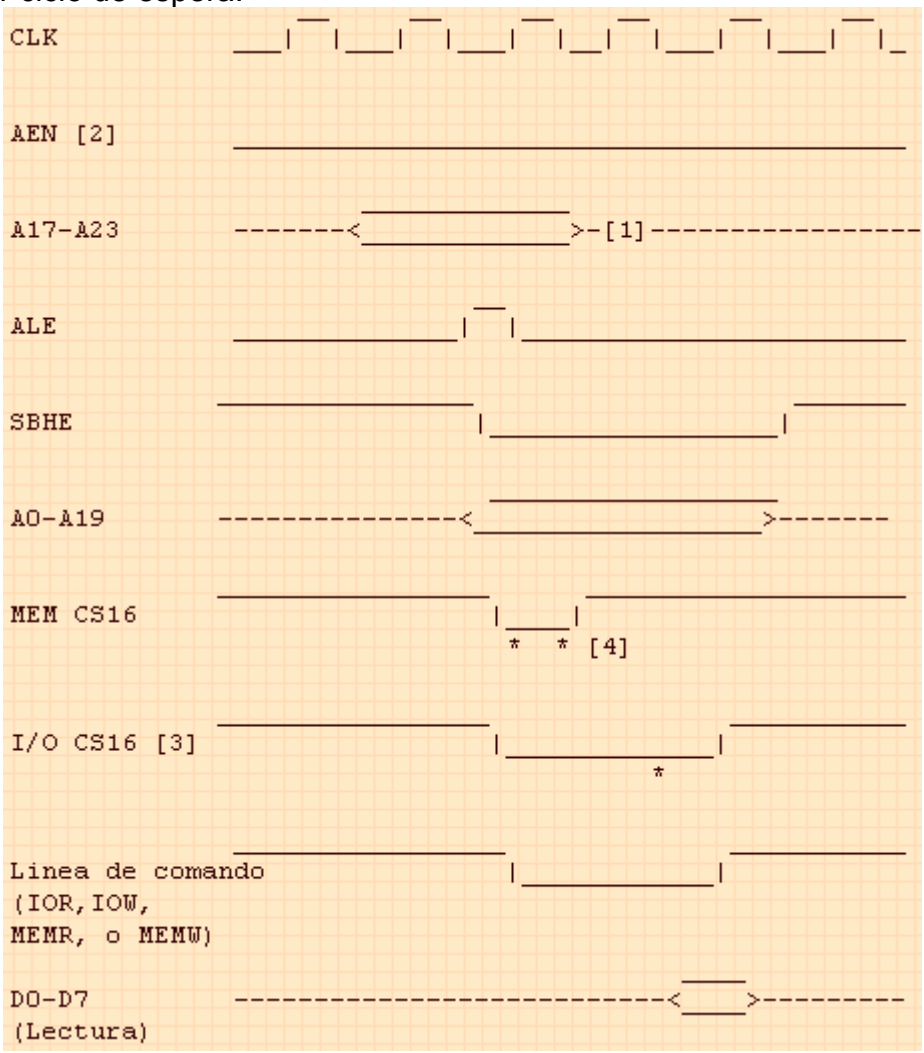
La línea de comando es puesta a nivel bajo (IOR o IOW para e/s, MEMR p MEMW para memoria, lectura y escritura respectivamente). Para operaciones de escritura, los datos permanecen en D0 a D7 hasta el resto del ciclo de transferencia. Para operaciones de lectura, los datos deben ser válidos en el flanco de bajada del último ciclo.

CARD SLCTD se emplea en la mitad de cada ciclo de espera. Si está a nivel bajo, el ciclo de transferencia termina sin más ciclos de espera. I/O CHRDY se emplea en la primera mitad de cada ciclo de espera. Si está a nivel bajo, más ciclos de espera serán introducidos.

Por defecto el ciclo de transferencia de 8 bits posee 4 ciclos de espera. La mayoría de las BIOS del ordenador pueden cambiar el número de ciclos de espera.

1.6 Diagrama de tiempos de acceso a memoria o e/s de 16 bits:

Se muestra 1 ciclo de espera:



Un asterisco (*) indica el punto donde la señal es tomada.

[1] La porción de direccionamiento del bus de extensión de 16 bits para el siguiente ciclo puede ser puesto ahora en el bus. Esto se emplea para que las tarjetas puedan comenzar a decodificar la dirección más rápidamente. Para ello el este tipo de acceso debe estar activado (pipeline).

[2] AEN se mantiene bajo durante todo el ciclo de transferencia, indicando que un ciclo normal (no DMA) está produciéndose.

[3] Algunos controladores de bus presentan esta señal durante el mismo ciclo de reloj que MEM CS16, en vez de durante el primer ciclo de espera, como se muestra en el diagrama. En este caso, I/O CS16 necesita ser puesto a nivel bajo tan pronto como la dirección ha sido decodificada, lo cual sucede antes que la activación de las líneas de comando.

[4] MEM CS16 es tomada una segunda vez, en caso que el adaptador no active la señal a tiempo durante la primera vez (normalmente debido a que el dispositivo no está monitorizando el bus de 16 bits para tomar el direccionamiento rápido, o está esperando al flanco de bajada de la señal ALE).

Las transferencias de 16 bits siguen los mismos tiempos básicos que las transferencias de 8 bits.

Un direccionamiento válido debe aparecer en el bus de extensión de 16 bits antes del comienzo del ciclo de transferencia, De lo contrario el bus extendido de 16 bits no es direccionado, y no es válido para el resto del ciclo de transferencia (en la mayoría de los ordenadores). El bus extendido de 16 bits debería ser direccionado en el flanco de bajada de ALE. Hay que mencionar que en algunos sistemas, el bus extendido de 16 bits sigue los mismos tiempos que el bus de 8 bits. En ambos sistemas, una dirección válida debe estar presente en el bus en el flanco de bajada de ALE.

Las tarjetas de expansión de e/s no necesitan monitorizar el bus extendido de 16 bits o ALE, ya que el espacio de direccionamiento de e/s siempre está dentro del rango del bus de 8 bits.

SBHE será puesta a nivel bajo por la placa base, y la tarjeta de expansión debe responder con I/O CS16 o MEM CS16 en el momento apropiado, o realizar dos transferencias separadas de 8 bits.

Muchos sistemas esperan a I/O CS16 o MEM CS16 antes que las líneas de comandos sean válidas. Esto requiere que I/O CS16 o MEM CS16 sean puestas a nivel bajo tan pronto como la dirección sea decodificada (antes que se sepa si el ciclo es de e/s o memoria). Si el sistema comienza un ciclo de memoria, ignorará I/O CS16 (y viceversa para ciclos de e/s con MEM CS16).

Para operaciones de lectura, los datos son tomados en el flanco de subida del último ciclo de reloj.

Para operaciones de escritura, los datos válidos aparecen en el bus antes del final del ciclo, como es mostrado en el diagrama de tiempos. Mientras que el diagrama indica que los datos necesitan ser tomados en el flanco de subida, en la mayoría de los sistemas permanecen válidos durante todo el ciclo de reloj.

Para transferencias de 16 bits se toma por defecto un tiempo de espera de 1 ciclo de reloj. Esto puede ser acortado o alargado de la misma forma que las transferencias de 8 bits, mediante las señales CARD SLCTD y I/O CHRDY. Mucho sistemas solo permiten dispositivos de memoria de 16 bits (y no dispositivos de e/s) para transferir empleando 0 ciclos de espera (CARD SLCTD no tiene efecto en los ciclos de e/s de 16 bits).

Las señales MEMR/MEMW presentes en el bus de 16 bits siguen los mismos tiempos que las presentes en el bus de 8 bits cuando el direccionamiento está dentro del primer megabyte de memoria. Si el direccionamiento es mayor (por encima del primer megabyte), las señales MEMR/MEMW del bus de 8 bits permanecen a nivel alto durante el resto del ciclo.

2 Diseño de la Interfaz al BUS ISA

Básicamente nuestra tarjeta deberá disponer de un módulo direccionador, el cual permite establecer las direcciones de cada uno de los submódulos integrantes de la placa y además deberá contar con Buffers que adapten electricamente nuestro sistema al BUS.

Una vez conocidas las señales del BUS nos disponemos a analizar e implementar la circuiteria básica para establecer la conexión de nuestro sistema al BUS. Esta disposición es válida para cualquier sistema.

