ADIO (Analogic Digital Input Output)
Developed by: Eng. Raimundo Rodulfo
1994

ORIGINAL PROJECT PAPERS (IN SPANISH)

El proyecto ADIO (Analogic Digital Input Output) consistió básicamente en el diseño e implementación de un sistema de interfaz analógico/digital y digital/analógico (de entrada y salida) con un microcomputador (PC), a través de un puerto paralelo; y un *software* de control necesario para operar el sistema y desarrollar una aplicación de salida sobre la plataforma, que puede ser como unidad DSP, generador de señales con amplitud y frecuencia variable, voltímetro u osciloscopio, todas en el rango de ±5 V, y permitiendo la introducción de parámetros al computador por el usuario.

La realización del proyecto involucró el diseño de un módulo de *hardware*, y varios módulos de *software*, consistente el primero en la circuitería necesaria para la interfaz de los datos de entrada/salida del puerto paralelo del computador con la entrada/salida de voltaje analógico, y los últimos en las rutinas codificadas en lenguaje de bajo y alto nivel que implementan la lógica de cálculo y control del sistema.

A pesar de la relativa especificidad del *performance* del *software* diseñado para el sistema, el módulo del *hardware* es de una enorme versatilidad funcional, que soporta cualquier tipo de aplicaciones en las que se desee leer, procesar o escribir una señal analógica dentro del rango de trabajo del equipo (±5 V), dejando de ser esto último una restricción si se incluye una etapa amplificadora-atenuadora de tensión, ajustable en el rango deseado, a la entrada/salida del sistema, permitiendo también aumentar la potencia de salida a los niveles requeridos por la carga que se vaya a acoplar. Esto hace al proyecto realizado una poderosa herramienta para laboratorios o plantas en los que se realice control de procesos automatizados a través de un computador, o para usuarios de procesos DSP en instrumentación, música y entretenimiento.

A continuación se presenta una descripción detallada de la estructura y el funcionamiento de las partes que conforman el proyecto realizado, así como los criterios asumidos en el diseño del mismo.

CRITERIOS ASUMIDOS

El proyecto realizado involucra la consideración de un gran número de aspectos técnicos y generales que ameritaron una planificación inicial de las directrices a tomar durante la fase inicial de planteamiento del problema y levantamiento de información, que habrían de esbozar los conceptos primarios de diseño.

El proyecto debía cumplir no sólo con las especificaciones dadas durante su asignación, sino también con una serie de condiciones inherentes a su naturaleza, que están implícitamente vinculadas a su concepción. Una metodología de diseño que permite tomar en cuenta las diferentes variables funcionales de un proyecto, es la conocida como *Top-Down Design*, la cual se aplicó teórica y empíricamente a la realización del proyecto. Los criterios básicos que esboza esta metodología se presentan a continuación, describiendo para cada uno de ellos la forma en que puede ser aplicado al caso particular del proyecto encomendado.

TOP-DOWN DESIGN.

1. *Uso del equipo*.
   Está relacionado directamente con la función o funciones que debe realizar el equipo, que para este caso particular están plenamente definidas en las especificaciones iniciales de utilización del proyecto. Este aspecto es el que mas influyó en la concepción inicial del proyecto, dirigiendo casi todos los esfuerzos a cumplir con las especificaciones de funcionamiento.
2. *Ambiente de trabajo.*
   El sistema diseñado opera con un microcomputador, por lo tanto el ambiente de trabajo en el que se utilice debe cumplir las mismas condiciones óptimas de temperatura, humedad y ruido eléctrico y mecánico tolerables que se requieren para la operación aceptable y segura de un equipo de computación. Esto pone pocas restricciones en lo que a eso respecta, pues prácticamente todos los dispositivos electrónicos disponibles en el mercado operan sin problema en los rangos de parámetros ambientales de los dispositivos de las computadoras.
3. *Quién lo utiliza.*
   El uso de este equipo pertenece a un amplio espectro de aplicaciones, que van desde el control de procesos automatizados, a la instrumentación electrónica e industrial, y el procesamiento de señales por software. Las dos primeras áreas son de aplicación básicamente industrial, académica y científica, mientras que la tercera puede encontrarse asociada a equipos de uso doméstico, como VCR's, procesadores de señal de sonido (Equalizadores, Delays, Mezcladores, Sintetizadores y Efectos de sonido digitales en general), etc. Para las aplicaciones escogidas para el sistema (generador

de señales y voltímetro), es obvio que el equipo seá usado por personal técnico, o estudiantes de cualquier especialidad vinculada a las mediciones eléctricas. Se encuentra en prototipo una aplicación de esta plataforma como sistema de procesamiento de señales de audio en tiempo real, que pienso utilizarla para guitarra eléctrica.