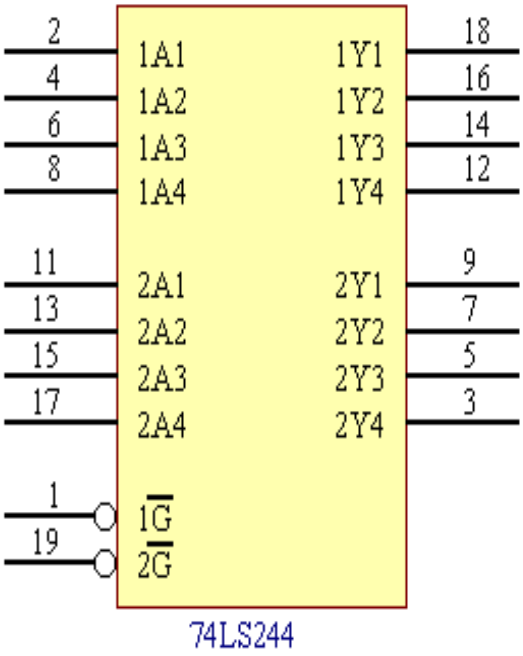
2.1 Carga Unitaria al BUS

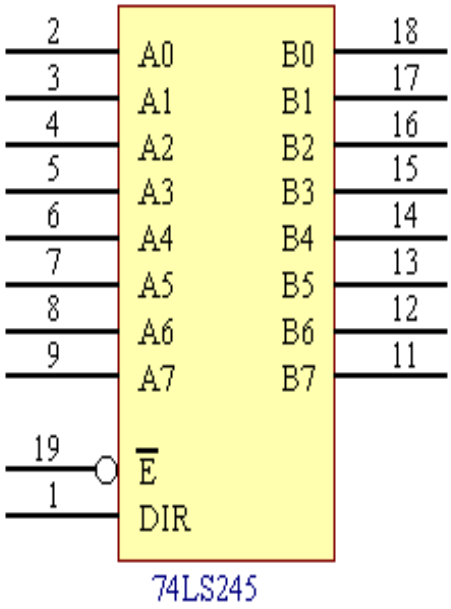
En primer lugar debemos asegurarnos que la carga hacia el BUS ISA se mantenga en una carga TTL baja (LS), esto es debido a que seguramente hay mas de una tarjeta conectada a las ranuras de expansión, si cada tarjeta no representa una carga unitaria se puede exceder el fan-out del controlador del BUS. Empleamos en nuestro circuito, un **buffer (Aislador) Tri-State 74LS244**

para reducir la carga al **BUS de Direcciones**. El buffer 74LS244 es unidireccional, por tal motivo es empleado para las señales del Bus de Address y las señales de Control.

Para el Bus de Datos empleamos un Buffer Bidireccional 74LS245 con capacidad de Tercer – Estado o Estado de Alta Impedancia.

Siempre que se trabaje con sistemas basados en Buses de comunicaciones es necesario emplear dispositivos con capacidad de tercer-estado o de alta impedancia.





Inputs		Output
$\overline{G}$	A	Y
L	L	L
L	H	H
H	X	Z

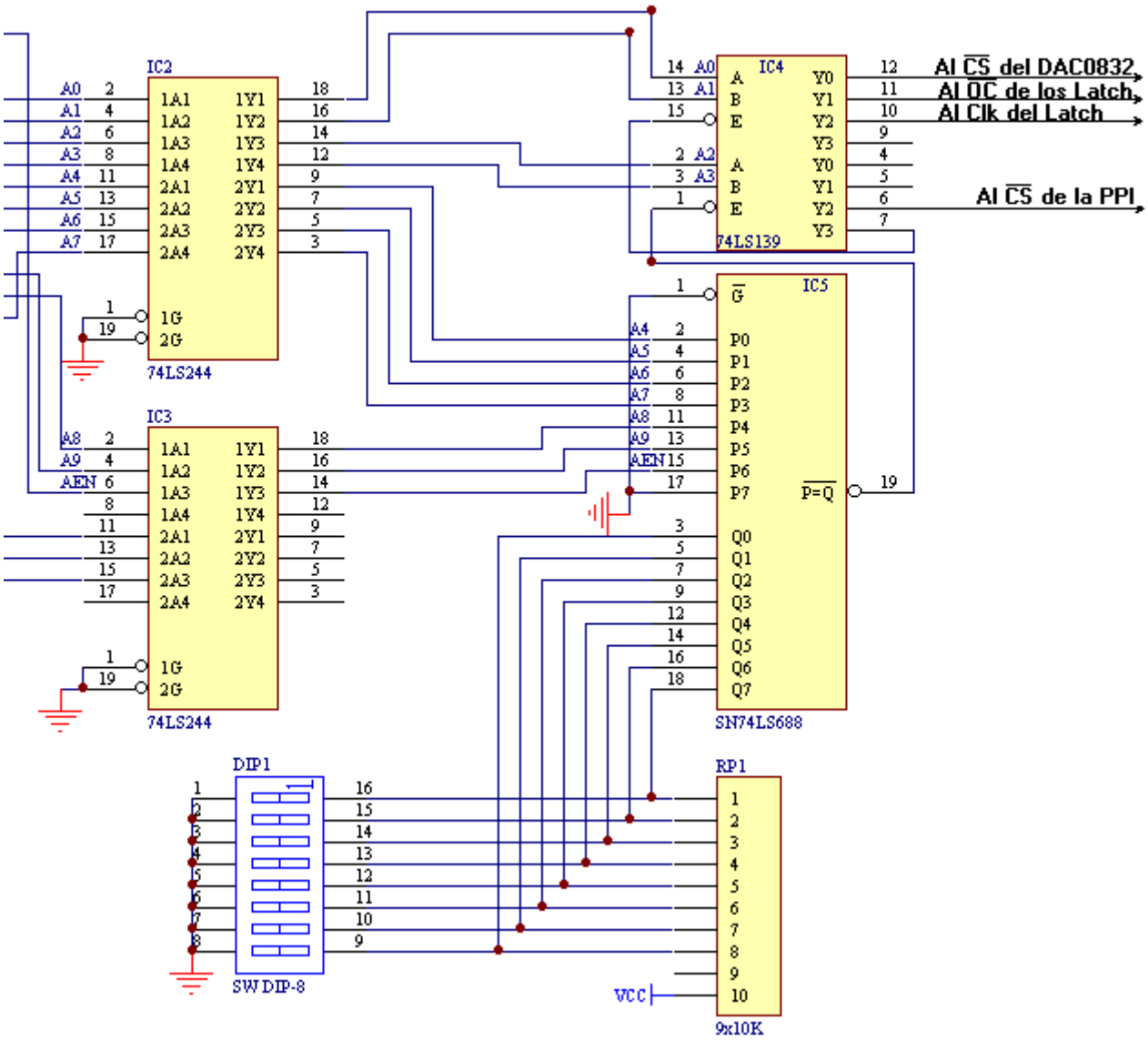
L = Low Logic Level
H = High Logic Level
X = Either Low or High Logic Level
Z = High Impedance

Enable $\overline{G}$	Direction Control DIR	Operation
L	L	B data to A bus
L	H	A data to B bus
H	X	Isolation

H = High Level, L = Low Level, X = Irrelevant

2.2 Direccionamiento de la Tarjeta:

En la computadora personal, el bus ISA está diseñado para operar en las direcciones desde E/S 0000H hasta 03FFH. Ahora bien, debemos tener en cuenta que nuestra placa no necesariamente debe ser la única conectada al BUS, por lo tanto debemos incorporarle la capacidad de cambiar su dirección base con el fin de evitar conflictos con otras posibles placas que estén en el sistema.
Para poder seleccionar la dirección base incorporamos un **Comparador de Magnitud de 8 bits 74LS688**.
A continuación detallamos la sección de direccionamiento y la conexión de los buffers:



Mediante el DIP-SWITCH seleccionamos la dirección base de nuestra placa, teniendo en cuenta que AEN (Address Enable) ingresa al comparador, el mismo debe ser comparado con cero, ya que cuando AEN esta en alto indica que el Controlador de DMA (Direct Memory Access) tiene el control del BUS, por lo tanto la tarjeta debe estar desactivada.

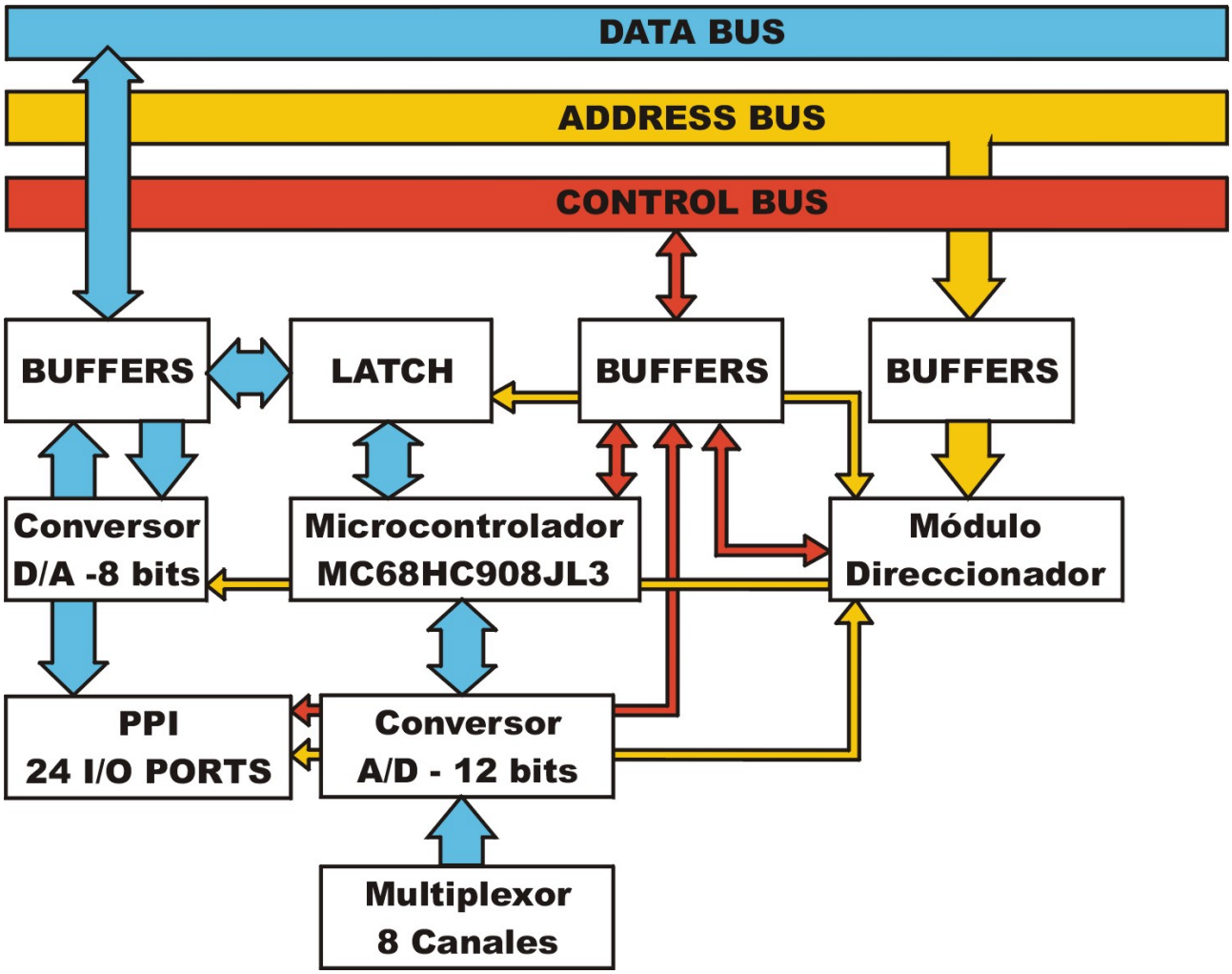
De acuerdo al esquema vemos que la placa puede ser ubicada desde 0000h – 03f8h inicialmente seleccionamos la dirección 300h como dirección base de nuestra placa.

3 Diseño Modular de la Placa de Adquisición:

Nuestra placa cuenta con la posibilidad de configurar 24 puertos como entrada o salida mediante una Interfaz Periferica Programable, seleccionar uno de ocho canales para capturar señales analogicas con una resolución de 1 mV y establecer un valor de tensión en función de un valor numerico establecido desde el programa principal. Cada una de estas funciones es realizada por un módulo específico, de aquí que placa de adquisición responde a un diseño modular, donde cada uno de los modulos tiene establecida su dirección de acceso.

A continuación examinamos un diagrama en bloques que nos ilustrará al respecto:

3.1 Diagrama en bloques de la Placa de Adquisición de Datos:



3.2 Mapeo de los dispositivos:

Siguiendo el esquema de conexionado obtenemos el siguientes mapa de direcciones:

A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Hexa	Dispositivo
1	1	0	0	0	0	1	0	0	0	308h	PPI Puerto A
1	1	0	0	0	0	1	0	0	1	309h	PPI Puerto B
1	1	0	0	0	0	1	0	1	0	30Ah	PPI Puerto C
1	1	0	0	0	0	1	0	1	1	30Bh	Control
1	1	0	0	0	0	1	1	0	0	30Ch	DAC 0832
1	1	0	0	0	0	1	1	0	1	30Dh	µC Entrada
1	1	0	0	0	0	1	1	1	0	30Eh	µC Salida Baja
1	1	0	0	0	0	1	1	1	1	30Fh	µC Salida Alta

3.3 Interfaz Periferica Programable 82C55:

Como anticipamos en la introducción, la placa cuenta con 3 puertos paralelos de 8 bits configurables, que son implementados mediante una PPI tipo 8255A.

De las hojas de datos de este dispositivo, podemos decir que es un dispositivo de entrada y salida de datos con 24 pines que pueden ser programados individualmente en 2 grupos de 12, y utilizados en 3 modos de operación.

En Modo 0, cada grupo de 12 pines, puede ser programado como I/O en subgrupos de 4.

En Modo 1, cada grupo puede ser programado para tener 8 lineas de entrada o sadida. De los 4 pines restantes, 3 son usados para handshaking y para la señal de control de interrupciones.

En Modo 2, es un modo de bus bidireccional, que usa 8 lineas para un bus bidireccional, requiriendo una linea del otro grupo para Handshaking.

Para mayor información remitirse a la Hoja de Datos del Dispositivo.

3.4 Módulo Conversor Digital – Analogico:

La placa cuenta con un DAC de 8 bits (256 datos) triestate, tipo DAC0832, compatible con µp para este procedimiento.

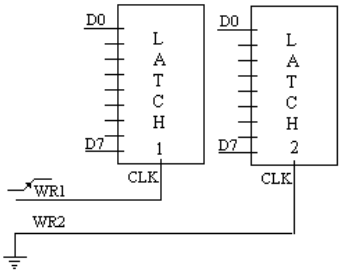
En un DAC común, hay que mantener el dato sobre el bus mientras realiza la converión, pero el DAC0832 tiene dos buffers internos; puede latchear dos datos (2 byets), propiedad útil tambien para cuando se usan DACS en paralelo, permitiendo la actualización de los mismos en forma simultánea.

Puede trabajar en 3 modos, a saber:

1) El dato en la entrada es automáticamente convertido.
2) El dato se almacena en el 1º latch solamente.
3) El dato va pasando por los latchs correlativamente.

Los latchs funcionan con CLK´s independientes, que se conectan en WR1 y WR2 respectivamente.

En nuestra placa:



Una de las características de este dispositivo, es que dispone de las señales (*CS, *RD y *WR) necesarias para una conexión directa a un sistema basado en microprocesador.

Para mayor información remitirse a la hoja de datos provista por el fabricante

3.5 Módulo Conversor Analogico – Digital:

¿Por qué usar un microcontrolador en un Sistema basado en Microprocasador?

Para responder esta pregunta debemos tener en cuenta que el conversor A/D empleado no dispone de las señales necesarias para conectarse a un Bus de Datos, ya que su modo de intercambiar datos es Serial Síncrono mientras que el Bus del sistema es Paralelo, es decir, todo lo opuesto.

Aquí es donde el uso de un microcontrolador cobra importancia ya que no solo permitirá vincular un sistema serial con uno paralelo sino también establecer tiempos regulares de conversion, esto ultimo es muy importante ya que si las muestras fueron obtenidas en tiempos regulares, luego será posible reconstruir la señal capturada.