4. *Quién lo mantiene.*
   La complejidad del sistema desde el punto de vista de *hardware* y *software*, hacen necesaria la restricción de las acciones de mantenimiento del equipo a personal técnico especializado. Esta consideración influye en el diseño, obligando a dar el mayor número de facilidades al técnico que va a prestar mantenimiento del equipo, haciendo desmontable las partes, con una distribución clara e inteligible de los componentes, dejando la posibilidad de un reemplazo sencillo de partes, y suministrando toda la documentación técnica necesaria, a través de un manual contentivo de todos los planos y diagramas requeridos para definir cabalmente el equipo en todas sus facetas.
5. *Diagrama de bloques del equipo.*
   Este diagrama esboza de manera superficial las partes y módulos que componen el sistema, así como el flujo de energía, señales e información entre éstos.

LA SOLUCIÓN POR *HARDWARE*

El *hardware* del sistema está compuesto principalmente por las siguientes etapas:

- Etapa de Conversión A/D (etapa de entrada).
- Etapa de Conversión D/A (etapa de salida).
- Etapa de Interfaz con el Computador (por el puerto paralelo).

A continuación se describen estas etapas por separado.

Etapa de Conversión A/D.

Esta etapa se basa principalmente en el conversor analógico-digital ADC0804, el cual se encarga de transformar la entrada analógica en el rango de ±5 V a una salida digital de 8 bits de codificación relativa. El puerto paralelo del computador sólo posee 5 líneas de entrada, lo cual obliga a capturar el byte de salida del ADC nibble a nibble. Esto se logró utilizando el multiplexor 74LS257, el cual permite leer una entrada de 8 bits de 4 en 4 bits (el nibble alto y el nibble bajo), seleccionando el nibble a leer con una entrada de control. Se seleccionó el ADC0804 por tener generador de reloj interno (a diferencia del ADC0800), lo cual representa una ventaja al involucrar menor circuitería externa (y menor costo, por supuesto). El ADC0804 está diseñado para convertir en el rango de 0 a 5 V en su entrada analógica; para configurarlo en el rango de -5 a 5 V, se colocó un divisor de tensión en su entrada, que permite obtener una tensión en el nodo intermedio de 0 a 5 V para voltajes de -5 a 5 V en el extremo variable. La correspondencia de entrada-salida del ADC se muestra en la tabla 1.

Las líneas de control que requirió esta etapa son la de inicio de conversión del ADC, y las de selección de nibble y habilitación del multiplexor, las cuales provienen del bus de control del puerto paralelo, y son manejadas por software. El control de habilitación del multiplexor 74LS257 (entrada G), permite multiplexar las lineas de entrada al puerto para ser utilizadas para identificación del puerto por software y para la entrada de conversión A/D. La unión de las líneas RD y WR del ADC0804 permiten un control de conversión más simple a través de una sola línea. .

V_{IN} (V)	B7	B6	B5	B4	B3	B2	B1	B0
5	1	1	1	1	1	1	1	1
:	:	:	:	:	:	:	:	:
0	1	0	0	0	0	0	0	0
:	:	:	:	:	:		:	:
-5	0	0	0	0	0	0	0	0

TABLA 1
Correspondencia de Entrada-Salida del ADC

Etapa de Conversión D/A.

Esta etapa utiliza los conversores digital-analógico DAC0800 y DAC0808, el primero para generar la salida analógica deseada a partir de la entrada proveniente del puerto, y el segundo para generar el valor de referencia del primero, que permita controlar la amplitud de la salida analógica. Ambos DAC obtienen sus entradas de 8 bits provenientes del bus de datos del puerto paralelo del μC, por lo que fué necesario multiplexar las líneas de datos para la referencia y la salida. Esto se implementó con dos latches octales tri-state 74LS373, que se habilitan con mutua exclusión (uno a la vez), dependiendo del dato a sacar (referencia o salida). Se seleccionó el DAC0800 para la salida por poseer excursión simétrica bipolar de hasta 20 Vp-p a la salida (lo cual permite el rango de trabajo deseado de ±5 V), y el DAC0808 para la referencia por poseer salida unipolar de hasta 10 V (lo que permite un rango de amplitud de 5 V).