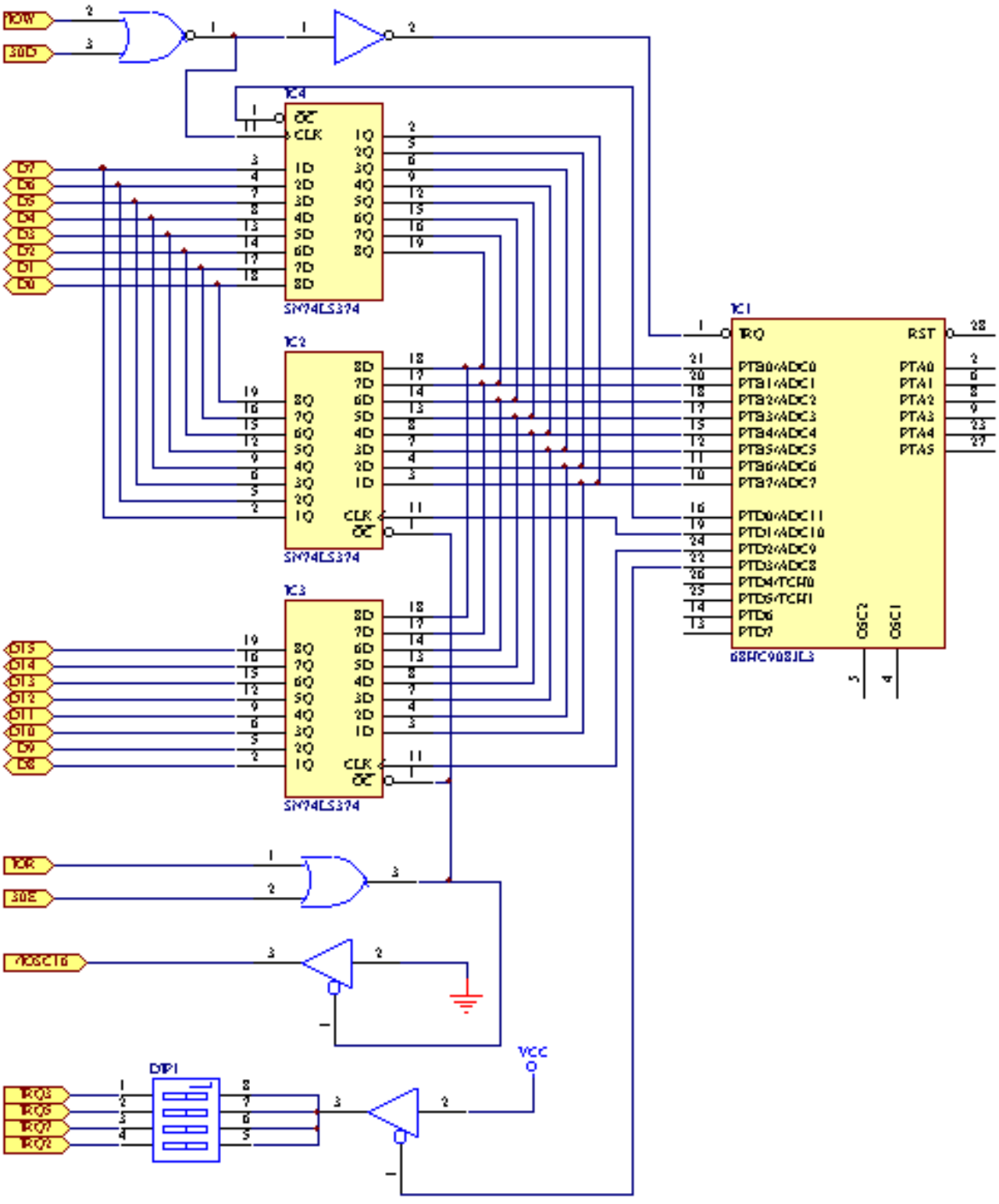
Los intervalos de conversion pueden ser seleccionados por el ususario a través del programa principal, configurando el Timer interno del microcontrolador.

3.5.1 Conexión del microcontrolador al BUS ISA:

Anteriormente vimos que cuando trabajábamos con sistemas basados en “buses de datos” era necesario emplear dispositivos con capacidad de Tri-State o estado de “desconexión” esto se debe a que varios dispositivos comparten el BUS y si dos o mas dispositivos configuran sus puertos como salida se puede producir un “bus contention”, que eléctricamente representaría un cortocircuito. Al emplear dispositivos Tri-State nos aseguramos que solo un dispositivo por ves sea configurado con sus puertos como salida, sin embargo puede haber varios dispositivos configurados como entradas siempre y cuando la cantidad de los mismos no supere el fan-out del dispositivo que enviaba los datos.

En particular, el microcontrolador MC68HC908JL3 de la firma Motorola no posee puertos con capacidad de Tercer Estado, esta falencia impediría conectar el MCU directamente al BUS de Datos, además habría problemas con el direccionamiento del mismo ya que el programa interno del micro debería chequear constantemente un puerto que este conectado al decodificador de direcciones, esto puede traer problemas con los tiempos ya que el MCU es mas lento que el BUS, entonces para evitar todo tipo de problemas referentes a los tiempos de acceso y el direccionamiento se emplearon tres latch tipo 74LS374 de manera de “aislar” del BUS al Micro.

Para enviar datos desde la PC al microcontrolador se utiliza uno de los latch antes mencionados, este esta mapeado en la dirección 30Dh. Cuando en el BUS de direcciones aparece esta dirección y el bit R/W está en cero, a través de una compuerta NOR, se activa la entrada de interrupción externa (IRQ) del micro, indicándole de esta forma que el dato esta listo para ser leído. El micro activa la salida del latch (pin OE) para acceder a los datos. Para enviar datos a la PC el micro utiliza los dos latch restantes almacenando en uno de ellos el byte mas significativo y en el otro el byte menos significativo, para lograr esto envía la señal de clock a los latch, de a uno por vez, almacenando de esta forma la información, luego el microcontrolador envía una señal de interrupción a la PC, a través del bus ISA, para indicar que la información esta lista para ser leída, la IRQ a utilizar se puede configurar a través de un dip-switch. La PC lee la información a través de los 16bits de datos del BUS ISA habilitando las salidas de los dos latch simultáneamente, con esta misma señal activa el pin MEM/IOSC16 del ISA necesario para la transferencia de datos de 16bits. A continuación se detalla el módulo MCU – Latch – BUS.



3.5.2 ADC12138

El ADC usado es de la firma National Semiconductor, un ADC12138, conversor analógico Digital de 12 Bits más signo de aproximaciones sucesivas.

3.5.3 Funciones de los terminales del ADC12138:

A continuacion brindamos una lista de los pines del ADC12138, sus funciones, y la conexión en la placa.

CCLK: Reloj del conversor de aproximaciones sucesivas, los flancos no deben exceder 1μs. En nuestra placa, este reloj depende directamente del reloj de salida del Bus ISA, solo que se a intercalado un Divisor de frecuencia, el 74HC4017, ya que la frecuencia de salida del Bus ISA, es de 14,3118 MHz, y la frecuencia máxima calculada para el ADC es de 5 MHz. El 4017, divide por 3 la frecuencia de entrada, con lo que conseguimos una frecuencia en el ADC de 4,77 MHz.

SCLK: Reloj de la comunicación. En el flanco de bajada, salen los datos por DO, excepto el 1º bit de la plabara de salida que sale con el flanco de subida del EOC (cuando el CS esta continuamente en 0´). Cuando el CS cambia, en el flanco de bajada sale el primer bit. El CS puede llevarse a 0´ cuando el SCLK esta en 0´.
Los flancos de este reloj no deben exceder de 1μs.

DI: Entrada de datos serial, los datos son desplazados hacia el ADC con el flanco de subida del SCLK.

DO: Salida de datos serie. Cuando CS esta en 1´esta salida esta en TRISTATE. Los datos salen con el flanco de bajada del SCLK.

EOC: Cuando está en 0´indica que el ADC está trabajando. El flanco de subida del EOC indica el fin del ciclo, sea este autozero, conversión, autocalibracion, etc.

CS: Cuando está en 0´se pueden sacar los datos de la conversión, cuando esta continuamente en 0´el 1º bit del dato sale con el flanco de subida del EOC. Si cambia, en el flanco de bajada sale el 1º bit del dato. El flanco de subida del CS, resetea la conversión actual. Si se mantiene continuamente en bajo, hay que tener la precaucion de mandar la cantidad justa de cloks, para no dessincronizar la comunicación. Cuando se alimenta el ADC, por defecto, la cantidad de cloks son 13. (No la utilizamos en nuestra placa)

DOR: Pin de dato de salida listo, indica con 0´que los datos estan saliendo, y con 1´que todos los datos salieron. (No la utilizamos en la placa)

CONV: Tiene que estar en 0´para poder configurar el ADC. Cuando esta en 1´ pone al ADC en modo de lectura solamente. (No la utilizamos en la placa).

PD: Pin de apagado, cuando esta en 1´el ADC está apagado, cuando está en 0´el ADC esta encendido, despues de encenderlo el ADC necesita 700 μs para estabilizarse. (A GND en nuestra placa).

CH0-CH7: Entradas analógicas.

COM: Entrada analógica, se usa como una pseudo masa cuando se usa en modo simple.

MUXOUT1 / 2: Son las salidas del multiplexor interno del ADC.

A/DIN1 / 2: Son las entradas al conversor propiamente dicho, En nuestro circuito se conecta a MUXOUT1 con ADIN1 y MUXOUT2 con ADIN2

V_{REF}^+: Entrada de referencia positiva. $V_{REF}=V_{REF}^+-V_{REF}^-$ = $1V_{DC}$, para 5 o V_{DC}. V_{REF}+ no debe exceder VI

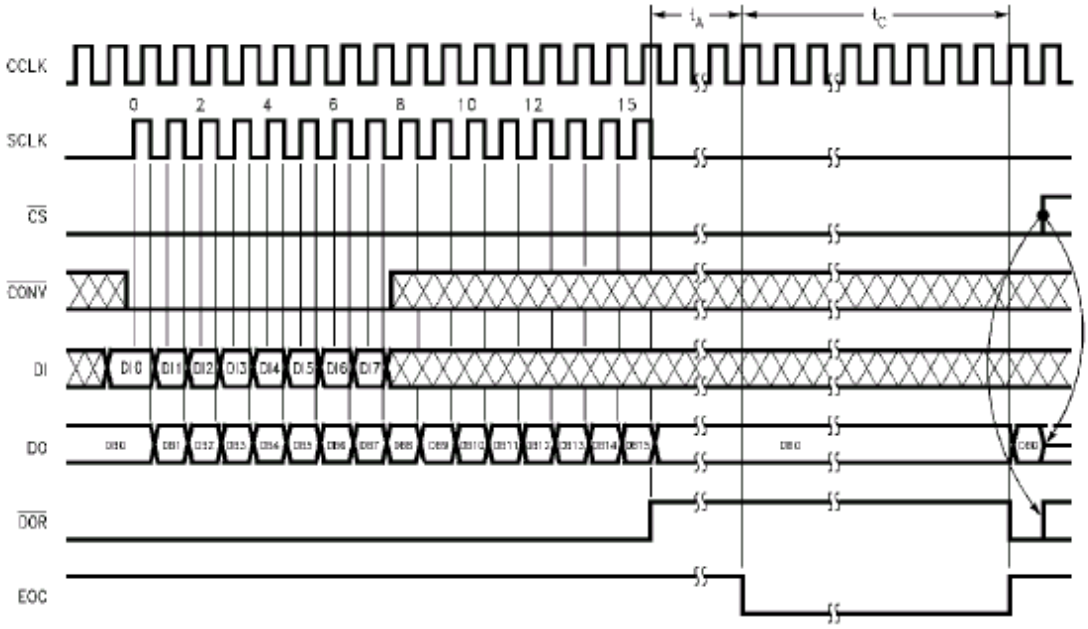
V_{REF}^-: Entrada de referencia negativa. No debe estar por debajo de GND y no debe exceder V_A^+

V_A^+, V_D^+: Tensión de alimentación analógica y digital de 3.0 V_{DC} a 5.5 V_{DC}

DGND: Masa digital.

AGND: Masa analógica.

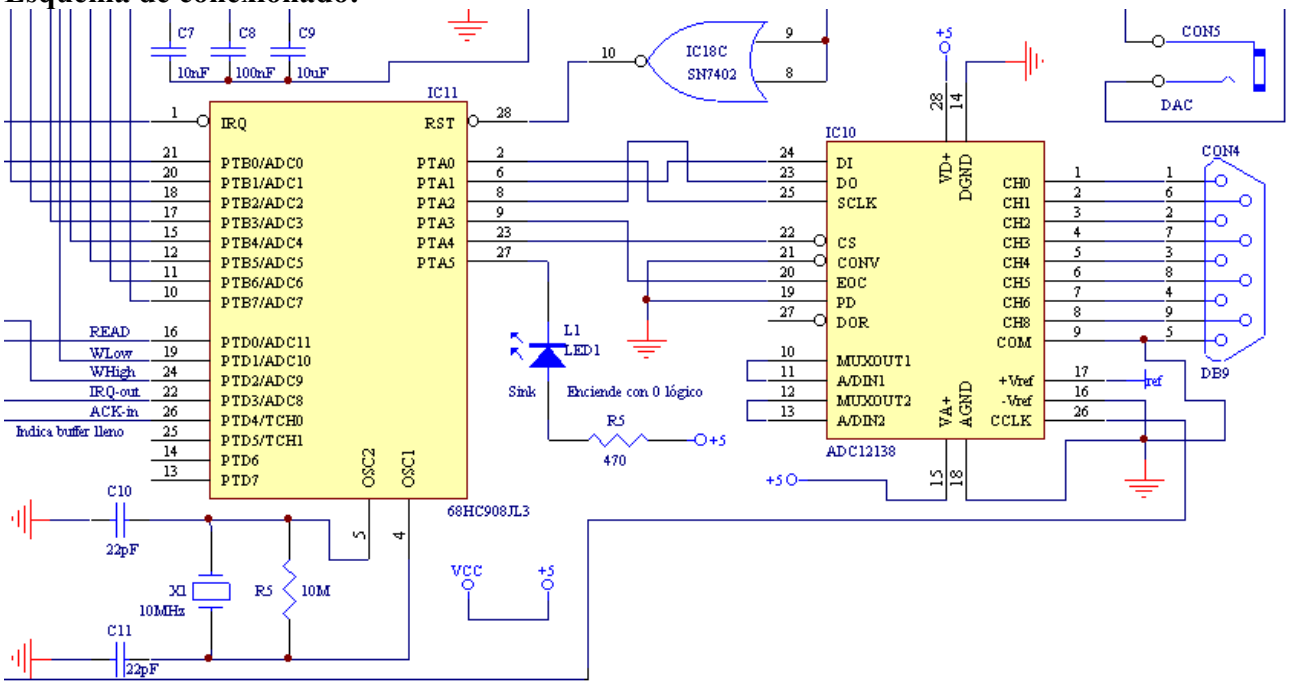
En el siguiente grafico, se puede observar el Diagrama de Tiempo de la comunicación μc / ADC:



3.5.4 Conexión del microcontrolador al ADC:

La transferencia entre el μc y el ADC, es Serial Sincrónica, que responde a la norma SPI de Motorola, y Microwire de National Semiconductor.
La señal analogica a convertir se ingresa a traves del conector DB9 a los 8 canales seleccionables del Conversor.

Esquema de conexionado:



El µc envía la palabra de control en forma serial, desde el bit 1 del puerto A hasta la entrada DI del conversor.
Esta palabra contiene la informacion necesaria para configurar el ADC, a saber:
_ Que canal se esta usando,
_ que modo (Diferencial o simple),
_ comandos (formato de la palabra de salida); mínimo 12 bits.

Canal

Diferencial
- 00000100 canal 0 - canal 1
- 00010100 canal 2 - canal 3
- 00100100 canal 4 - canal 5
- 00110100 canal 6 - canal 7
- 01000100 canal 0 - canal 1
- 01010100 canal 2 - canal 3
- 01100100 canal 4 - canal 5
- 01110100 canal 6 - canal 7

Simple
- 10000100 canal 0
- 10010100 canal 1
- 10100100 canal 2
- 10110100 canal 3
- 11000100 canal 4
- 11010100 canal 5
- 11100100 canal 6
- 11110100 canal 7

Desde el bit 2 del puerto A, el µc entrega el reloj de la comunicación al ADC, directamente sobre el pin SCLK.

La señal de EOC ingresa al µc, a traves del bit 3 del puerto A, para informarle que el ADC ha terminado de convertir y se encuentra desocupado.
Al mismo tiempo que ya está disponible el 1º bit de la conversión anterior en DO.