Cálculo de las resistencias de polarización de los DAC's.

Para una corriente de full escala I_{FS} de 1 mA en el DAC0808, tenemos:

$$I_{FS1} = \frac{Vref_1}{Rref_1} \Rightarrow Rref_1 = \frac{Vref_1}{Iref_1} = \frac{5}{1} = 5\ K\Omega$$

Con una resistencia a la salida igual a $Rref_1$ (5 KΩ), obtenemos una salida de 0 a 5 V.

Para una corriente de full escala máxima $I_{FS2(max)}$ de 1 mA en el ADC0800, tenemos:
$Rref_2$ = 5 KΩ (para una entrada de referencia máxima $Vref_2(max)$ de 5 V)

A la salida del DAC0808:

$$I_o + \overline{I_o} = I_{FS2} \qquad (1)$$

$V_o = V_{o+} - V_{o-}$

$V_{o+} = Vref_1 + I_o R$

$V_{o-} = Vref_1 + \overline{I_o} R$

$\Rightarrow V_0 = R(Io - \overline{Io})$ (2)

(1) en (2):

$V_o = R\left[I_o - (I_{FS2} - I_o)\right] = R(2I_o - I_{FS2})$ (3)

De la hoja técnica del DAC0800 (ver anexos) obtenemos la correspondencia entre la entrada y la salida que se muestra en la tabla 2.

B1	B2	B3	B4	B5	B6	B7	B8	Io (mA)	$\overline{Io}$ (mA)
1	1	1	1	1	1	1	1	I_{FS}	0
:	:	:	:	:	:	:	:	:	:
1	0	0	0	0	0	0	0	$I_{FS}/2$	$I_{FS}/2$
:	:	:	:	:	:	:	:	:	:
0	0	0	0	0	0	0	0	0	I_{FS}

TABLA 2
Correspondencia de Entrada-Salida del DAC0800

Sea D el valor decimal de $B_1..B_8$ $(0 \le D \le 255)$; Entonces:

$$I_o = \frac{D}{255} I_{FS2} \quad (4)$$

(4) en (3):

$$V_o = R(2\frac{D}{255} - 1) I_{FS2} \quad (5)$$

$$I_{FS2} = \frac{Vref_1}{Rref_2} \quad (6)$$

(6) en (5):

$$V_o = \frac{R}{Rref_2}(\frac{2}{255} D - 1) Vref_1$$

Si hacemos $R = Rref_2 = 5\ K\Omega$, tenemos:

$$V_o = (\frac{2}{255} D - 1) Vref_1$$

De esta forma obtenemos la correspondencia de entrada-salida de voltaje para el DAC0800 que se muestra en la tabla 3.

B1	B2	B 3	B 4	B 5	B 6	B 7	B 8	Vo (V)
1	1	1	1	1	1	1	1	V_{ref1}
:	:	:	:	:	:	:	:	:
1	0	0	0	0	0	0	0	0
:	:	:	:	:	:	:	:	:
0	0	0	0	0	0	0	0	$-V_{ref1}$

TABLA 3
Correspondencia de Entrada-Salida de Voltaje del DAC0800

V_{ref1} determina la amplitud, que va desde 0 a 5 V, obteniéndose así una excursión del voltaje de salida máxima de -5 a 5 V.

Etapa de Interfaz con el Computador.

Esta etapa consiste físicamente en un conector DB-25 que se acopla al puerto paralelo del μC, y conecta a éste con el circuito a través de un cable plano. Del puerto paralelo se toman las señales de control del sistema manejadas por software desde el computador, se reciben los datos de entrada de la etapa de conversión D/A y se envían los datos de salida de la etapa de conversión A/D.

Para efecto de hacer el sistema lo más versátil posible, y previendo la posibilidad de utilización del equipo con computadores con mas de 1 puerto paralelo, se proveyó al circuito de un elemento de identificación, que permitiera al computador (comandado por el software del sistema) saber si éste se encuentra conectado a un puerto paralelo examinado en particular. Esto se implementó con un buffer tri-state con sus salidas conectadas al puerto de status, y con sus entradas colocadas a valores fijos (palabra de identificación del módulo) distintos a los valores fijos del puerto de status sin carga conectada, de manera que el computador al examinar un puerto paralelo y leer esa palabra de identificación, sepa que es ése al que va a direccionar las operaciones de entrada/salida del sistema. Como las líneas de status también son utilizadas por el computador para la lectura de datos exteriores provenientes de la etapa del ADC, se multiplexaron estas líneas desde software, habilitando el buffer o el 74LS257 uno a la vez (controlando sus entradas de habilitación con líneas del puerto).