Palabras de configuración:

Estas son las palabras que la PC le envia al microcontrolador para autocalibrar el ADC, poner en cero, realizar una conversión, o iniciar una conversión continua a intervalos regulares. Antes de realizar cualquiera de estas operaciones se deben enviar las palabras de configuracion para especificar el canal del ADC a utilizar, el modo (simple o diferencial), y el intervalo de tiempo entre una conversión y la siguiente para la conversión continua. Estas son enviadas una tras otra y la primera especifica el modo y el canal, la segunda y la tercera son el byte mas significativo y el menos significativo respectivamente de un numero el cual multiplicado por 1 milisegundo da el tiempo entre converciones. Cabe aclarar que le microcontrolador realizara una configuracion por defecto si estos parametros no son especificados, la cual es, modo simple, canal de entrada 1, e intervalo entre conversiones de 1 milisegundo.

Comandos:

02h 00000010 Conversión simple
[Devuelve la conversion de lo que hay en la entrada seleccionada en ese instante]
03h 00000011 Inicio conversión continua
[Inicia la conversión a intervalos regulares, previamente configurados]
06h 00000110 Stop conversión continua
[Detiene la conversión a intervalos regulares]
08h 00001000 Autocalibración
[Calibra automaticamente el ADC]
09h 00001001 Autocero
[Ajusta el cero del ADC]

3.5.5 Firmware del Microcontrolador:

A continuacion se puede observar el programa del microcontrolador (en assembler):

```
RAM             EQU  $0080
ROM             EQU  $F600                          ; Valid for all JL3, JK3, JK1
VECTOR          EQU  $FFDE

OUTADC          EQU  $0081                          ;REGISTRO DE SALIDA AL ADC
INADCH          EQU  $0082                          ;REGISTRO DE ENTRADA DEL ADC ALTO
INADCL          EQU  $0083                          ;REGISTRO DE ENTRADA DEL ADC BAJO
CONT1           EQU  $0084                          ;CONTADOR DE LOS DATOS DE CONFIGURACION
CONF            EQU  $0085                          ;PALABRA DE ENTRADA PARA CONFIGURACION DE LOS CANALES
Y EL MODO(SIMPLE O DIFERENCIAL)

AUTOCAL         EQU  $0008                          ;PALABRA PARA AUTOCALIBRACION
AUTOZERO        EQU  $0009                          ;PALABRA PARA AUTOCERO
READSTAT        EQU  $000C                          ;PALABRA PARA LEER EL REGISTRO DE STATUS
CCLK_ACQ        EQU  $004E                          ;TIEMPO DE ADQUISICION 10 CICLOS DE CCLK
NO_SIGN         EQU  $000D                          ;PALABRA PARA CONFIGURAR SALIDA SIN SIGNO
SIGN            EQU  $008D                          ;PALABRA PARA CONFIGURAR SALIDA CON SIGNO
SIMPLE          EQU  $0002                          ;PALABRA PARA CONVERSION SIMPLE \
START           EQU  $0003                          ;PALABRA PARA CONVERSION CONTINUA|COMANDOS DEL
USUARIO DE LA PC PARA EL   MANEJO DE LA CONVERSION
STOP            EQU  $0006                          ;PALABRA PARA PARAR CONVERSION   /
DEFECTO         EQU  $0084                          ;CONFIGURACION POR DEFECTO (CANAL 0, MODO SIMPLE, 12 BIT +
BIT DE SIGNO, LSB PRIMERO)

$Include 'jl3regs.inc'                              ; For the 68HC908JL3, 68HC908JK3, 68HC908JK1

        ORG ROM

INI:    MOV   #$11,CONFIG1                          ;REGISTRO DE CONFIGURACION
                                                    ;cop deshabilitado
                                                    ;lvi deshabilitado
        MOV   #$00,CONFIG2                          ;REGISTRO DE CONFIGURACION
                                                    ;PULL-UP IRQ CONECTADA
```

```
        MOV   #$00,INTSCR                       ;CONFIGURACION DE LA INTERRUCION POR IRQ
        MOV   #$33,DDRA                         ;BIT 0=OUT_SCLK   \
                                                ;BIT 1=OUT_DI       | ESTE PUERTO
                                                ;BIT 2=IN_DO        | CONTROLA EL ADC
                                                ;BIT 3=IN_EOC       |
                                                ;BIT 4=OUT_CS       |
                                                ;BIT 5=OUT_LED    /
        BSET  4,PORTA                                   ;ACTIVO CS
        MOV   #$00,DDRB                         ;PUERTO PARA LOS DATOS (BIDIRECCIONAL)
        MOV   #$0F,DDRD                         ;BIT 0=OUT_OE-LATCH(ACTIVO POR BAJO) ENTRADA DE DATOS
        \ESTE PUERTO CONTROLA LA
                ;BIT 1=OUT_CLK-LATCH(ACTIVO POR ALTO) SALIDA LATCH LSB |COMUNICACION CON LA PC
                                                ;BIT 2=OUT_CLK-LATCH(ACTIVO POR ALTO) SALIDA LATCH MSB |
                ;BIT 3=OUT_IRQPC (ESTA CONECTADO A LA IRQ DE LA PC)    /
                ;BIT 4=IN_LECTURA DE DATO
        MOV   #$03,TMODH                        ;CONFIGURA EL MODULO DEL TIMER
        MOV   #$E8,TMODL                        ;CONFIGURA EL MODULO DEL TIMER
        MOV   #$05,PORTD                        ;PONGO EN 1 EL OUT ENABLE DEL LATCH DE ENTRADA DE DATOS
(SE ACTIVA POR BAJO)
        CLRX
        CLRA
        CLR   PORTB
        CLR   INADCH
        CLR   INADCL
        CLR   CONT1

        MOV   #AUTOCAL,OUTADC                   ;GUARDO PARA ENVIAR AUTOCALIBRACION
        BSR   ENVIAR                            ;ENVIO Y RECIBO LOS DATOS

LOOP4 BRCLR 3,PORTA,LOOP4                       ;ESPERO EOC

        MOV   #READSTAT,OUTADC
        BSR   ENVIAR1                           ;ENVIO PALABRA PARA LEER STATUS

LOOP5 BRCLR 3,PORTA,LOOP5                       ;ESPERO EOC

        MOV   #READSTAT,OUTADC
        BSR   ENVIAR1                           ;ENVIO PALABRA PARA LEER STATUS
LOOP6 BRCLR 3,PORTA,LOOP6                       ;ESPERO EOC
        MOV   #CCLK_ACQ,OUTADC
        BSR   ENVIAR1                           ;ENVIO PALABRA PARA CONFIGURAR EL TIEMPO DE ADQUISICION
EN 10 CICLOS DE CCLK

LOOP7 BRCLR 3,PORTA,LOOP7                       ;ESPERO EOC

        MOV   #SIGN,OUTADC
        BSR   ENVIAR1                           ;ENVIO PALABRA PARA CONFIGURAR SALIDA CON SIGNO

LOOP8 BRCLR 3,PORTA,LOOP8    ;ESPERO EOC

        MOV   #AUTOZERO,OUTADC
        BSR   ENVIAR1                           ;ENVIO PALABRA DE AUTOCERO

LOOP9 BRCLR 3,PORTA,LOOP9                       ;ESPERO EOC

        MOV   #DEFECTO,CONF                     ;GUARDO EN LA PALABRA DE CONFIGURACION DE LA PC CANAL
CERO MODO SIMPLE

        MOV #$10,CONT1
VU3   BCLR 5,PTA                                ;APAGA EL LED
        BSR  DELAY
        BSET 5,PTA                              ;ENCIENDE LED
        BSR  DELAY
        DEC  CONT1
        LDA  CONT1
        BNE  VU3
        CLR  CONT1
        CLI

NADA  NOP
        NOP
        JMP NADA

DELAY:
        LDX #$FF
VU2     NOP
        LDA #$FF
```

```
VU1     NOP
        DBNZA VU1
        DBNZX VU2
        RTS


*******************************RUTINA DEL IRQ PARA LA ENTRADA DE DATOS*****************************************
INT_IRQ:
        BSET   5,PORTA                          ;ENCIENDE EL LED
        LDA   CONT1
        CBEQA  #$00,CONFIG
        CBEQA  #$01,TIMERL
        CBEQA  #$02,TIMERH
        RTI

************************SUBRUTINA SETEO DEL TIEMPO ENTRE MUESTREOS LSB************************************
TIMERL:

        MOV   #$00,DDRB                         ;CONFIGURO EL PUERTO DE DATOS COMO ENTRADA
        BCLR  0,PORTD                           ;HABILITO LAS SALIDAS DEL LATCH DE ENTRADA DE DATOS
        NOP
        NOP
        NOP
        LDA   PORTB                             ;GUARDO LA PALABRA DE ENTRADA
        BSET  0,PORTD                           ;DESHABILITO LAS SALIDAS DEL LATCH DE ENTRADA
        STA   TMODL                             ;CONFIGURO EL MODULO DEL TIMER LSB
        INC   CONT1                             ;INCREMENTO CONTADOR
        RTI

ENVIAR1:
        BSR   ENVIAR2
        RTS
************************SUBRUTINA SETEO DEL TIEMPO ENTRE MUESTREOS MSB************************************
TIMERH:

        MOV   #$00,DDRB                         ;CONFIGURO EL PUERTO DE DATOS COMO ENTRADA
        BCLR  0,PORTD                           ;HABILITO LAS SALIDAS DEL LATCH DE ENTRADA DE DATOS
        NOP
        NOP
        NOP
        LDA   PORTB                             ;GUARDO LA PALABRA DE ENTRADA
        BSET  0,PORTD                           ;DESHABILITO LAS SALIDAS DEL LATCH DE ENTRADA
        STA   TMODH                             ;CONFIGURO EL MODULO DEL TIMER LSB
        CLR   CONT1                             ;RESETEO CONTADOR
        RTI



***SUBRUTINA DE CONFIGURACION***

CONFIG:
        MOV   #$00,DDRB                         ;CONFIGURO EL PUERTO DE DATOS COMO ENTRADA
        BCLR  0,PORTD                           ;HABILITO LAS SALIDAS DEL LATCH DE ENTRADA DE DATOS
        NOP
        NOP
        NOP
        LDA   PORTB                             ;GUARDO LA PALABRA DE ENTRADA
        BSET  0,PORTD                           ;DESHABILITO LAS SALIDAS DEL LATCH DE ENTRADA

        CMP   #SIMPLE
        BNE   SALTO_1

        MOV   CONF,OUTADC
        BSR   ENVIAR2                           ;ENVIO PALABRA DE CONVERSION
ESP1    BRCLR 3,PORTA,ESP1                      ;ESPERO EOC
        MOV   CONF,OUTADC
        BSR   ENVIAR2                           ;ENVIO PALABRA DE CONVERSION PARA LEER CONVERSION
ANTERIOR
ESP2    BRCLR 3,PORTA,ESP2                      ;ESPERO EOC
        MOV   INADCL,PORTB                      ;PONGO LA PARTE BAJA DEL DATO EN EL PUERTO
MOV   #$FF,DDRB                                 ;CONFIGURO EL PUERTO DE DATOS COMO SALIDA
        BSET  1,PORTD                           ;LATCHEO EL DATO LSB
        NOP
        NOP
        NOP
        BCLR  1,PORTD
```

```
        MOV   INADCH,PORTB                  ;PONGO LA PARTE ALTA DEL DATO EN EL PUERTO
        BSET  2,PORTD                       ;LATCHEO EL DATO MSB
        NOP
        NOP
        NOP
        BCLR  2,PORTD
;WAIT1   BRSET 4,PORTD,WAIT1                ;VERIFICA QUE LA PC LEYO EL DATO ANTERIOR, A TRAVES DEL
FLIP-FLOP
        BCLR  3,PORTD                       ;INTERRUMPO A LA PC
        NOP
        NOP
        NOP
        NOP
        BSET  3,PORTD
        CLR   CONT1                         ;PONGO EN CERO EL CONTADOR PORQUE NO SE VA A SETEAR EL
TIEMPO ENTRE          MUESTREOS
        RTI

SALTO_1  CMP   #AUTOCAL
        BNE   SALTO_2

        MOV   #AUTOCAL,OUTADC
        BSR   ENVIAR                        ;MANDO PALABRA PARA AUTOCALIBRACON
ESP3    BRCLR 3,PORTA,ESP3                  ;ESPERO EOC
        MOV   #$FF,DDRB                     ;CONFIGURO EL PUERTO DE DATOS COMO SALIDA
        MOV   #$FF,PORTB                    ;ENVIO FFFF PARA CONFIRMAR AUTO CALIBRACION
        BSET  1,PORTD                       ;LATCHEO DATO LSB
        NOP
        NOP
        NOP
        BCLR  1,PORTD
        BSET  2,PORTD                       ;LATCHEO DATO MSB
        NOP
        NOP
        NOP
        BCLR  2,PORTD
        BCLR  3,PORTD                       ;INTERRUMPO LA PC
        NOP
        NOP
        NOP
        BSET  3,PORTD
        CLR   CONT1                         ;PONGO EN CERO EL CONTADOR PORQUE NO SE VA A SETEAR EL
TIEMPO ENTRE MUESTREOS
        RTI

ENVIAR2  BSR   ENVIAR
        RTS

SALTO_2  CMP   #AUTOZERO
        BNE   SALTO_3

        MOV   #AUTOZERO,OUTADC
        BSR   ENVIAR                        ;MANDO PALABRA PARA AUTOCERO
ESP4    BRCLR 3,PORTA,ESP4                  ;ESPERO EOC
        MOV   #$FF,DDRB                     ;CONFIGURO EL PUERTO DE DATOS COMO SALIDA
        MOV   #$FF,PORTB                    ;ENVIO FFFF PARA CONFIRMAR AUTO CERO
        BSET  1,PORTD                       ;LATCHEO DATO LSB
        NOP
        NOP
        NOP
        BCLR  1,PORTD
        BSET  2,PORTD                       ;LATCHEO DATO MSB
        NOP
        NOP
        NOP
        BCLR  2,PORTD
        BCLR  3,PORTD                       ;INTERRUMPO LA PC
        NOP
        NOP
        NOP
        BSET  3,PORTD
        CLR   CONT1                         ;PONGO EN CERO EL CONTADOR PORQUE NO SE VA A SETEAR EL
TIEMPO ENTRE MUESTREOS
        RTI

SALTO_3
        CMP   #START
```

```
        BNE   SALTO_4                   ;ESTA RUTINA CONFIGURA Y PRENDE EL TIMER

        MOV   #$60,TSC                  ;CONFIGURA EL TIMER(TOIE,TSTOP,PRESCALER/1)
        BCLR  5,TSC                     ;SACA EL BIT DE STOP
        CLR   CONT1                     ;PONGO EN CERO EL CONTADOR PORQUE NO SE VA A SETEAR EL
TIEMPO ENTRE MUESTREOS
        CLI
        RTI

SALTO_4 CMP   #STOP
        BNE   SALTO_5                   ;ESTA RUTINA APAGA EL TIMER

        BSET  5,TSC                     ;PARO EL CONTADOR
        LDX   TSC                       ;PARA PODER PONER EN CERO EL TOF HAY QUE LEER EL TSC
        BCLR  7,TSC                     ;PONGO EN CERO EL TOF
        BSET  4,TSC                     ;RESETEO EL CONTADOR
        MOV   #$00,TSC                  ;DESHABILITO LA INTERRUPCION
        BSET  5,TSC                     ;PARO EL CONTADOR
        CLR   CONT1                     ;PONGO EN CERO EL CONTADOR PORQUE NO SE VA A SETEAR EL
TIEMPO ENTRE MUESTREOS
        RTI

SALTO_5 STA   CONF                      ;GUARDO LA CONFIGURACION DE LOS CANALES Y EL MODO
        INC   CONT1                     ;INCREMENTO CONTADOR
        RTI

;////////////////////RUTINA PARA ENVIAR Y RECIBIR DATOS AL Y DESDE EL ADC ////////////////////
ENVIAR:
        BCLR  0,PORTA                   ;LIMPIO SCLK
        BCLR  4,PORTA                   ;PONGO EN 0 CS
        BCLR  1,PORTA                   ;LIMPIO DI

        LDA   #$08                      ; 8 BITS ES EL PRIMER BYTE
        NOP
        NOP
        NOP
LOOP1   BCLR  0,PORTA                   ;FLANCO DE BAJADA DEL SCLK
        BRSET 2,PORTA,SETC              ;\
        CLC                             ; | GUARDO EL DATO
FIN_SETC ROR  INADCL                    ;/
        ROL   OUTADC                    ;\
        BCC   CLRDI                     ; | PONGO EN EL PUERTO EL DATO A ENVIAR
        BSET  1,PORTA                   ;/
        NOP
        NOP
        NOP
        NOP
FIN_CLRDI BSET  0,PORTA                 ; FLANCO DE SUBIDA DEL SCLK
        DBNZA LOOP1                     ; DECREMENTO Y SALTO SI NO ES CERO

        LDA   #$05                      ; 5 BITS PARA COMPLETAR LOS 13
        NOP
        NOP
        NOP
LOOP2   BCLR  0,PORTA                   ;FLANCO DE BAJADA DEL SCLK
        BRSET 2,PORTA,SET1              ;\
        CLC                             ; | GUARDO EL DATO
FIN_SET1 ROR  INADCH                    ;/
        BCLR  1,PORTA                   ;LIMPIO DI PORQUE NO ES NECESARIO ENVIAR DATOS
        NOP
        NOP
        BSET  0,PORTA                   ; FLANCO DE SUBIDA DEL SCLK
        NOP
        NOP
        DBNZA LOOP2                     ; DECREMENTO Y SALTO SI NO ES CERO
        BCLR  0,PORTA

        LDA   #$03                      ;LOS TRES RESTANTES PARA COMPLETAR LOS 2 BYTES
LOOP3   CLC                             ;LOS PONGO A TODOS EN CERO
        ROR   INADCH
        DBNZA LOOP3
        BSET  4,PORTA                   ;LEVANTO EL CS
        BCLR  5,PORTA                   ;APAGA EL LED
        RTS
SETC:
        SEC
        JMP FIN_SETC
```

```
CLRDI:
        BCLR  1,PORTA
        JMP   FIN_CLRDI
SET1:
        SEC
        JMP  FIN_SET1

;////////////////////////////////////////////////////////////////////////

TIMER:

        BSET  5,TSC                             ;PARO EL CONTADOR DEL TIMER
        MOV   #$10,TSC                          ;RESETEO EL TIMER
        BSET  5,TSC

MOV   CONF,OUTADC
        BSR   ENVIAR                            ;ENVIO PALABRA DE CONVERSION
ESP5    BRCLR 3,PORTA,ESP5                      ;ESPERO EOC
        MOV   CONF,OUTADC
        BSR   ENVIAR                            ;ENVIO PALABRA DE CONVERSION PARA LEER CONVERSION
ANTERIOR
ESP6    BRCLR 3,PORTA,ESP6                      ;ESPERO EOC
        MOV   #$FF,DDRB                         ;CONFIGURO EL PUERTO DE DATOS COMO SALIDA
        MOV   INADCL,PORTB                      ;PONGO LA PARTE BAJA DEL DATO EN EL PUERTO
        BSET  1,PORTD                           ;LATCHEO EL DATO LSB
        NOP
        NOP
        NOP
        BCLR  1,PORTD
        MOV   INADCH,PORTB                      ;PONGO LA PARTE ALTA DEL DATO EN EL PUERTO
        BSET  2,PORTD                           ;LATCHEO EL DATO MSB
        NOP
        NOP
        NOP
        BCLR  2,PORTD
;WAIT2    BRSET 4,PORTD,WAIT                     ;VERIFICA QUE LA PC LEYO EL DATO ANTERIOR, A TRAVES DEL
FLIP-FLOP
        BCLR  3,PORTD                           ;INTERRUMPO A LA PC
        NOP
        NOP
        NOP
        BSET  3,PORTD

        MOV  #$40,TSC                           ;HABILITO LA INTERRUPCION
        RTI

dummy_isr:

        rti                                     ; return


 ORG  VECTOR

        dw  dummy_isr     ; ADC Conversion Complete Vector
        dw  dummy_isr     ; Keyboard Vector
        dw  dummy_isr     ; (No Vector Assigned $FFE2-$FFE3)
        dw  dummy_isr     ; (No Vector Assigned $FFE4-$FFE5)
        dw  dummy_isr     ; (No Vector Assigned $FFE6-$FFE7)
        dw  dummy_isr     ; (No Vector Assigned $FFE8-$FFE9)
        dw  dummy_isr     ; (No Vector Assigned $FFEA-$FFEB)
        dw  dummy_isr     ; (No Vector Assigned $FFEC-$FFED)
        dw  dummy_isr     ; (No Vector Assigned $FFEE-$FFEF)
        dw  dummy_isr     ; (No Vector Assigned $FFF0-$FFF1)
        dw  TIMER         ; TIM1 Overflow Vector
        dw  dummy_isr     ; TIM1 Channel 1 Vector
        dw  dummy_isr     ; TIM1 Channel 0 Vector
        dw  dummy_isr     ; (No Vector Assigned $FFF8-$FFF9)
        dw  INT_IRQ       ; ~IRQ1
        dw  dummy_isr     ; SWI Vector
        dw  INI           ; Reset Vector
```