LA SOLUCIÓN POR *SOFTWARE*

El diseño de un sistema controlado por un microcomputador (PC), proporciona al proyectista muchos recursos, derivados de disponer de una plataforma

de *hardware* y *software* para el desarrollo de su aplicación particular. La gran versatilidad de un sistema de computación de propósito general de los que hoy se consiguen en el mercado, constituye una poderosa herramienta de diseño de sistemas automatizados, que requieran de las prestaciones de velocidad, capacidad y *performance* de un PC como módulo principal de control. Disponer de un computador dedicado en el control de un sistema de tiempo real, facilita en gran medida el proceso de diseño, y permite disfrutar de las ventajas que proporcionan los lenguajes de alto nivel y el OS para gestión y manejo de los recursos del sistema.

Para el desarrollo del sistema particular que se diseñó, se utilizaron los recursos de las rutinas de manejo de interrupciones del BIOS por el DOS, y la programación de alto y bajo nivel para los módulos de control y la interfaz con el usuario.

La codificación del software en un lenguaje de alto o medio nivel (como Pascal o C), posee grandes ventajas, pero a la vez serios inconvenientes en el control de un sistema en tiempo real. Las principales ventajas que proporcionan los lenguajes de alto nivel son las instrucciones de manejo de la pantalla, el teclado y los diversos recursos del sistema (memoria, I/O, etc.), que permiten de una manera sencilla realizar despliegues gráficos y operaciones de entrada/salida, lográndose una interfaz interactiva amigable con el usuario, y de relativamente fácil programación. La principal desventaja que conlleva la codificación del software en lenguaje de alto nivel, es que la ejecución de las instrucciones consumen tiempo extra respecto a sus equivalentes en lenguaje máquina, por los procesos de traducción (ensamblaje, enlazado, etc.) que están asociados a cada lenguaje en particular, para poder ser transformados a código directamente ejecutable por el microprocesador. Esto trae serios problemas en el control de sistemas de tiempo real, en donde la velocidad de respuesta es un factor importante a ser tomado en cuenta, y más aún en aplicaciones con especificaciones de frecuencia, como es el caso del sistema proyectado.

Por otro lado, la codificación del software del sistema en lenguaje de bajo nivel, trae consigo una complejidad mucho mayor en la codificación de rutinas de manejo de la pantalla y el teclado, y a la vez una mayor simplicidad en el manejo de los puertos y de los registros, y lo que es más importante: una mayor rapidez en la ejecución de las rutinas de entrada/salida y acceso a memoria. Esto permite obtener la mayor velocidad de ejecución posible de las rutinas de control y procesamiento de datos del sistema, pero a su vez hace prácticamente imposible (por motivos de tiempo) la realización de una interfaz amigable con el usuario, por lo complejo que resulta realizar despliegues gráficos con instrucciones de lenguaje máquina.

La Solución: Fusión de Niveles.

Consiste en integrar en el mismo programa, módulos codificados en bajo nivel, para el manejo de las operaciones de control de I/O, transferencia y procesamiento de datos del sistema, que requieren de alta velocidad de ejecución; y módulos codificados en lenguaje de alto nivel, para la realización de las rutinas de interfaz con el usuario (despliegues gráficos, introducción de datos por teclado, visualización de mensajes, etc.) y las rutinas de control del sistema que no requieran de mayor velocidad de ejecución, y se simplifique su programación en alto nivel. Para el diseño del software del sistema, se utilizó Turbo Pascal de Borland como lenguaje de alto nivel, por poseer una extensa librería gráfica y de manejo de la pantalla y el teclado, además de una gran versatilidad para el llamado a servicios del DOS y la gestión de registros internos y puertos del micro. Turbo Pascal soporta, además, la declaración de procedimientos externos, en lenguaje máquina, que son enlazados por el compilador al programa principal. Los módulos en bajo nivel se codificaron en el lenguaje ensamblador del 8086/88, y se ensamblaron con el programa ensamblador de Borland Turbo Assembler, y se enlazaron con el programa Turbo Linker, creando programas con extensión .EXE directamente ejecutables por el computador. Posteriormente se enlazaron al programa principal (elaborado en Turbo Pascal) durante la compilación del mismo, creándose el programa completo de control del sistema.

Las rutinas codificadas en bajo nivel fueron las de control de la etapa del generador de señales, ya que éste exige la mayor rapidez de ejecución posible para cumplir con las especificaciones de frecuencia. Con Turbo Pascal se codificaron todas las rutinas de interfaz con el usuario, y de la etapa del voltímetro, ya que éste no amerita velocidades de respuesta tan altas como el generador. A continuación se explica como se implementaron las dos aplicaciones, de entrada y salida, del sistema desde *software*.

Generador de Señales.

Para esta aplicación, los valores correspondientes a la amplitud y la frecuencia son leidos desde el teclado, y codificados en valores binarios de referencia y retardo, respectivamente (en alto nivel), que son introducidos a las rutinas de generación en bajo nivel como variables globales del programa general, luego de botar por el puerto de control los valores de inicialización adecuados. Previamente a la codificación de la frecuencia, se ejecuta un módulo en bajo nivel que calcula la frecuencia máxima que puede generar el sistema, en base a las prestaciones del computador. En el módulo de generación de señales, de bajo nivel, se encuentra grabada una tabla con los códigos correspondientes a los valores de la señal sinusoidal, para el número de muestras determinado. A continuación se ilustra el proceso analítico de obtención de los valores de la tabla.

Dividiendo el período del seno en N intervalos, obtenemos N muestras del seno, donde el intervalo de muestreo es $\Delta x = \frac{2\pi}{N}$.

Para generar la señal $y = A\ sen\ (x)$, donde A es la amplitud en Voltios y $x = 2\pi f$ (f: frecuencia en Hertz), necesitamos obtener los N códigos de y para $x = m\Delta x$ (m = 0,1,2,..,N-1), que producirán salidas analógicas del sistema en el rango de -A hasta A, donde A es la amplitud codificada en la referencia.
Si D es el valor decimal del código de 8 bits del valor de la señal (dato de salida), entonces $0 \le D \le 255$ (correspondiente a $-A \le Vo \le A$), y su relación con el valor de la señal es proporcional, aproximando D a una variable continua, de manera que

$$D \approx \frac{255}{2A}(y + A) = \frac{255}{2}\left[\text{sen}(x) + 1\right]$$

cuyo valor es independiente de la amplitud, como era de esperarse.
Aproximando D al valor entero mas cercano a su estimado continuo, obtenemos así los N valores de la tabla para generación de la señal sinusoidal:

$$D = Round\left(\frac{255}{2}(\text{sen}(m\frac{2\pi}{N}) + 1)\right)\ \text{m} = 0,1,..,\text{N}-1$$

Para decidir el número de muestras óptimo a tomar, se elaboró un programa en Turbo Pascal que generara las tablas de los códigos de la señal sinusoidal, y calculara los errores porcentuales debidos a la aproximación por redondeo del valor de D, para números de muestra por cuarto de ciclo ($0 \le x \le \pi/2$) entre 90 y 110, siendo 100 el número de muestras que produjo menor error porcentual promedio de los valores de las muestras (se comparó el valor de voltaje de la salida con el valor real que debería salir). En la figura 3 se presenta un listado de los valores obtenidos para $N = 4*100 = 400$ muestras/ciclo.

La tabla es barrida de principio a fín (a la frecuencia determinada) una y otra vez, y sus valores son botados por el puerto de datos, hasta que se genera una interrupción por el teclado (oprimiendo una tecla), implementada con la rutina de interrupción del BIOS para el teclado gestionada por el DOS (GetKbdInt). La frecuencia de barrido está determinada por una rutina de retardo ejecutada entre muestreos, cuya duración depende de la frecuencia introducida al sistema.

Voltímetro Digital.
Esta aplicación se implementó totalmente desde alto nivel, utilizando las instrucciones para manejo de puertos de entrada/salida incluidas en Turbo Pascal (instrucción Port). Luego de botar por el puerto de control los valores de inicialización adecuados, se inicia la conversión del ADC, y se entra en un breve ciclo de espera de fín de conversión, para después leer el dato nibble a nibble. El dato se codifica en su valor de voltaje correspondiente, y se visualiza por pantalla.