4 Conclusiones

Luego de haber analizado las especificaciones y requerimientos para establecer una interfaz para el BUS ISA se procedió a la integración de los diversos módulos conformantes. Una vez finalizada la construcción de la placa fue conectada a una PC con dos ranuras para BUS ISA, pudiéndose comprobar su correcto funcionamiento.

En el presente infrome han quedad temas sin tratar, posteriormente se elavorará una segunda revisión del mismo contemplando los temas faltantes, sin embargo se han tratado todos los temas necesarios como para emprender el diseño y puesta en marcha de cualquier sistema basado en el BUS ISA de 16 bits.

Deseamos agradecer a la Cátedra Técnicas Digitales II de la UTN Regional Córdoba por la ayuda recibida.

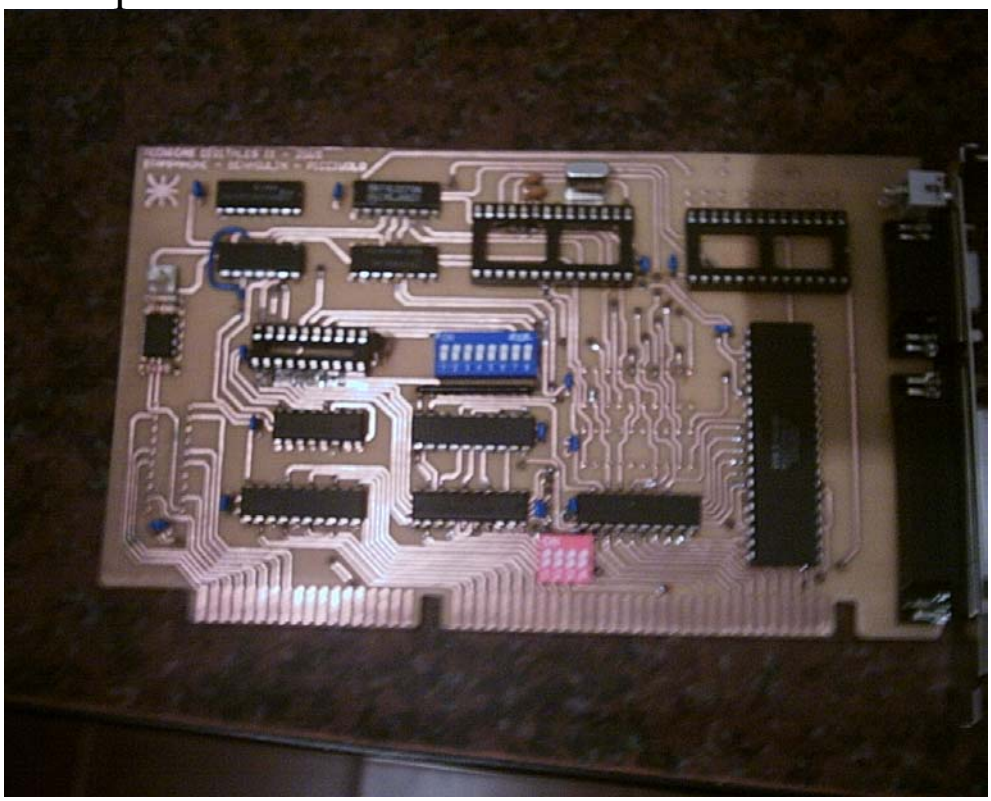
Esperamos que los nuevos estudiantes de Técnicas Digitales II amplien el trabajo realizado, extendiendoló a desarrollos basados en otros estándares de conexión como pueden ser el BUS PCI o el puerto USB.

Integrantes del Grupo:

Leandro Stampanone -leo222222@hotmail.com
Benasulin, Dimas -dimas88@hotmail.com
Picciuolo Fabricio -zonazener@hotmail.com

5 Anexo : Fotos

Lado de Componentes:



Lado de Cobre: