6

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

Facultad de Ingeniería

"ANALIZADOR DIGITAL DE RESPUESTA EN FRECUENCIA"

TESIS

Que para obtener el título de

INGENIERO ELECTRICO Y ELECTRONICO

presenta

CESAR ALEJANDRO JUAREZ MELCHOR

Director de Tesis: M. I. ANTONIO SALVA CALLEJA

México, D. F. Noviembre 2000

Agradecimientos:

Al Maestro Antonio Salvá Calleja por las facilidades e interés mostrados para la realización de esta tesis y sin los cuales no hubiera sido posible.

Al Laboratorio de Óptica Aplicada del Centro de Instrumentos de la Universidad Nacional Autónoma de México. Donde se realizó la parte inicial de este trabajo. En particular a los Doctores: Roberto Ortega y Dmitri Kouznetzov y a los Maestros: Carlos Román y Antonio Rodríguez.

A la Universidad Nacional Autónoma de México. Por brindarme la oportunidad de aprender.

Dedicado a mis padres y hermanos:

Gracias.

"Tener información no implica poseer inteligencia,
tener inteligencia no implica poseer sabiduría ..."

Frank Zappa 1979.

Índice

Capítulo 4 Ejemplo de aplicación

Introducción

Una de las ventajas que proporciona el empleo de las computadoras personales (PC's) en los sistemas de medición es la automatización del proceso y la facilidad para almacenar y procesar la información. Esto aunado al relativo bajo costo de una PC, resulta una excelente opción para el desarrollo de sistemas de instrumentación [4].

Con la automatización de los sistemas de medición, nacen nuevos términos con los que se hace referencia a éstos. Por ejemplo: Instrumentos Inteligentes e Instrumentos Virtuales. Un instrumento inteligente es un dispositivo con un microprocesador, al que se le programan ciertas actividades que realizará sin depender de elementos externos; sino al contrario, es capaz de auto evaluarse, corregirse y dar una respuesta satisfactoria a partir de algunos datos de entrada

El término instrumento virtual hace referencia a una simulación computarizada de un instrumento de medición, teniendo como objetivo principal brindar elementos de software personalizado, convirtiendo a la computadora en el instrumento mismo.

Algunas características importantes de ambos tipos de instrumentos se mencionan a continuación [4] [5]:

Instrumentos inteligentes:

Permiten procesamiento de datos.
Realizan acciones de detección, autoajuste y detección de fallas
Codifican y transmiten la información en forma digital

Instrumentos virtuales:

Poseen funciones definidas por el usuario
Orientados a conectividad con redes, periféricos y aplicaciones
El software es la parte más importante
Son reutilizables
Su mantenimiento es más barato

Aún más, el contar con un instrumento virtual o inteligente proporciona la ventaja de mejorar la precisión y la velocidad con que las mediciones son realizadas. También puede evitar el desgaste físico del usuario y aumentar la productividad del entorno de trabajo. Incluso en ciertas ocasiones puede ser operado por personal no especializado en el área donde se utilice el dispositivo.

El objetivo del presente trabajo es diseñar y construir un dispositivo automático capaz de obtener las curvas de respuesta en frecuencia de sistemas electrónicos sencillos, utilizando una arquitectura basada en el microcontrolador 68HC11 de la compañía Motorola y una interface computarizada programada en el lenguaje Visual Basic.

De acuerdo con los párrafos anteriores este instrumento podría ser clasificado como un instrumento inteligente, con la característica adicional de contar con un elemento interactivo computarizado que permitirá el despliegue gráfico de la información obtenida. Facilitando así la visualización e impresión de los datos generados después de realizado el análisis.

El estudio de los sistemas de control ocupa un lugar importante en el campo de la ingeniería. Es por eso que todo ingeniero debe contar con los conocimientos suficientes que le permitan analizarlos, diseñarlos e implementarlos.

El análisis de sistemas y de señales investiga el comportamiento de la señal de salida de un sistema con respecto a la entrada del mismo. A través del tiempo se han desarrollado diversos métodos para lograr este fin. Uno de estos es el de respuesta en frecuencia. La respuesta en frecuencia se define como la respuesta en estado estacionario de un sistema ante una entrada senoidal. Con este método se varia la frecuencia de la señal a la entrada del sistema, para posteriormente observar el comportamiento de la señal de salida.

El concepto central de la respuesta en frecuencia es la respuesta senoidal en magnitud y en fase como función de la frecuencia. Un mérito especial que posee este método, es el hecho de que a menudo estos datos pueden obtenerse experimentalmente

La respuesta en frecuencia proporciona la información suficiente para analizar la estabilidad relativa de los sistemas, ayuda a encontrar los parámetros necesarios para diseñar compensadores y permite buenas aproximaciones para diseños de elementos de control. Es particularmente útil en sistemas cuya función de transferencia se desconoce o posee cierta incertidumbre. La respuesta en frecuencia brinda una imagen cualitativa de la respuesta transitoria de los sistemas.

Además, el diseño de un sistema en el dominio de la frecuencia proporciona al diseñador un control del ancho de banda y alguna medida de la respuesta del sistemas a ruido y perturbaciones que no son deseados.

Los siguientes capítulos, presentarán los procesos seguidos para construir el dispositivo antes mencionado, el capítulo 1, presenta un panorama descriptivo del dispositivo y sus funciones, el capítulo 2 describe el proceso de diseño del sistema electrónico, el capítulo 3 describe tanto los programas hechos con lenguaje ensamblador, como el programa de control que se ejecuta en la PC. El capítulo 4 muestra la aplicación del sisytema al analizar un ejemplo real, presentando un acomparación entre la respuesta teórica y la respuesta obtenida con el prototipo. Finalmente se presentan las conclusiones del trabajo realizado.

Capítulo 1.

Descripción del funcionamiento del Analizador Digital de Respuesta en Frecuencia.

Descripción: *Este capítulo presenta un panorama general de los componentes del sistema, en él se describen brevemente las funciones de cada elemento. Se presenta también la terminología utilizada para hacer referencia a los elementos del mismo. Además de describirse por separado los módulos, se explica también la correlación que debe existir entre ellos para que el sistema opere correctamente.*

El Analizador Digital de Respuesta en Frecuencia (ADRF), es un dispositivo capaz de evaluar el comportamiento de sistemas electrónicos; en el dominio de la frecuencia, cuyo intervalo de operación se encuentre comprendido entre los 10 y los 50000 Hz. El dispositivo está diseñado para solicitar información al usuario, por medio de un programa de computadora, transmitir las instrucciones derivadas de estos datos al módulo de adquisición y comando, para finalmente desplegar en forma gráfica los resultados obtenidos una vez terminado el análisis.

Los tres componentes funcionales que integran el sistema ADRF se describen brevemente en seguida, además se ilustran en forma de bloques en la figura 1.

- Tarjeta de desarrollo, basada en el microcontrolador 68HC11F1, la cuál fue desarrollada por el director de esta tesis; se denomina como **SIMMP2.** Los detalles característicos de la misma se incluyen en el apéndice A

- Hardware de adquisición y arbitraje de funciones (**HAAF**). Es un conjunto de elementos electrónicos que sirven como puertos de comunicación, ligados a la SIMMP2; que a su vez controlan las funciones de elementos como, un generador de funciones, un convertidor digital – analógico, multiplexores y compuertas TTL y CMOS, entre otros.

- Computadora personal ejecutando software de manejo del ADRF.

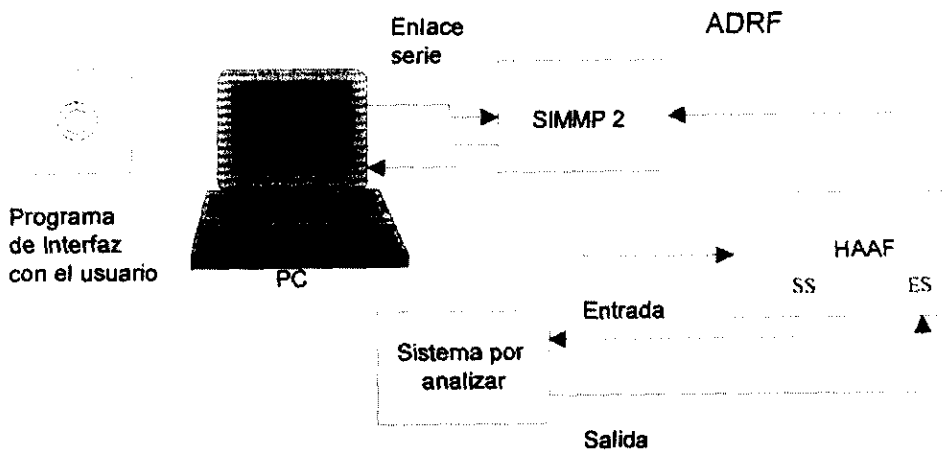


Figura 1. Diagrama de bloques del sistema ADRF.

Los pasos a seguir para obtener las curvas de respuesta en frecuencia se enumeran a continuación:

1.- Conectar la salida (SS) del HAAF con la entrada del sistema por analizar.

2.- Conectar la salida del sistema por analizar a la terminal (ES) del HAAF.

3.- Habilitar el ADRF ejecutando el programa de manejo en la PC.

4.- Al emplear la Interfaz gráfica del usuario efectuar lo siguiente:

a) Especificar la frecuencia inicial del intervalo de interés.
b) Especificar la frecuencia final del intervalo de interés.
c) Oprimir el botón analizar.

5.- Esperar a que el ADRF despliegue las curvas de magnitud y de fase para el intervalo de interés.

A continuación se describe la composición de los bloques mostrados en la figura 1.

1.1. Computadora monotablilla SIMMP- 2

La computadora monotablilla SIMMP-2 es una tarjeta basada en el microcontrolador 68HC11F1 de Motorola. Fue diseñada en el Departamento de Control de la Facultad de Ingeniería para propósitos didácticos y de desarrollo de aplicaciones de instrumentación y/o control. La SIMMP-2 incorpora un microcontrolador 68HC11F1 [1].

El microcontrolador seleccionado se programa con lenguaje ensamblador de 8 bits, su arquitectura es Von – Newman y es de tipo CISC (Complex Instruction Set Computer). Posee un convertidor analógico a digital de 8 canales, cada uno de 8 bits de resolución, líneas de entrada, salida y bidireccionales, memorias RAM y EEPROM, un puerto serie, además la versión F1 en particular fue diseñada para trabajar en modo expandido.

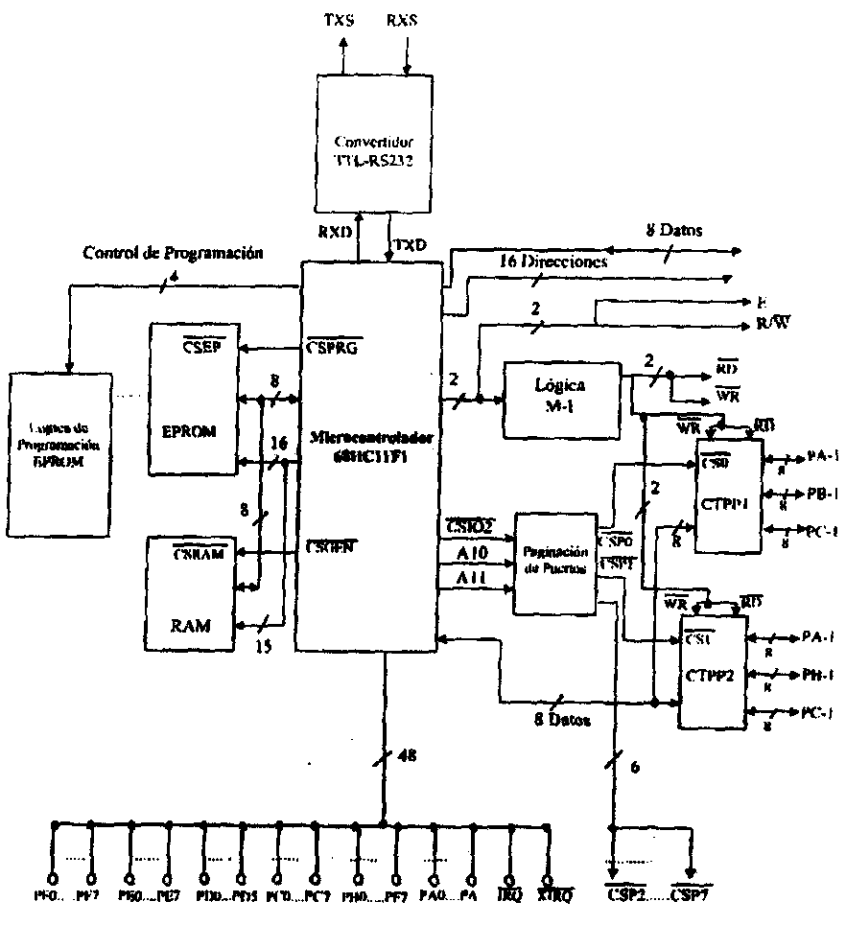


Figura 2. Diagrama de bloques de la tarjeta SIMMP -2.

La tarjeta SIMMP2 está diseñada de modo que el microcontrolador 68HC11F1 de la misma puede operar en cualquiera de los cuatro modos asociados con este dispositivo [1], para la aplicación aquí descrita el modo de operación es el expandido. En la Figura 2, se muestra un diagrama de bloques de la tarjeta SIMMP-2. Para mayores detalles técnicos acerca de ella puede consultarse el apéndice A de este trabajo.

La función que cumple la tarjeta SIMMP2 dentro del ADRF, es servir como enlace entre el programa interactivo que se ejecuta en la computadora personal y el módulo HAAF de administración de funciones. La tarjeta aloja en la memoria EEPROM del microcontrolador un programa con rutinas de comunicación básicas (Talker). Este programa permite al microcontrolador recibir instrucciones mediante el puerto serie para su inmediata ejecución, así la tarjeta SIMMP2 es capaz de actuar sobre el módulo HAAF a partir de las instrucciones que el usuario ingrese a la computadora.

1.2. Hardware de Adquisición y Arbitraje de Funciones (HAAF)

El bloque HAAF es un conjunto de circuitos integrados y elementos discretos. A su vez este bloque se divide en dos subcircuitos. El primero de ellos sirve para medir la respuesta de amplitud, mientras que el segundo brinda una estimación de la fase.

1.2.1 Circuito Analizador de Respuesta de Amplitud

El analizador de respuesta de amplitud tiene como núcleo un circuito integrado XR2206 [2] de EXAR. Este "chip" es un generador de funciones monolítico, configurado para producir una señal senoidal, que se utiliza como entrada al sistema por analizar (terminal SS). La señal mencionada posee dos características importantes que son :

- La amplitud es un valor fijo (2 Vpp).
- El valor de la frecuencia puede ser modificado

La segunda característica se debe a que el XR2206 posee un oscilador de relajación, cuya operación está regida por un circuito RC.

El proceso de medición de la respuesta del sistema se efectúa de la siguiente manera:

- En la salida SS se genera una señal senoidal colocando los valores adecuados de resistencia y capacitancia en un circuito RC, a partir de las instrucciones que le envía la computadora. Para esta función se emplean puertos de comunicación adicionales a la tarjeta SIMMP2 y que forman parte del bloque HAAF. La señal SS se conecta a la entrada del sistema por analizar, generando así la señal ES a la salida del mismo.

- Se detectan los valores pico de la amplitud de las señales SS y ES con un bloque de circuitería para cada señal. Cada valor de voltaje se envía directamente a un canal del convertidor analógico a digital del microcontrolador.

- Se realiza la conversión A/D y los resultados se envían a la PC en forma de arreglos de valores por el puerto serie para su procesamiento y despliegue.

El procedimiento antes descrito debe realizarse para cada valor de frecuencia para la que se desee obtener la magnitud y la fase correspondiente. Para hacer automático el registro de valores de respuesta sobre un intervalo de frecuencias, se implementó un barrido de valores de la misma; mediante un circuito convertidor digital a analógico (DAC), cuya salida se conectó a una terminal de voltaje de control del VCO empleado (XR2206).

El proceso del párrafo anterior complementa el uso de los puertos que controlan los valores de resistencia y capacitancia, que en conjunto determinan la frecuencia de la señal de salida del ADRF.

La figura 3 muestra un diagrama de bloques del analizador de respuesta de amplitud. Cada bloque del sistema será explicado con mayor detalle en capítulos siguientes.

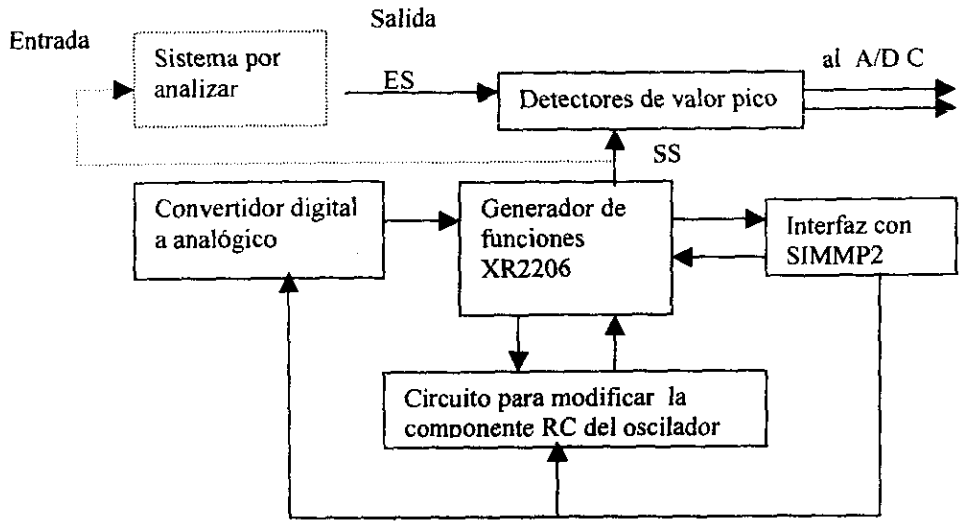


Figura 3. Circuito analizador de la respuesta de amplitud.

1.2.2 Circuito estimador de fase.

El estimador de fase está constituido por dos circuitos comparadores, una compuerta lógica (XOR) y un filtro paso – bajas.

Los comparadores cuadratizan las señales ES y SS conservando la diferencia de fase entre ellas, posteriormente las salidas de ambos comparadores se operan lógicamente para obtener una secuencia de pulsos de anchura proporcional a dicha diferencia. Finalmente la señal es ingresada a un filtro paso – bajas de noveno orden, obteniéndose así una señal de voltaje proporcional a la diferencia de fase entre las dos señales (SF). Ésta se introduce a un canal del convertidor analógico a digital para obtener su equivalente binario y se envía por el puerto serie a la computadora para su proceso y despliegue en pantalla.

La función de control de la tarjeta SIMMP2 en esta etapa se limita a las funciones ya descritas para generar la señal SS, al uso de un de los canales del convertidor analógico a digital y al envío de la información a la computadora vía enlace serie.

La figura 4 muestra en un diagrama de bloques la estructura del analizador de fase, así como las etapas de la señal SF.

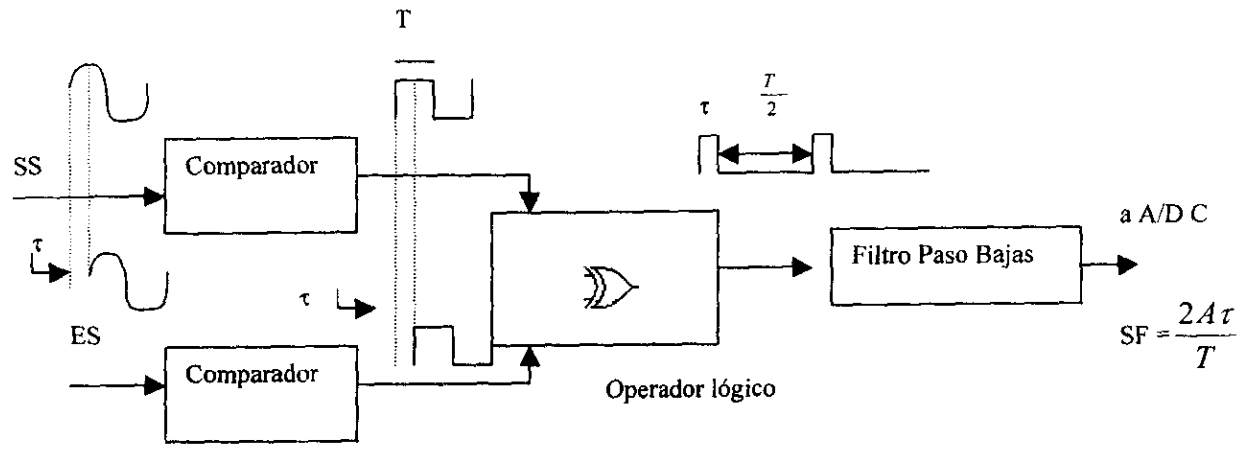


Figura 4. Estimador de fase.

1.3. Software de interfaz con el usuario. IGU

Para el desarrollo de la interfaz gráfica con el usuario (IGU) se empleó VisualBasic versión profesional 4.0 de 32 bits. La versión final de la misma puede ser ejecutada en una computadora que posea el sistema Windows 95 o 98. Pero ésta puede transformarse en una aplicación de 16 bits compatible con versiones de Windows 3.1 y 3.11 [3].

El programa de manejo del ADRF despliega una pantalla como la que se muestra en la figura 5. El usuario cuenta con cuadros de texto y botones de ingreso de datos y comandos.

Una vez que el usuario ha ingresado los datos apropiadamente, el programa realiza el procesamiento de éstos, generando las instrucciones adecuadas para el microcontrolador, para posteriormente enviarlas en forma de cadenas vía el puerto serie. Además, recibe la información proveniente del HAAF, la procesa y la despliega de manera gráfica en la pantalla.

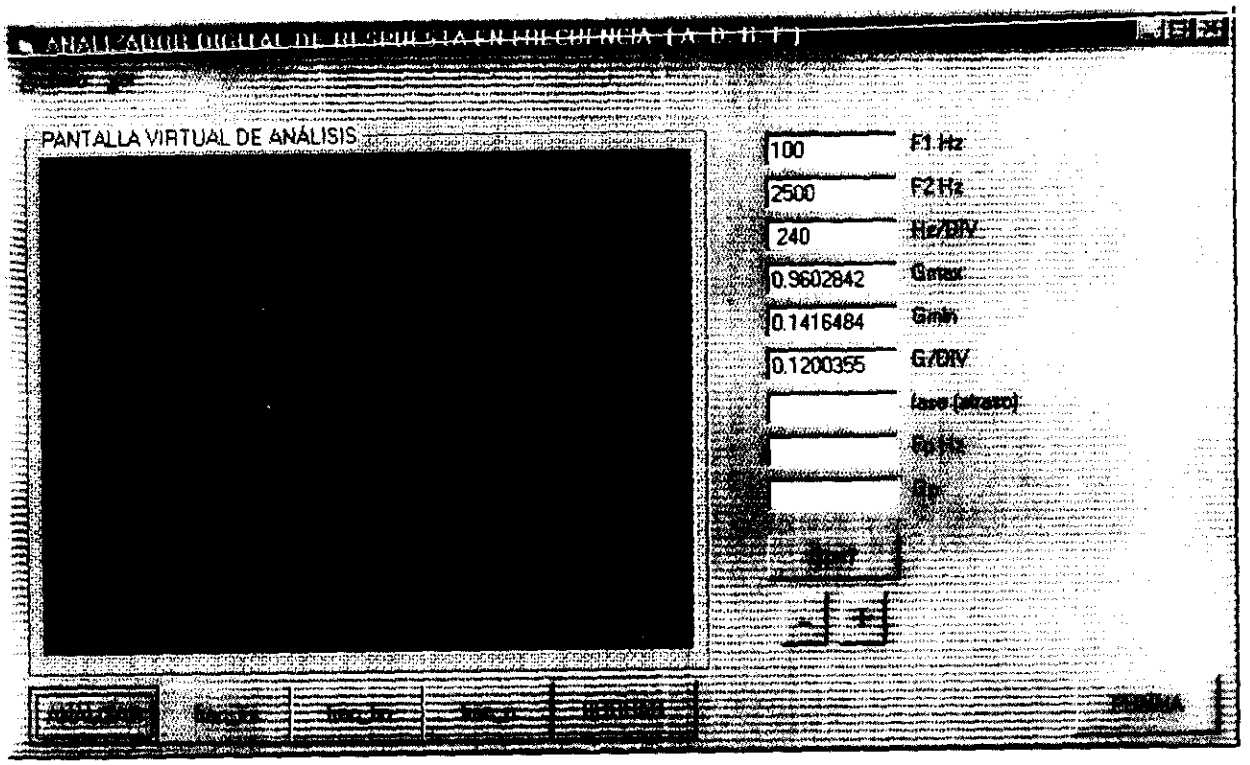


Figura 5. Pantalla de la IGU.

A continuación se explica brevemente la función de las opciones mostradas en la pantalla del módulo IGU.

- **Botón frec_ini**, al ser pulsado, el sistema requiere del usuario la frecuencia inicial del intervalo por analizar, una vez dada, el valor de la misma aparece en el cuadro de texto con etiqueta **F1 Hz.**

- **Botón frec_fin**, al ser pulsado, el sistema pregunta por la frecuencia final del intervalo por analizar, una vez dada, el valor aparece en cuadro con etiqueta **F2 Hz**, mostrándose además los Hertz por división correspondientes en el cuadro de texto con etiqueta **Hz/DIV**.

- **Botón ANALIZAR**, al ser pulsado, se inicia el proceso de análisis, efectuándose esto para 175 puntos dentro del intervalo especificado. Al concluir esta operación aparecerán respectivamente la ganancia máxima, la ganancia mínima, la graduación

vertical y el valor aproximado de la fase en los cuadros de texto etiquetados: **Gmax**, **Gmin** y **G/DIV y fase [atraso]**.

- **Botón frec_p**, al ser pulsado, el sistema solicita el valor de una frecuencia puntual que el usuario desee sea presentada por el HAAF, una vez hecho esto el valor referido aparecerá en el cuadro de texto con etiqueta **Fp Hz**.

- **Botón +**, al ser pulsado, la frecuencia fijada por el botón anterior es incrementada para un ajuste fino de la misma, testificándose en el cuadro de texto **Fp Hz** el cambio correspondiente.

- **Botón -**, al ser pulsado, la frecuencia fijada por el botón frec_p es decrementada para un ajuste fino. desplegando en el cuadro de texto **Fp Hz** el cambio correspondiente.

- **Botón Gp=?**, al ser pulsado, el sistema determina la magnitud de la respuesta en frecuencia correspondiente al valor puntual fijado previamente, apareciendo el valor obtenido en el cuadro de texto etiquetado como **Gp.**

- **Botón BORRAR**, elimina la gráfica obtenida cuando esto se desee.

- **Botón PUMMA, restablece el firmware de comunicación a la tarjeta SIMMP-2, en el caso de que esto sea necesario.**

La figura 6 muestra un diagrama de flujo que describe el procedimiento seguido por el dispositivo para realizar un análisis sobre un intervalo de frecuencia, una vez que la IGU determinó y generó la información que será enviada al microcontrolador.

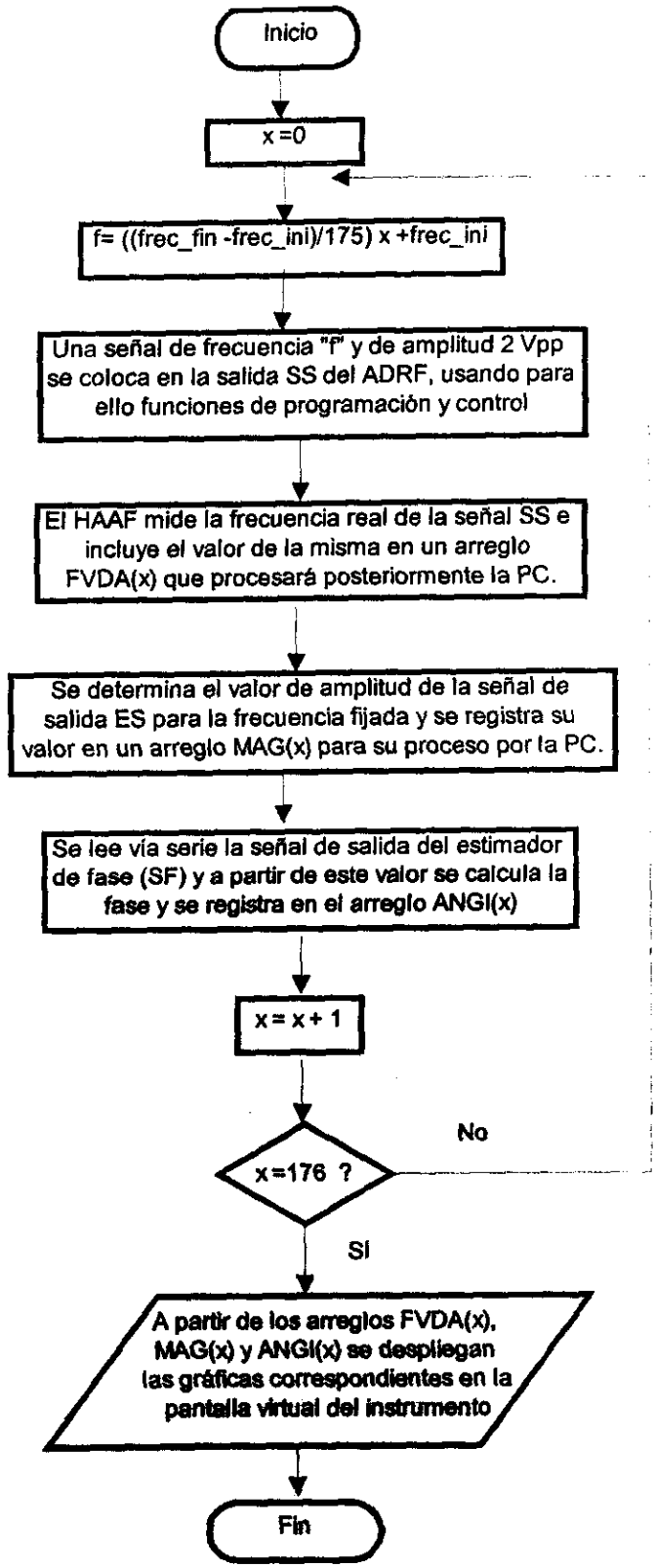


Figura 6. Secuencia de acciones realizadas por el ADRF para realizar un análisis.

Capítulo 2.

Diseño del sistema.

Descripción: *El presente capítulo describe la constitución de cada elemento que integra el módulo HAAF, presentando brevemente el procedimiento seguido para diseñar e interconectar los bloques de acuerdo a su función, además se muestran los procesos seguidos por la señales más importantes que el sistema utiliza. Se utiliza la nomenclatura descrita en el capítulo anterior.*

2.1 Generador de funciones y circuito convertidor digital a analógico.

Como se ha mencionado anteriormente, el circuito para generar una señal senoidal de referencia (SS) es el XR2206. Consultando la hoja de especificaciones del fabricante pueden conocerse las ecuaciones que rigen el comportamiento del mismo. De manera que puedan ser útiles al interconectar otros elementos.

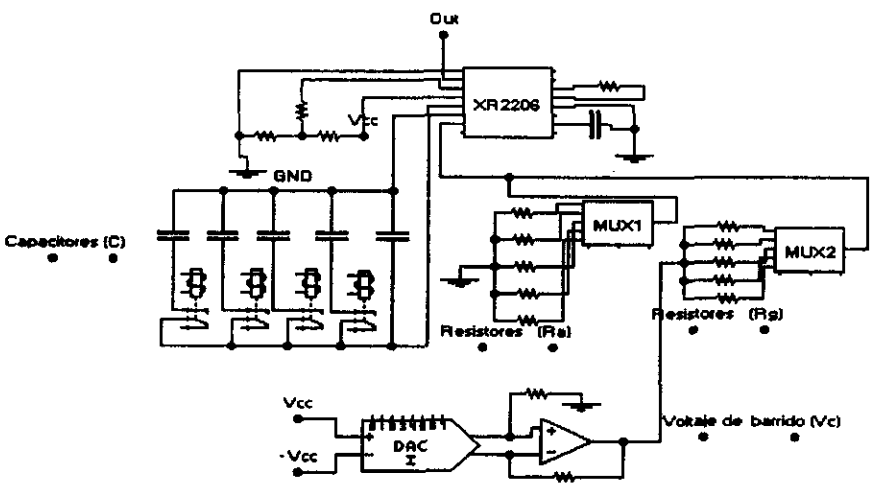


Figura 7. Circuito para generar una señal senoidal de frecuencia variable.

La Frecuencia que el dispositivo puede generar puede ser expresada por la siguiente ecuación: [2]

$$f = \frac{1}{RaC} + \frac{1}{RgC} - \frac{Vc}{3RgC} \qquad (2.1)$$

Donde Ra y Rg son arreglos de resistores (figura 7), de los cuales el microcontrolador de la tarjeta SIMMP2 puede seleccionar el valor adecuado a la frecuencia requerida; el primer sumando corresponde exclusivamente al oscilador de relajación, mientras que los dos últimos se producen cuando existe una entrada de voltaje de barrido de frecuencia Vc. Para el desarrollo aquí descrito el voltaje Vc es función de una palabra binaria de 8 bits, proporcionada por el convertidor analógico a digital DAC0800. La gráfica que se muestra en la figura 8 es la función que representa la salida del DAC, respecto a la palabra binaria de entrada [6].

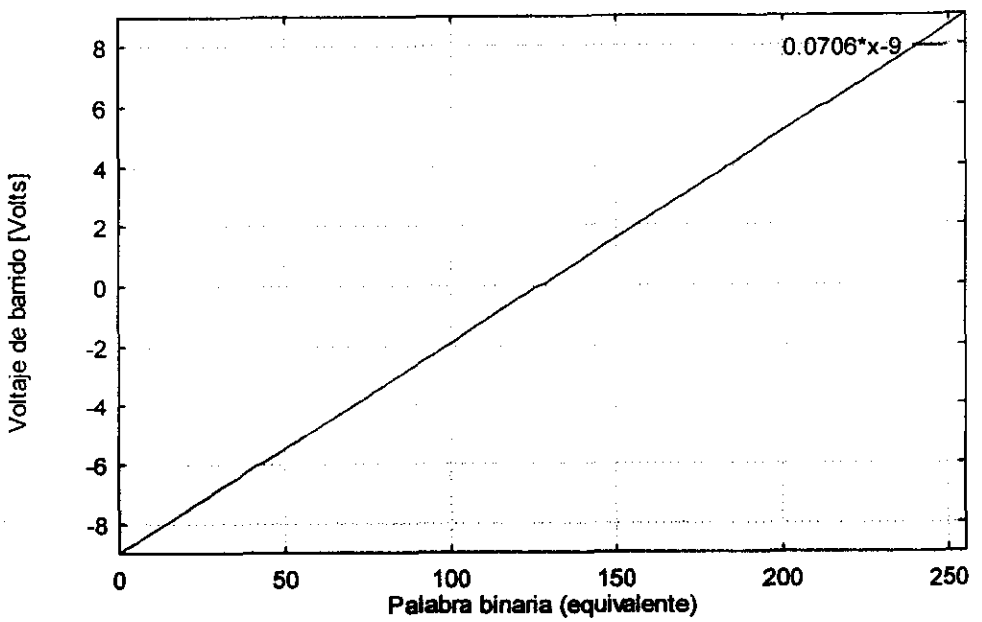


Figura 8. Gráfica del voltaje de barrido vs el valor de la palabra binaria.

Se observa que el valor de salida para la entrada igual a cero es –9 volts, mientras que para la entrada igual a255 (máxima con 8 bits), el valor de salida es de 9 volts. Encontrando la ecuación de la recta se encontró la siguiente ecuación:

$$V_c = \frac{18}{255} y - 9 \cdots \qquad (2.2)$$

Al combinar las ecuaciones 2.1 y 2.2 se obtiene:

$$f = \frac{1}{C}\left(\frac{1}{Ra} + \frac{1}{Rg}\left(4 - \frac{6}{255} y\right)\right) \cdots \qquad (2.3)$$

El DAC está limitado como ya se mencionó por los valores de entrada [0,255] y los valores de salida [-9,9]. Conocidos estos límites y sustituyendo los valores correspondientes en la ecuación 2.3 se encuentra que:

$$f_{MAX} = \frac{1}{C}\left(\frac{1}{Ra} + \frac{4}{Rg}\right)$$
$$f_{\min} = \frac{1}{C}\left(\frac{1}{Ra} - \frac{2}{Rg}\right) \quad \ldots \quad (2.4)$$

Si definimos un parámetro llamado relación de barrido como:

$$\alpha = \frac{f_{MAX}}{f_{\min}} = \frac{Rg + 4Ra}{Rg - 2Ra} \quad \ldots \quad (2.5)$$

Es fácil observar que si Rg > 2Ra, el circuito funcionará apropiadamente. Además, el conjunto de ecuaciones descritas como 2.4 y 2.5 , permiten elegir dos frecuencias de operación del circuito en función de elementos discretos como lo son los resistores y el capacitor. Puesto que el usuario elegirá las frecuencias de operación desde el módulo interactivo, dichos elementos serán las incógnitas de las ecuaciones, cuando se conoce la frecuencia deseada.

La ecuación denominada 2.5 tiene la particularidad de establecer la relación entre las frecuencias máxima y mínima de operación, en función exclusivamente de los resistores. Así que para una relación de la frecuencia máxima a la frecuencia mínima conocida, es fácil determinar los resistores requeridos.

Sean: $Ga = \frac{1}{Ra}$ y $Gg = \frac{1}{Rg}$ Las ecuaciones 2.4 pueden ser escritas como a continuación se muestra:

$$Cf_{MAX} = Ga + 4Gg$$
$$Cf_{\min} = Ga - 2Gg \quad \ldots \quad (2.6)$$

Resolviendo el sistema de ecuaciones se obtiene que:

$$Ga = \frac{C}{6}(4f_{\min} + 2f_{MAX})$$
$$Gg = \frac{C}{6}(f_{MAX} - f_{\min}) \quad \ldots \quad (2.7)$$

Estas ecuaciones son importantes cuando se requiere generar una señal que varíe su frecuencia desde 100 Hz a 1000 Hz, por ejemplo, ya que se conoce que la relación entre estas frecuencias es $\alpha = 10$, pueden encontrarse los valores de resistores indicados si se selecciona un valor de capacitancia previamente. Además si el capacitor se multiplica por

una constante las frecuencias final e inicial lo harán en la misma proporción y la relación α se mantiene sin cambios.
Este sencillo razonamiento nos lleva a pensar que conmutando el valor del capacitor en el arreglo, pueden lograrse escalamientos de frecuencia proporcionales al valor en el que la capacitancia se incrementó. Logrando así intervalos de frecuencia diferentes.

Para el desarrollo del ADRF se seleccionaron los valores de las frecuencias deseadas en intervalos, se seleccionaron arbitrariamente los capacitores y posteriormente se calcularon los resistores correspondientes con las ecuaciones 2.7. Los valores correspondientes se muestran en la siguiente tabla:

f_{min}	F_{MAX}	α	C [μF]	Ra [KΩ]	Rg [KΩ]
10	20	2	220	3.409	27.272
10	40	4	220	2.272	9.090
10	60	6	220	1.704	5.454
10	80	8	220	1.363	3.896
10	100	10	220	1.136	3.030
100	200	2	22	3.409	27.272
100	400	4	22	2.272	9.090
100	600	6	22	1.704	5.454
100	800	8	22	1.363	3.896
100	1000	10	22	1.136	3.030
1000	2000	2	2.2	3.409	27.272
1000	4000	4	2.2	2.272	9.090
1000	6000	6	2.2	1.704	5.454
1000	8000	8	2.2	1.363	3.896
1000	10000	10	2.2	1.136	3.030
10000	20000	2	0.22	3.409	27.272
10000	40000	4	0.22	2.272	9.090
10000	60000	6	0.22	1.704	5.454
10000	80000	8	0.22	1.363	3.896
10000	100000	10	0.22	1.136	3.030

Tabla 1. Valores seleccionados para operación del HAAF.

2.2 Puertos de salida agregados a la SIMMP2.

El siguiente paso del diseño lo constituye la automatización de los cambios tanto de los resistores, como del capacitor, al igual que el voltaje generado por el DAC. Para este propósito fue necesario acondicionar puertos de salida extra en la tarjeta SIMMP2. El circuito utilizado se muestra en seguida:

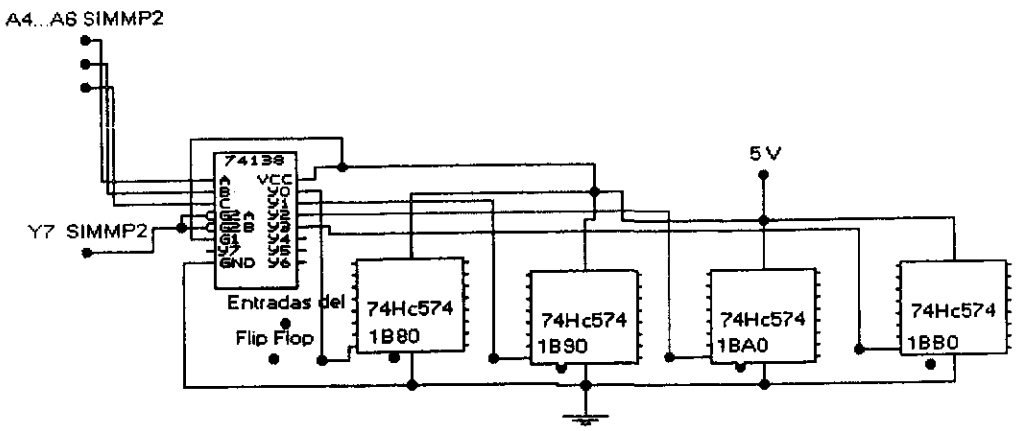


Figura 9. Circuito decodificador de direcciones de los puertos de salida

Los bloques señalados como 74HC574, corresponden a un circuito flip flop tipo D octal cuya parte inferior se encuentra marcada por la dirección con la que el microcontrolador hará referencia a cada puerto de salida. Las direcciones asignadas se decodifican a través de un circuito 74LS138, tomando las líneas Y7, A4, A5 y A6 de la tarjeta SIMMP2. Las entradas a cada circuito flip flop son las líneas del bus de datos D0 ... D7 de la SIMMP2.

Con esta implementación es posible, utilizando las líneas mencionadas, escribir valores binarios (o hexadecimales) en las salidas de cada puerto, contando con 8 de ellas. Estas salidas deben controlar el voltaje de salida del circuito DAC0800, el cambio de resistrores Ra y Rg y finalmente el cambio del capacitor C.

2.3 Interfaz entre los puertos y el circuito XR2206.

Como puede observarse en la figura 10, el cambio de resistores se implementó con un multiplexor analógico para cada arreglo (Ra, Rg), el cambio de capacitores se implementó con pequeños relevadores de 5 volts; mientras que la conexión con el DAC fue directamente de las salidas del puerto 74HC574 a las entradas del DAC0800. Se eligió el puerto en la dirección 1B80 para controlar este último.

Para realizar un barrido de voltaje en el intervalo [-9, 9] volts, se necesita una secuencia de niveles lógicos en el DAC0800, éstos son generados por el microcontrolador y transmitidos al circuito por medio del puerto de salida 1B80. Por lo que para obtener el valor del voltaje deseado en la salida del circuito DAC0800 solo debe escribirse un número hexadecimal en la dirección 1B80 [6].

Se eligió el puerto localizado en la dirección 1B90 para controlar el cambio en los capacitores, la conexión entre el puerto y los relevadores se hizo utilizando un buffer

digital (74LS05) y algunos diodos de protección. En este caso al presentar un nivel lógico alto, en la salida del puerto se habilita el capacitor, conectado a dicha salida, en el arreglo correspondiente.

Finalmente se escogió el puerto con dirección 1BA0 para controlar la conmutación de resistores, en este caso se utilizaron 2 multiplexores analógicos de 8 líneas a 1, como es conocido este tipo de circuitos se controlan por 3 líneas, en las que al escribir la adecuada combinación de valores, se selecciona una línea determinada. Solo fue necesario un puerto de salida para controlar ambos multiplexores, pues el 74HC574 cuenta con 8 líneas de salida. El circuito correspondiente a esta etapa se muestra a continuación:

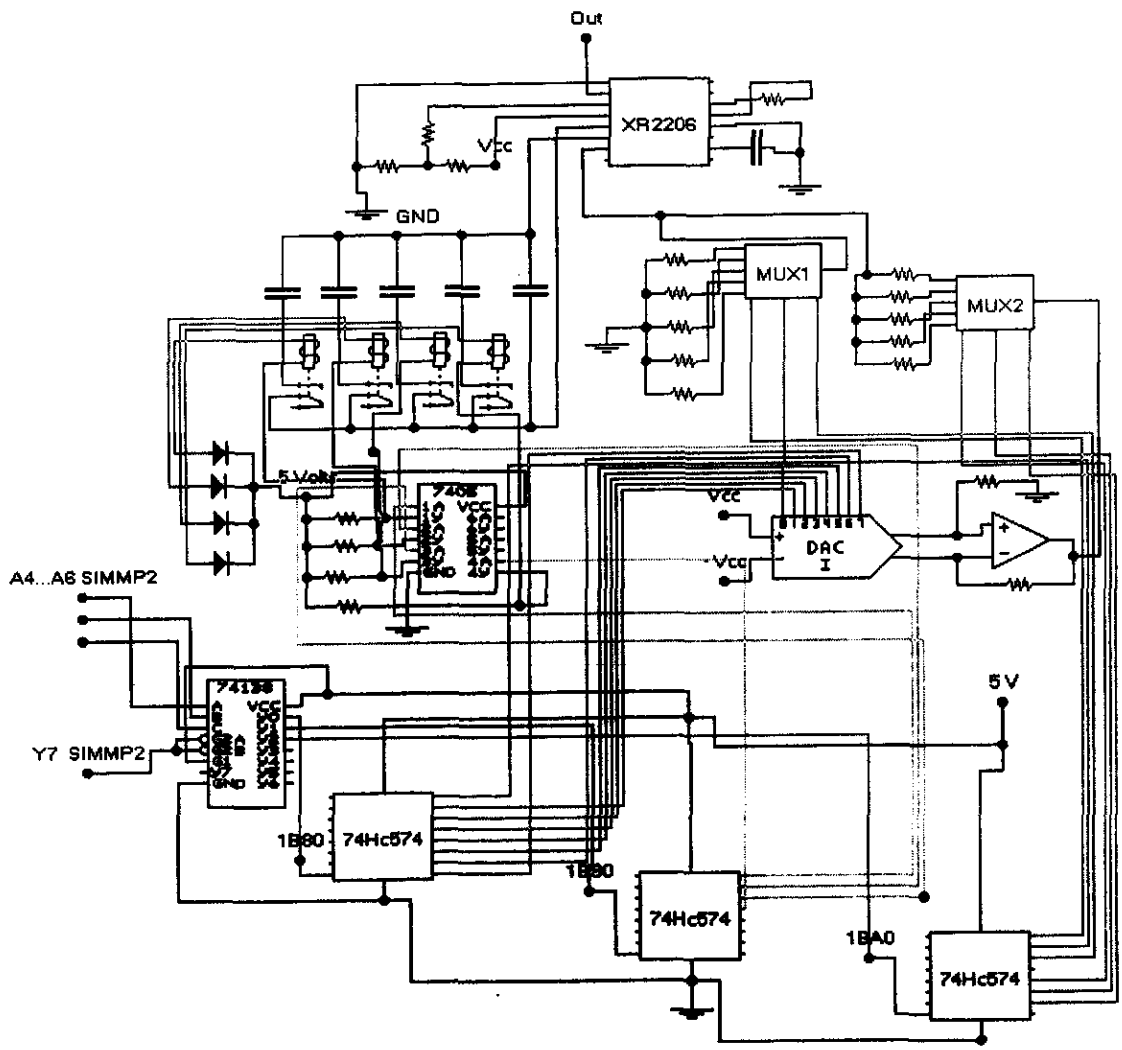


Figura 10 . Circuito interfaz

A continuación se muestra una tabla de los valores necesarios en cada dirección de control para lograr una salida determinada en el intervalo de frecuencias indicado:

						Direcciones		
Fmin	FMAX	α	C [μF]	Ra [KΩ]	Rg [KΩ]	1B80	1B90	1BA0
10	20	2	220	3.409	27.272	[FF , 00]	01	00
10	40	4	220	2.272	9.090	[FF , 00]	01	11
10	60	6	220	1.704	5.454	[FF , 00]	01	22
10	80	8	220	1.363	3.896	[FF , 00]	01	33
10	100	10	220	1.136	3.030	[FF , 00]	01	44
100	200	2	22	3.409	27.272	[FF , 00]	02	00
100	400	4	22	2.272	9.090	[FF , 00]	02	11
100	600	6	22	1.704	5.454	[FF , 00]	02	22
100	800	8	22	1.363	3.896	[FF , 00]	02	33
100	1000	10	22	1.136	3.030	[FF , 00]	02	44
1000	2000	2	2.2	3.409	27.272	[FF , 00]	04	00
1000	4000	4	2.2	2.272	9.090	[FF , 00]	04	11
1000	6000	6	2.2	1.704	5.454	[FF , 00]	04	22
1000	8000	8	2.2	1.363	3.896	[FF , 00]	04	33
1000	10000	10	2.2	1.136	3.030	[FF , 00]	04	44
10000	20000	2	0.22	3.409	27.272	[FF , 00]	08	00
10000	40000	4	0.22	2.272	9.090	[FF , 00]	08	11
10000	60000	6	0.22	1.704	5.454	[FF , 00]	08	22
10000	80000	8	0.22	1.363	3.896	[FF , 00]	08	33
10000	100000	10	0.22	1.136	3.030	[FF , 00]	08	44

Tabla 2. Distribución de intervalos de frecuencia, valores de componentes y direcciones.

Los valores en las direcciones son hexadecimales.
El intervalo en la columna 1B80, indica los valores para obtener los extremos mínimo y máximo en la escala delimitada por fmin y fMAX.

2.4 Detectores de valores pico y C A/D.

Otro bloque que constituye al HAAF, es el que forman en conjunto los detectores de valor pico que son alimentados por las señales de entrada y salida del HAAF, respectivamente. Estos circuitos registran el valor más alto que alcanzan las señales senoidales; posteriormente, al estar interconectados (cada uno) con la entrada de un canal del C A/D envían directamente a este último una señal de corriente directa que el convertidor interpretará en un equivalente hexadecimal.

En la figura 11, se muestra el bloque descrito anteriormente, los interruptores mostrados a la salida de la primera etapa, se implementaron con un relevador de 5 volts cada uno, al cerrarse el interruptor, se ocasiona la descarga inmediata del capacitor que guarda el valor pico momentáneamente.

La activación de dicho interruptor se hace a través de la tarjeta SIMMP2 en la dirección 1BC0.

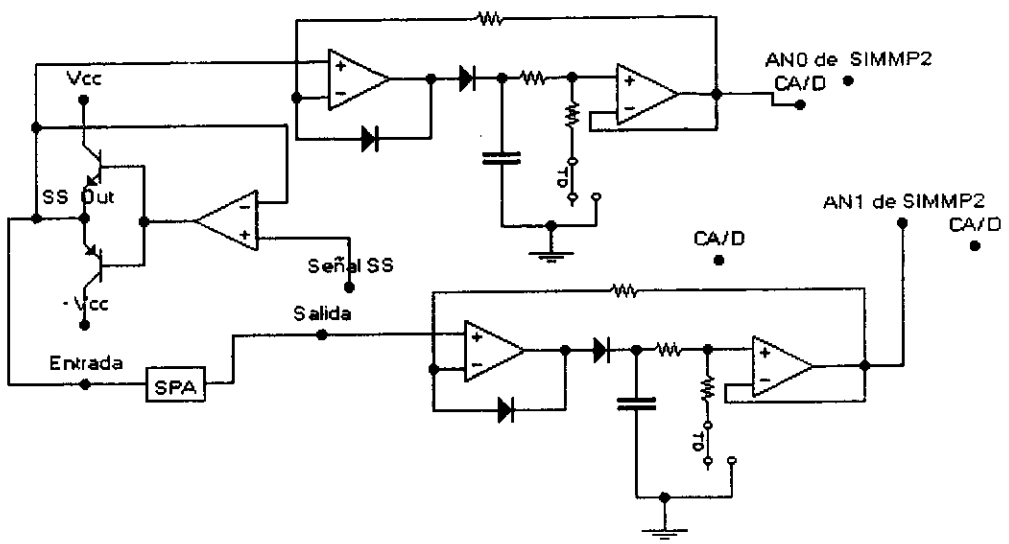


Figura 11. Circuitos detectores de valores pico.

2.5 Bloque estimador de frecuencia.

Este bloque se compone de 2 comparadores, una compuerta TTL XOR y un filtro pasobajas. Los comparadores convierten las señales senoidales de entrada y salida en señales cuadradas de 5 volts de amplitud. La señal cuadrada correspondiente a la señal de entrada, se introduce directamente a un canal del temporizador (IC1) del microcontrolador de la SIMMP2. La determinación de la frecuencia se obtiene contando los flancos de la señal durante un intervalo de 1 segundo; es decir con funciones de programación.

La señal de entrada SS, a su vez, se opera a través de una compuerta XOR con la señal de salida SE para obtener una señal de pulsos, cuya duración, es proporcional a la relación de fase entre ambas señales. La nueva señal pulsante es ingresada directamente a un filtro pasobajas, para obtener una señal de corriente directa que puede ser digitalizada por un canal de CA/D. El esquema correspondiente y el diseño del filtro se muestran a continuación:

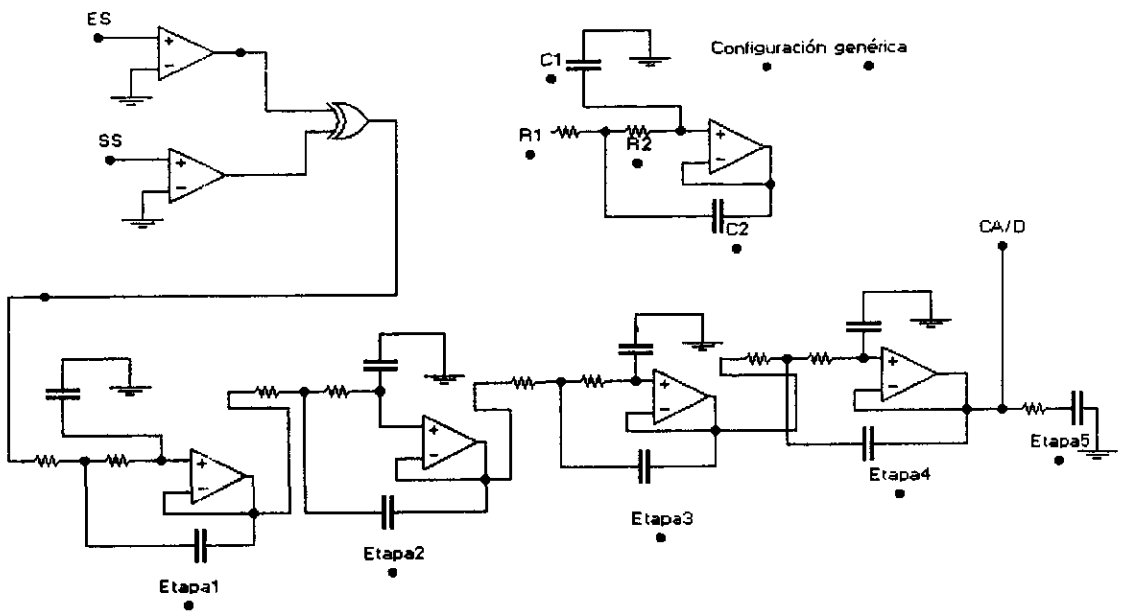


Figura 12. Circuito para medir la fase entre SS y SE

Las características deseadas para el filtro son:

Fcorte = 50Hz
Fsupresión = 100 Hz
Atenuación = 50dB

El cálculo del filtro se realizó para obtener un filtro pasobajas, Butterworth y configurado como un sistema de retroalimentación múltiple. Para encontrar de los valores de los componentes se empleó un programa de computadora. La configuración de retroalimentación múltiple se muestra bajo la leyenda de "Configuración genérica" en la parte superior izquierda de la figura 12. Esta configuración representa un filtro de segundo orden [7] ¡[8].

El orden adecuado para conseguir el comportamiento de atenuación deseado es 9. Razón por la cuál, se requirieron de 4 etapas de segundo orden y una de primero. Los valores seleccionados de capacitores y calculados de resistores se muestran a continuación:

Etapa 1:	Etapa 2:	Etapa 3:	Etapa 4:	Etapa 5:
R1 = 11599 Ω	R1 = 3587 Ω	R1 = 2174 Ω	R1 = 590 Ω	R = 318309 Ω
R2 = 43674 Ω	R2 = 28243 Ω	R2 = 46593 Ω	R2 = 98696 Ω	C = 1 μF
C1 = 1 μF	C1 = 0.1 μF	C1 = 0.1 μF	C1 = 0.1 μF	
C2 = 0.02 μF	C2 = 1 μF	C2 = 1 μF	C2 = 1 μF	

La figura 13 muestra gráficamente los pasos seguidos par obtener la estimación de la fase.

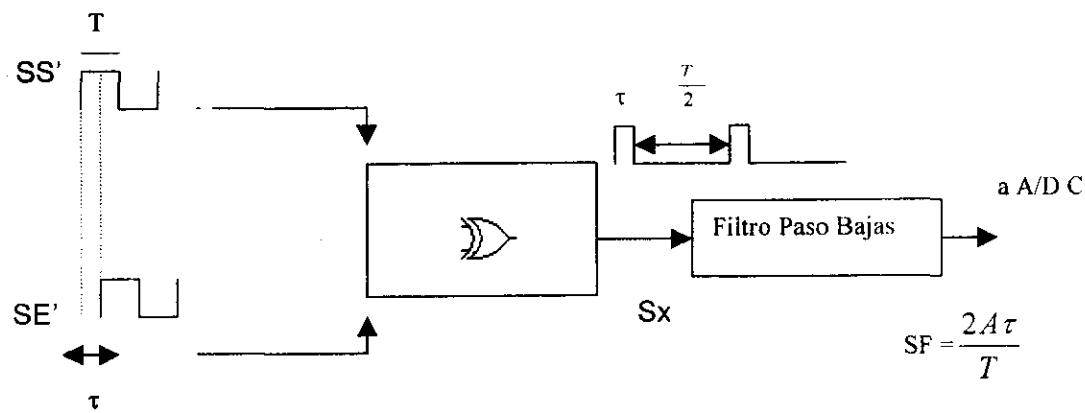


Figura 13 Procesos seguidos para estimar la fase.

Las señales etiquetadas como SS' y SE', corresponden al las señales de entrada y salida al sistema por analizar tras pasar por el bloque de comparación respectivo. La diferencia de fase entre las señales se representa por τ y el periodo de la señal se representa con T. La compuerta XOR, obtiene una señal con periodo igual a T/2, etiquetada como Sx. El ancho de los pulsos obtenidos está dado por la diferencia de fases τ, pues solo en este intervalo la función XOR se verifica alta.

La señal Sx ingresa al filtro para obtenerse la señal SF, que está conectada directamente a la entrada del

La señal SF cumple: $SF = \frac{A\tau}{\frac{T}{2}}$,donde A es un valor constante.

Anteriormente se mencionó que la señales se adecuaban a niveles TTL, por lo que en en este caso la constante A es aproximadamente 5v. La fase se obtiene por medio de un cálculo en la computadora con la siguiente expresión:

$$SF = \frac{2A\tau}{T}$$

Capítulo 3.

Software: Rutinas en lenguaje ensamblador y Programa IGU.

Descripción: El presente capítulo describe los programas hechos en lenguaje ensamblador que controlan la ejecución de acciones del ADRF. Estos programas como ya se mencionó son enviados por el puerto serie de la PC. Los listados de cada programa incluyen además un diagrama de flujo esquemático que ilustra de manera gráfica la función de cada uno.

Finalmente se incluye la descripción del programa que se ejecuta desde la PC, dicho programa se denomina Interfaz Gráfica de Usuario (IGU) y se encarga de administrar el sistema, presentando un panel de control desde el cual, el usuario introduce los parámetros de análisis con los que el dispositivo realizará su función.

3.1 Rutinas en lenguaje ensamblador para control del HAAF.

Las rutinas que se describirán a continuación, fueron programadas con ayuda del ensamblador de la compañía P&E Microcomputer Systems IASM11. Dichas rutinas están escritas en el formato que el microcontrolador HC11 puede ejecutar, sirviendo de esta manera como medio de comunicación entre la PC y el módulo HAAF.

El control de las acciones del dispositivo se hacen con ayuda de la comunicación serie de la PC. La tarjeta SIMMP2 ejecuta las acciones que recibe por este medio de parte de la computadora. Estas instrucciones para poder ser interpretadas por el microcontrolador deben ser escritas en el lenguaje ensamblador propio del circuito como se mencionó antes.

A continuación se muestran los listados correspondientes a las acciones básicas realizadas por el ADRF el nombre de cada archivo aparece en la primera línea del listado. La instrucción `jmp $0000` regresa el flujo de ejecución al programa monitor, en espera del siguiente comando a ejecutarse. En este programa los valores asignados (AA, BB, CC) son "mudos", pues su función es únicamente para señalar el byte correspondiente a un valor que será cambiado dinámicamente por el programa IGU.

3.1.1 Programa que especifica las direcciones de control y sus valores iniciales.

Ponefrec.asm

```
org  $c000
     ldaa #$aa
     staa $1b80
     ldaa #$bb
     staa $1b90
     ldaa #$cc
```

```
staa $1ba0
jmp  $0000
```

El diagrama de flujo correspondiente se muestra a continuación.

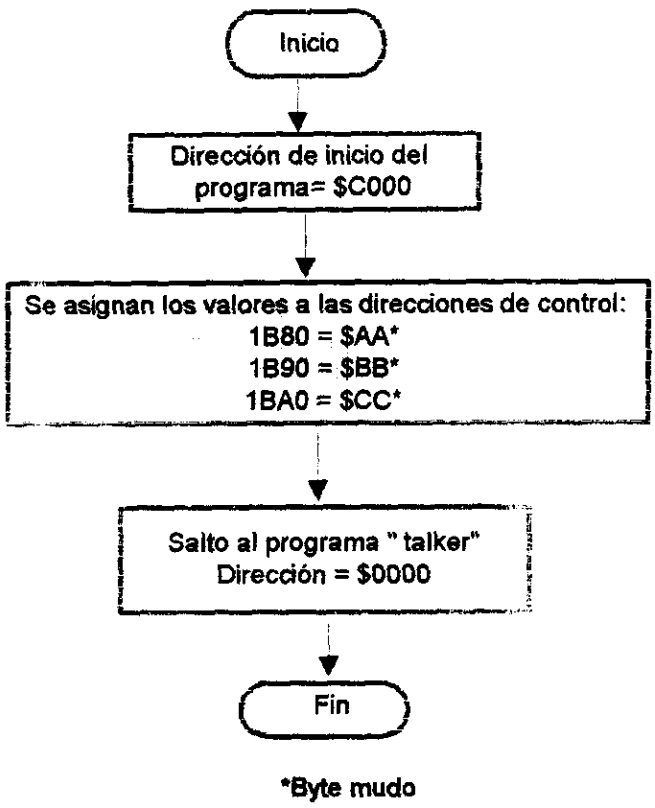


Figura 14 . Diagrama de flujo para el programa 3.1.1

Este programa es transmitido en cuanto se ejecuta el programa de manejo que corre en la PC. Su función principal es colocar en los puertos adecuados, los valores iniciales requeridos por el HAFF para generar una señal senoidal. El salto final a la dirección $0000 es para regresar al programa monitor en espera de otra instrucción.

3.1.2 Programa que realiza una medición puntual de las magnitudes de la señales de entrada y salida.

Midemag.asm

```
org $c000
        ldaa #$01
        staa $1bc0 ;activa detectores de pico de entrada y salida
        bsr ret    ;espera 100ms antes de hacer la medición.
        ldaa #$90
        staa $1039
inicony: ldaa #$10
        staa $1030
termino: ldaa $1030
        anda #$80  ;fin de conversión
        beq termino
        ldab $1031
        jsr $50    ;transmite byte entrada
```

```
        ldab $1032
        jsr $50     ;transmite byte de salida
        clr $1bc0   ;desactiva detectores de pico de entrada y salida
        jmp $0000
ret:    pshx
        ldx #$6184
vuelta: nop
        dex
        bne vuelta
        pulx
        rts
```

El diagrama de flujo se muestra a continuación:

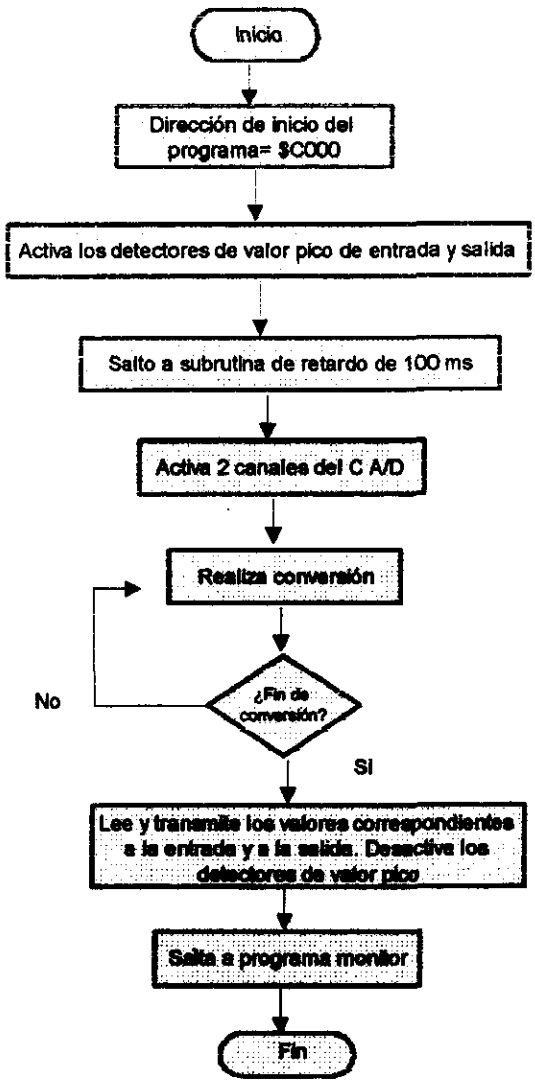


Figura 15. Diagrama de flujo para el programa 3.1.2

Este programa activa los elementos que se encuentran en la dirección 1BC0, correspondientes a los detectores de valor pico de entrada y salida, espera 100 ms antes de activar dos de los canales del convertidor analógico a digital (escribiendo el valor $90 en la dirección $1039), para posteriormente realizar la conversión. Una vez finalizada ésta, los valores correspondientes se toman de los registros correspondientes (direcciones $1031 y $1032) y se transmiten usando el puerto serie, uno después de otro. Finalmente el programa regresa al programa monitor en espera de la siguiente instrucción.

3.1.3 Programa que realiza una medición puntual de la frecuencia de la señal de entrada al sistema por analizar (SS).

Midefr.asm

```
org $c000
          bra inicio
contador: fcb $00,$00
tespp:    fcb $FF
tesfin:   fcb $FF
contb100: fcb $00
inicio:   ldaa #$7e
          staa $dc
          staa $e8
          ldx #rutserv
          stx $e9
          ldx #servoc2
          stx $dd
          ldaa #$10
          staa $1021      ;habilita captura por flanco de subida
          ldaa #$40
          staa $1023
          staa $1022      ;habilita interrupci¢n oc2
          cli
nofin:    ldaa tesfin
          bne nofin
          ldab contador
          jsr $50         ;transmite byte alto de frecuencia
          ldab contador+1
          jsr $50         ;transmite byte bajo de frecuencia
          sei
          jmp $0000
rutserv:  ldx contador
          inx
          stx contador
          ldaa #$04
          staa $1023
          rti
servoc2:  ldd $1018
          addd #$4e20
          std $1018
          ldaa #$40
          staa $1023
          ldaa tespp
          bne nopp
          clr tespp
          ldaa #$44
          staa $1022      ;habilita interrupci¢n de captura
nopp:     inc contb100
          ldaa contb100
```

```
            cmpa #$64
            bne salida
            ldaa #$40
            staa $1022      ;deshabilita interrupci¢n de captura
            clr tesfin
salida:     rti
```

Diagrama de flujo.

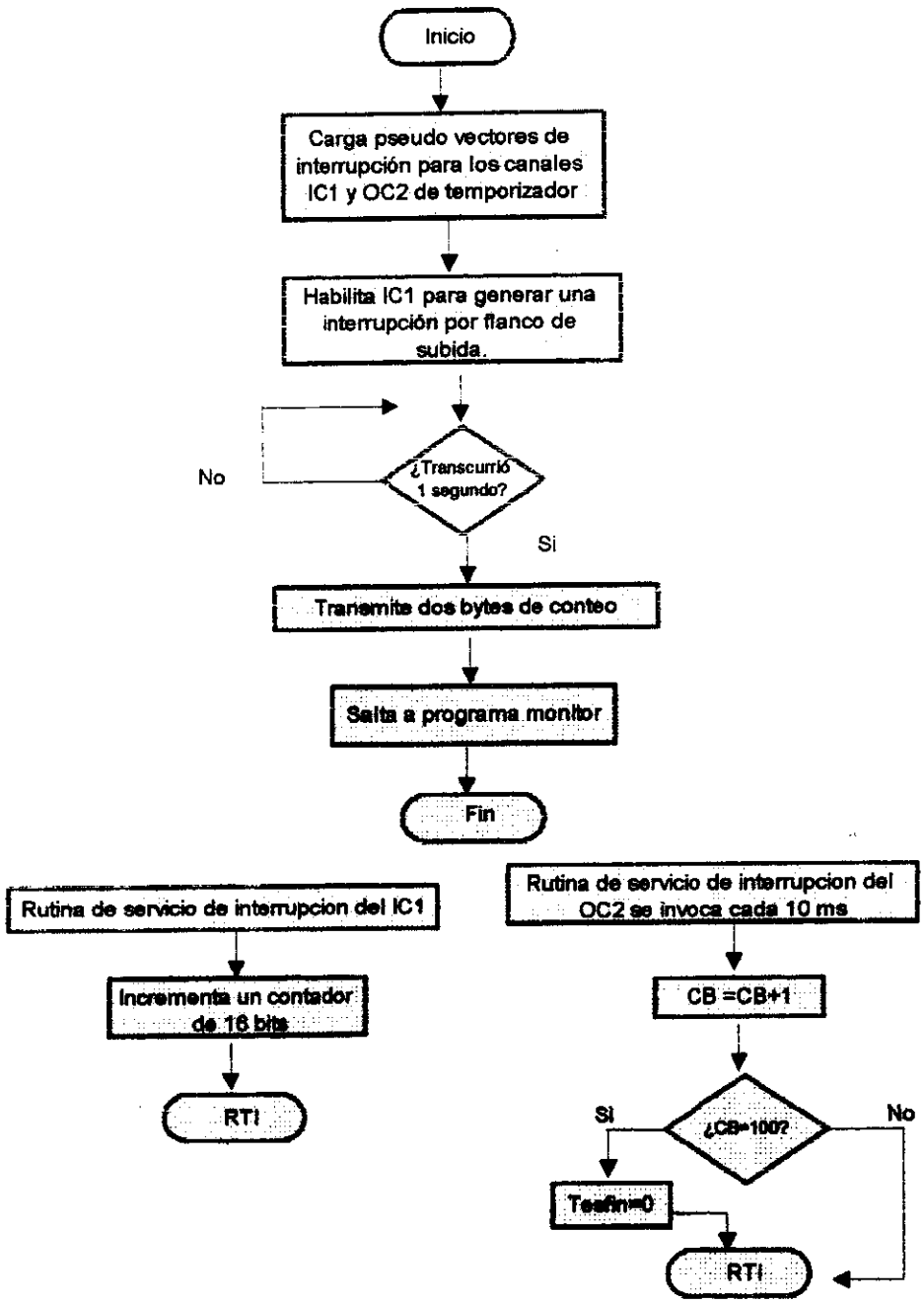


Figura 16 . Diagrama de flujo para el programa 3.1.3

El programa anterior desempeña una de las funciones más importantes del sistema, pues determina la frecuencia que en el momento de ejecutarse, se está generando por el XR2206. Cuando este programa se ejecuta, realiza una interrupción cada 10 ms, incrementando un contador; además cada vez que se detecta un flanco de subida en la terminal del IC1, se incrementa otro contador; en esta misma rutina se verifica el valor del contador de ms, si la cuenta de éste es 100, se transmiten los valores de ambos contadores a la PC. El objetivo de estas acciones, es entonces determinar cuantos flancos de la señal pueden registrarse en un segundo. La operación final se realiza en la PC.

3.1.4 Programa que realiza una medición puntual del ángulo entre la señal de entrada (SS) y la señal de salida (ES).

Midefase.asm

```
org $c000
            LDAA #$90
            STAA $1039
CONV:       LDAA #$02
            STAA $1030
NOFIN:      LDAA $1030
            ANDA #$80
            BEQ NOFIN
            LDAB $1031
            JSR $50
            JMP $0000
```

El diagrama de flujo es:

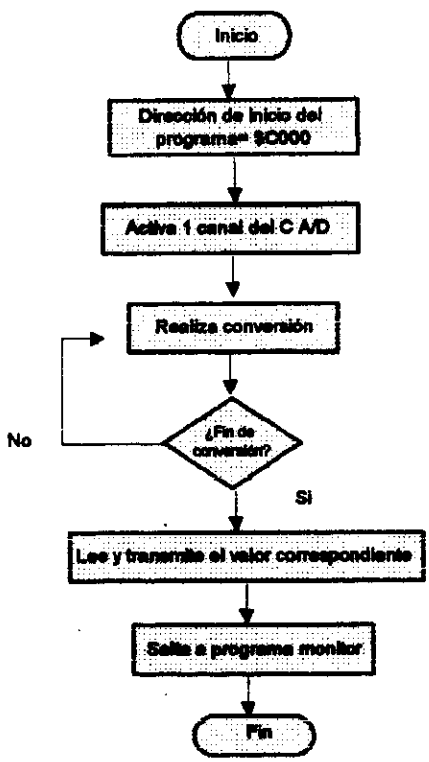


Figura 17 . Diagrama de flujo para el programa 3.1.4

Este programa activa uno de los canales del convertidor analógico a digital (escribiendo el valor $90 en la dirección $1039), para posteriormente realizar la conversión. Una vez finalizada ésta, el valor correspondiente se toma del registro de datos (dirección $1031) y se transmite usando el puerto serie. Finalmente el programa regresa al programa monitor en espera de la siguiente instrucción. Debe recordarse que el valor leído corresponde a un valor de voltaje proporcional a la fase entre las señales.

3.1.5 *Programa que determina el signo del ángulo entre la señal de entrada* (SS) y la señal de salida (ES).

Ang.asm

```
org $c000
            LDAA #$7E
            STAA $E8
            LDX #SERV
            STX $E9
            CLR $1001
            LDAA #$04
            STAA $1022;HABILITA INTERRUPCIàN IC1
            STAA $1023;BORRA BANDERA IC1F
            LDAA #$10
            STAA $1021;TCTL2 PARA APTURA DE FLANCOS DE SUBIDA
            CLI
HHHH:       LDAA $1022
            ANDA #$04
            BNE HHHH
            SEI
            JMP $0000
SERV:       LDAB $1000
            ANDB #$02
            JSR $50;TRANSMITE DATO; $00 SI ATRASO, $02 SI ADELANTO.
            LDAA #$04
            STAA $1023
            CLR $1022
            RTI
```

Este programa verifica, cual de las señales, ya sea la de entrada o la de salida, registra el primer flanco de subida, esto determina si la señal de salida está adelantada con respecto a la de la entrada o viceversa. El valor se enmascara con $04, esto es: se realiza la operación AND de este valor con el obtenido de la terminal IC2. por la posición que el dato proveniente del registro de esta terminal, el resultado es $00 si la señal de salida está atrasada con respecto a la de entrada, ell resultado es 02, si es el caso contrario. En cualquiera de los casos se transmite el resultado a la PC.

El diagrama de flujo correspondiente se muestra a continuación en la figura 18.

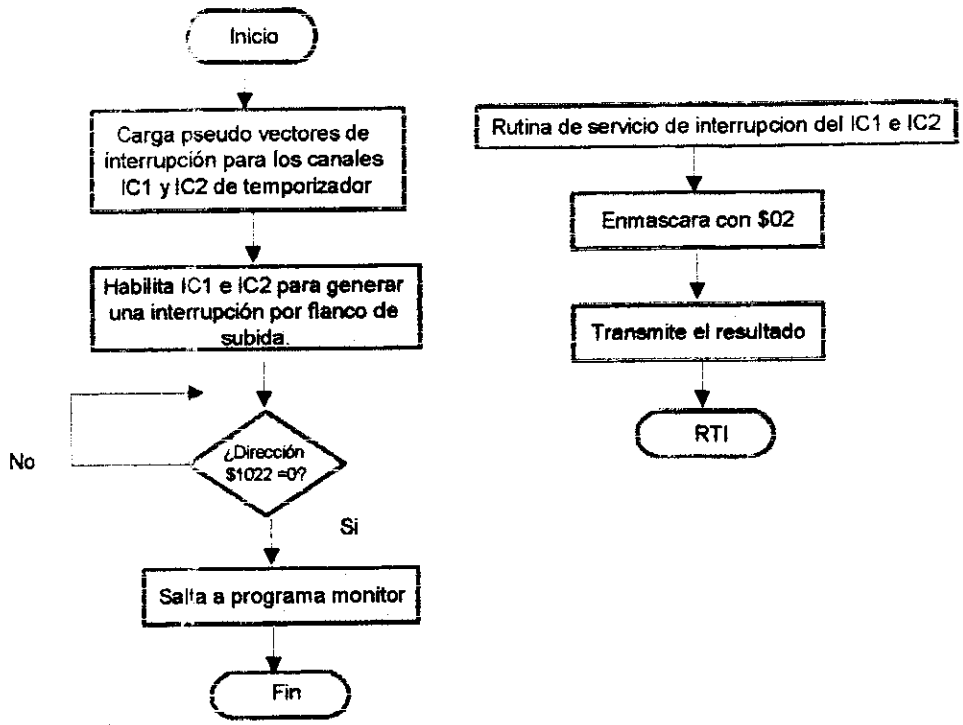


Figura 18 . Diagrama de flujo para el programa 3.1.5

Los programas anteriormente listados, son los códigos fuente de las listas binarias que recibe la tarjeta SIMMP2, la correspondiente lista, tendrá el mismo nombre que el programa fuente, pero la extensión será **.blm**.

3.2 Interfaz Gráfica del Usuario

La Interfaz Gráfica del Usuario (IGU), es un programa de computadora escrito en Visual Basic versión 4.0. Este compilador del lenguaje BASIC, fue seleccionado por su capacidad para generar aplicaciones sobre plataforma WINDOWS. La variante del programa usado, permite que la aplicación desarrollada se ejecute en las versiones del sistema WINDOWS 95 y superiores.

Se escogió trabajar con la plataforma Windows, por ser un ambiente común para los usuarios de computadoras a la vez que sencillo y que no requiere de muchos conocimientos previos a su uso.

La función que cumple la IGU, es la de proporcionar al usuario del sistema un medio para comunicarse con el dispositivo físico. Este medio consta de un panel de ingreso de datos, con opciones y menúes en los cuales se registran y se escogen los valores y las opciones de operación del ADRF. El IGU, se encargará de interpretar los datos selecionados e ingresados, en instrucciones que el HAAF ejecutara, para después enviar los resultados de manera gráfica en el panel de control.

El panel principal de la IGU se muestra a continuación:

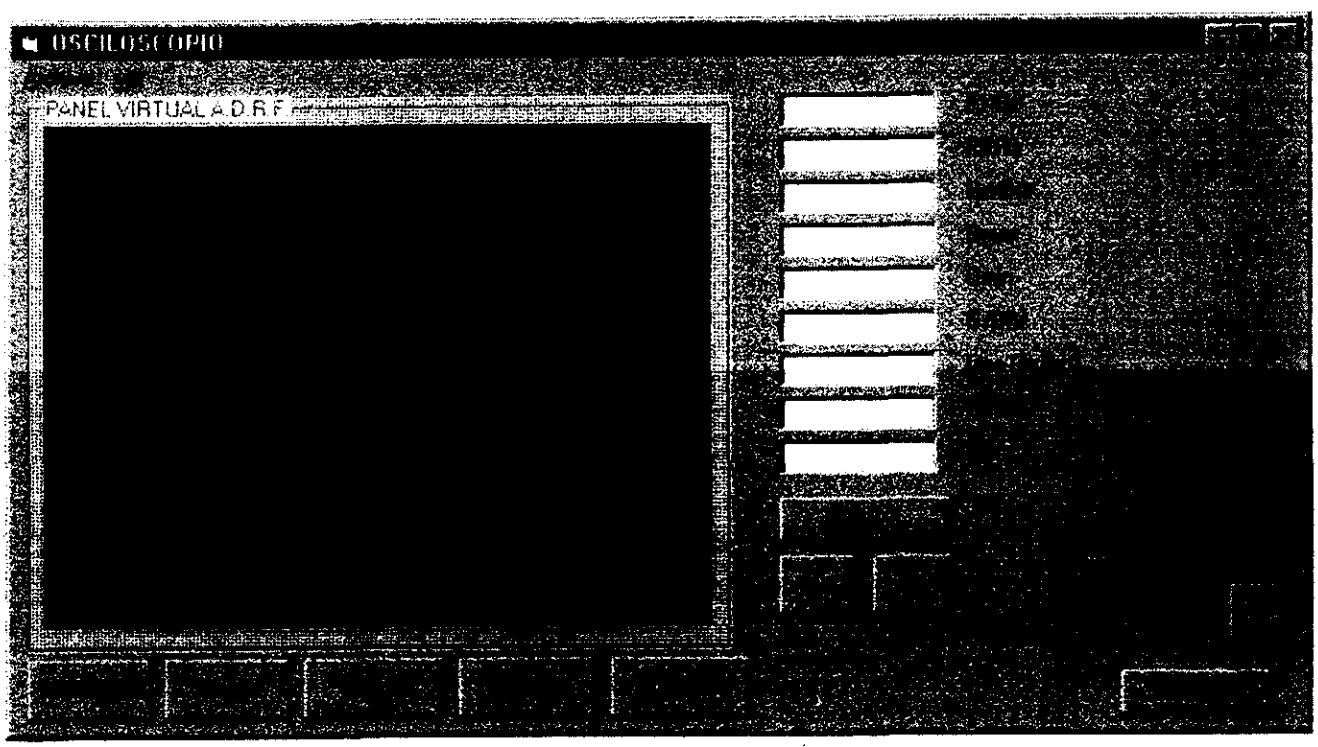


Figura 19 . Panel principal de la IGU

Los botones y sus funciones fueron descritos con anterioridad. En este capítulo se describirá la función programada que se invoca en cada caso, además de las propiedades de los elementos que componene el programa.

3.2.1 Forma principal del programa.

Esta sección del programa se ejecuta cada vez que el mismo corre por primera vez en la PC, es decir cada vez que se " carga " la forma principal. Las acciones iniciales son ajustarse a un tamaño determinado (durante el diseño del programa), indicar a la PC cual será el directorio de trabajo y configurar el puerto serie para trabajar con la SIMMP2. En este proceso la forma principal adopta la apariencia que se mostró en la figura 19.

La configuración del puerto es para trabajar con 1200 bps, en el puerto COM2. Después de esta configuración se genera un evento `Command3_Click` que envía por puerto serie el programa " Talker" llamado PUMMA. Este paso puede hacerse manual mente pulsando el botón con este nombre. El generar el evento mencionado, es equivalente a presionar el botón PUMMA.

Posteriormente, se definen los valores por defecto que el programa enviará en caso de que el usuario no seleccione otros para el análisis. Finalmente se definen los "esqueletos" de los programas requeridos para la operación. El término de esqueleto, se usó porque algunos bytes de las listas binarias (blm) pueden ser cambiados dinámicamente sin afectar los demás, esto es particularmente útil, cuando un valor en memoría desea cambiarse sin afectar la operación normal del programa. Para determinar que byte se cambiará, se utilizan una serie de coordenadas que se muestran en las instrucciones `cadponefrec = Chr(15) + Chr(192) + Chr(0) + Chr(19)+ ...`

El listado de este apartado se muestra acontinuación. Acompañado por su diagrama de flujo en la figura 20.

```
Private Sub Form_Load()
VScroll1.Min = 0
VScroll1.Max = 5250
VScroll1.Value = 0
' DEFINE CARPETA DE TRABAJO
ChDrive "d"
ChDir "d:\TESADRF"
'Pinta graticula
pinta_graticula
```

```
'Inicializa puerto serie y baja pumma
inips 1200, 2
retseg 1
Command3_Click

'Define rangos default
nbrs = 0 'rango de salida -1 a 1 v.
nbrf = 1 'rango de frecuencia de 10 a 100 hz
byfmm = 0 'fmax/fmin=2

'Define esqueletos de programas auxiliares
apblmimm = 1: nomarch = trayec & "ponefrec.blm": abreblm
cadponefrec = Chr(15) + Chr(192) + Chr(0) + Chr(192) + Chr(17) + codig
apblmimm = 1: nomarch = trayec & "midemag.blm": abreblm
cadmidemag = Chr(15) + Chr(192) + Chr(0) + Chr(192) + Chr(49) + codig
apblmimm = 1: nomarch = trayec & "midefrec.blm": abreblm
cadmidefrec = Chr(15) + Chr(192) + Chr(0) + Chr(192) + Chr(80) + codig
apblmimm = 1: nomarch = trayec & "midefr.blm": abreblm
cadmidefr = Chr(15) + Chr(192) + Chr(0) + Chr(192) + Chr(114) + codig
apblmimm = 1: nomarch = trayec & "midefase.blm": abreblm
cadmidefase = Chr(15) + Chr(192) + Chr(0) + Chr(192) + Chr(52) + codig
apblmimm = 1: nomarch = trayec & "ang.blm": abreblm
cadang = Chr(15) + Chr(192) + Chr(0) + Chr(192) + Chr(24) + codig
'pone cero en puerto 1bb0
escbyte 7088, 0

End Sub
```

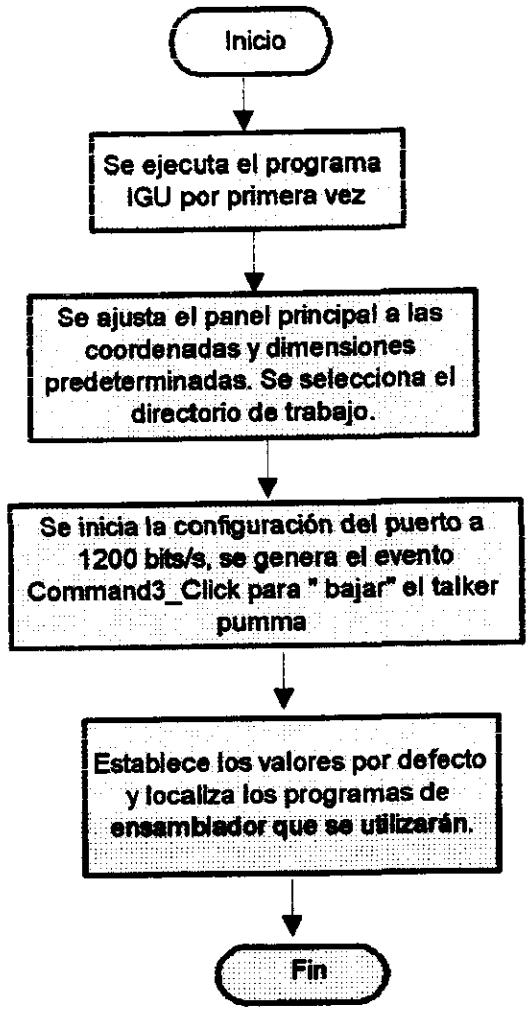


Figura 20 .Diagrama de flujo de la carga de la forma principal de la IGU

3.2.2 Botón PUMMA.

Como se mencionó en el apartado anterior, este botón tiene la función de enviar a la memoria RAM del microcontrolador de la SIMMP2, el programa talker de comunicación. El botón mencionado ejecuta las instrucciones necesarias para que el talker sea cargado adecuadamente a partir de la dirección $0000 del mapa de memoria del microcontrolador, lo que justifica que al final de todos los programas de ensamblador se incluya el salto a la dirección $0000.

El procedimiento ejecutado con este botón (o con el evento `Command3_Click`), inicia las banderas para efectuar la transmisión, localiza la lista binaria correspondiente al talker PUMMA (Pumma99.bin) e invoca a la función bajar, para enviar dicha lista por el puerto serial; una vez enviado cierra la comunicación del puerto.

```
Private Sub Command3_Click()
MousePointer = 11
filenum = FreeFile

apblminm = 1: nomarch = "d:\tesadrf\PUMMA99.BLM": abreblm

Rem ........
XXZZXX:
bapumma = 1
bajar
MousePointer = 11

salida:

retseg 2


MousePointer = 0

mscomm1.PortOpen = False
inips 1200, 2


End Sub

Public Function bajar()
MousePointer = 11
If bapumma = 1 Then
txsi (255)
GoTo bajlistcod
End If


txsi (15): txsi (badi): txsi (bbdi)
txsi (badf): txsi (bbdf)

bajlistcod:

For i = 1 To d2 - d1 + 1
cc = Mid(codig, i, 1)
ccb = Asc(cc): txsi (ccb)

Next i


MousePointer = 0
bapumma = 0

End Function
```

3.2.3 Botones frec_ini y frec_fin

Cuando se pulsa alguno de estos botones se invoca una función (una por cada uno), que pide al usuario ingrese un dato, revisa que se ingrese un número y le asigna una identidad. La identidad para el valor ingresado tras pulsar frec_ini es, `frecini`; mientras que para el caso de frec_fin su identidad es `frecfin`. El código de la función frec_ini se muestre en seguida. No se incluye el de frec_fin pues solo varía en el nombre de la asignación del valor.

```
Private Sub Command5_Click()
Otravez:
frecini = InputBox("Dar frecuencia inicial en Hertz")
'Aquí va filtro de valor inválido para frecini
If IsNumeric(frecini) = False Then
MsgBox "Dato inválido"
GoTo Otravez
End If

fr1 = Val(frecini)
Text1.Text = frecini
Text4.Text = ""
Text5.Text = ""
Text8.Text = ""
End Sub
```

El diagrama de flujo se muestra en la figura 20.

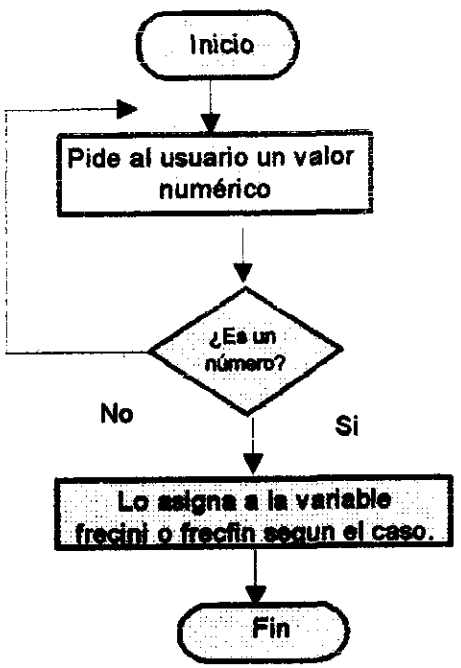


Figura 21 .Diagrama de flujo de los botones frec_ini y frec_fin

3.2.4 Botón Analizar

El botón analizar, se pulsa cuando las opciones de los cuadros de diálogo se han llenado con datos válidos y se desea comenzar con el estudio de la repuesta en frecuencia de un sistema. Cuando este botón se encuentra habilitado, los procedimientos de carga de la forma principal y de envío del programa talker por el puerto serial, han sido ejecutados. Al ser pulsado este botón, se compara primero que los datos correspondientes a las frecuencias inicial y final sean válidos y que cumplan que la frecuencia final sea mayor que la frecuencia inicial.

Posteriormente se divide el intervalo indicado por el usuario, en 175 puntos. Además se inicializan los contadores y banderas que indican la frecuencia máxima y mínima así como los correspondientes a las magnitudes máxima y mínima.

El siguiente paso es llamar a la función correspondiente que mide la magnitud a la salida del sistema cuando la frecuencia se encuentra fija en un valor determinado. Esta funcion se nombró: `mag()`.Dicha función se encarga de llamar al procedimiento de `ponefrec2`, que calcula los valores necesarios en las direcciones de memoria, para generar una frecuencia aproximada al valor indicado por el argumento de `mag()`. Estos valores corresponden a los bytes mudos mencionados en la sección de rutinas de ensamblador. Una vez fijados los bytes mudos se toma una medición del valor de la magnitud de voltaje a la salida del sistema, y se regresa como valor al flujo de ejecución.

Una vez que la función `mag()` terminó su ejecución el valor obtenido se registra en un arreglo de valores `magni(i)`, el cuál por medio de una operación aritmética se convierte en otro arreglo con los valores reales de la medición `xvda(i)=5250*(fvda - fr1)/ delf`.

A continuación se procede a determinar el ángulo en ese instante, se llama a la función que primero determina si la relación entre la señal de entrada con la señal de salida es de atraso o de adelanto `fase frp`. Posteriormente calcula de acuerdo a la relación, el ángulo en sí; regresando al flujo de ejecución el valor correspondiente que será registrado en el arreglo `angi(i)`.

Finalmente el procedimiento determina cuales son los valores máximos y mínimos de amplitud y fase para que con éstos y con los arreglos `xvda(i)` y `angi(i)`; presente en la pantalla del panel virtual la gráfica obtenida en el análisis correspondiente.

El listado correspondiente se muestra a continuación:

```
Dim xr, yr As Single

MousePointer = 11
'Analizar respuesta de magnitud para el intervalo f1-f2
If fr1 = 0 Or fr2 = 0 Then
MsgBox "Datos inválidos"
GoTo salanalmag
End If

If fr1 > fr2 Then
MsgBox "frec_ini debe ser menor a frec_fin"
GoTo salanalmag
End If

delf = fr2 - fr1
mipend = delf / 175
maxmag = 0: minmag = 1E+30
'********
maxang = 0: minang = 1E+30

'********
delmax = 0
For i = 0 To 175 'Step 5
frp = mipend * i + fr1
magni(i) = mag(frp)
xvda(i) = 5250 * (fvda - fr1) / delf
```

```
fase frp
'Medición de fase
angi(i) = angulo
If i > 0 Then
'delmag = magni(i) - magni(i - 5)
End If
'retseg 1
'MsgBox Str(i) & "frec=" & Str(frp) & "mag=" & Str(magni(i)) & "delta=" &
Str(delmag)
If Abs(delmag) > delmax Then
delmax = Abs(delmag)
End If
'aquí generar cotas inferior y superior de
'angulo(i)
If angi(i) > maxang Then
maxang = angi(i)
End If
If angi(i) < minang Then
minang = angi(i)
End If


'*********************************
If magni(i) > maxmag Then
maxmag = magni(i)
End If
If magni(i) < minmag Then
minmag = magni(i)
End If
'*********************


Next i

Text4.Text = maxmag
Text5.Text = minmag
Text8.Text = maxmag / 8

'MsgBox Str(delmax) & "maxmag=" & Str(maxmag)
'Graficación de curva obtenida

MsgBox Str(maxmag) & "     " & maxang

'Picture1.DrawWidth = 2
For j = 0 To 175
'xr = 30 * j
xr = xvda(j)
yr = 4200 - magni(j) * 4200 / maxmag

'yang = 4200 - angi(j) * 4200 / maxang
yang = angi(j) * 4200 / maxang
Picture1.PSet (xr, yr), RGB(255, 255, 255)
Picture1.PSet (xr, yang), RGB(255, 255, 0)

Next j
Picture1.DrawWidth = 1
MousePointer = 0
salanalmag:
End Sub
```

```
Public Function angulo() As Single
zoto = midfase
mscomm1.Output = cadang
recs (1)
ss = Asc(datrec)
miang = 4.94 * ss / 255
miang = 180 * miang / 4.6
If zoto = "atraso" Then
angulo = miang
Else
angulo = 360 - miang
End If

End Function

Public Function midfase() As String
mscomm1.Output = cadmidefase
recs (1)
If datrec = Chr(2) Then
midfase = "adelanto"
Else
midfase = "atraso"
End If

End Function
```

El diagrama de flujo correspondiente a la figura 22, ilustra de manera global los procedimientos llevados a cabo para realizar un análisis en un intervalo de frecuencias, los diagramas de flujo correspondientes a las figuras 23, 24 y 25 son los procesos que en la figura 22, se marcaron como bloques y que requieren de una descripción posterior.

En lo que corresponde a la descripción de la interface IGU, solo falta por describir la función del botón borrar, la cual es simplemente restablecer las opciónes de las gráficas a su estado inicial; dejando libre el área de trazado para un nuevo par de gráficas.

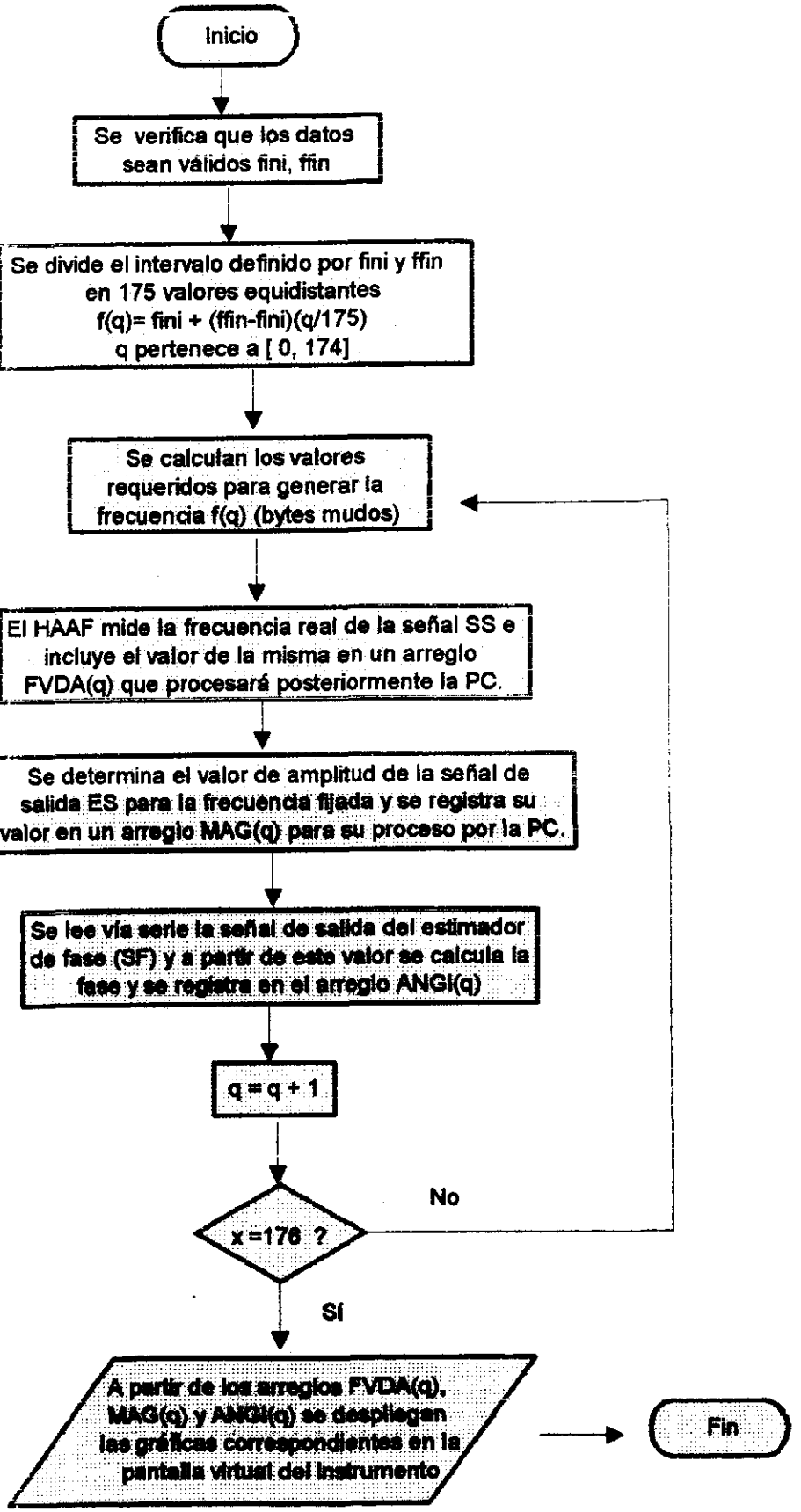


Figura 22 .Diagrama de flujo del botón analizar (global).

El siguiente diagrama ilustra el procedimiento del cálculo de los bytes mudos. Los valores de la ecuación de rango se obtienen de despejar la ecuación 2.3 y sustituir los valores de capacitancia de las tablas 1 y 2.

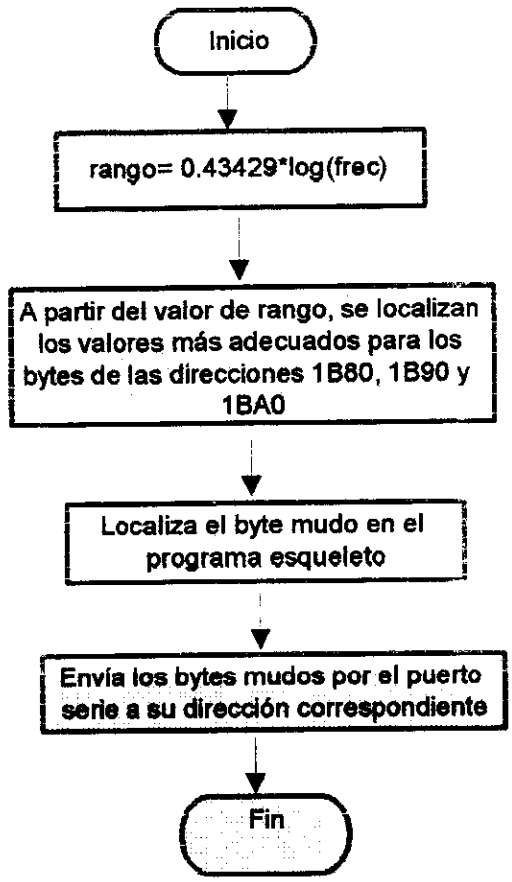


Figura 23 .Cálculo de los bytes mudos.

El siguiente diagrama corresponde a la secuencia de acciones realizadas por el dispositivo para leer un valor de magnitud.

La varable indicada como pend corresponde a la división del intervalo de frecuencias entre los 175 puntos.

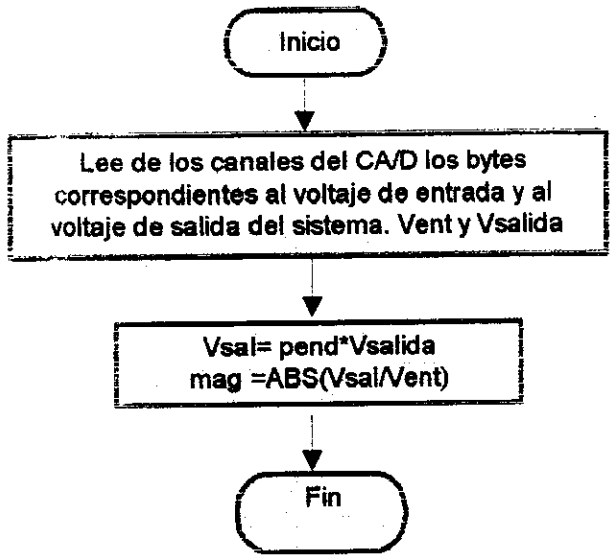


Figura 24. Obtención del valor de magnitud.

Finalmente se presenta el proceso de cálculo de la fase a manera de diagrama de flujo: El valor de 4.5 corresponde a la constante dada por el convertidor a niveles TTL.

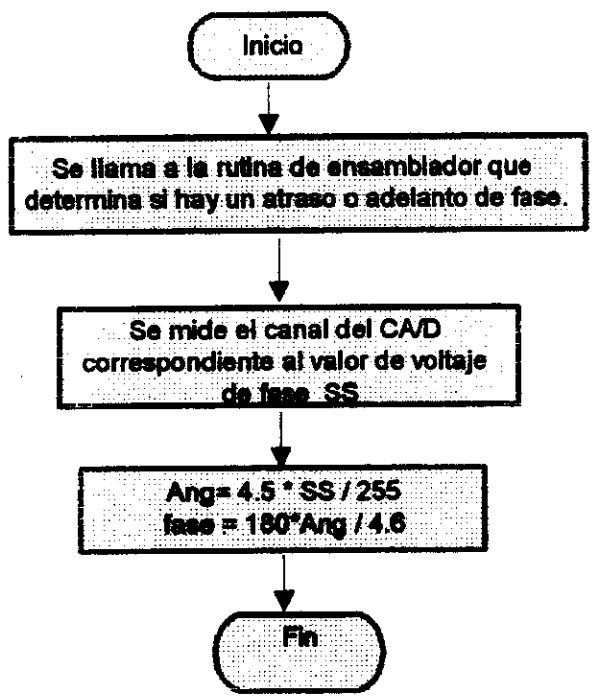


Figura 25. Proceso de obtención de fase

Capítulo 4.

Ejemplo de aplicación

Descripción: *El presente capítulo muestra un ejemplo de aplicación del sistema ADRF. Se presenta en primer lugar el diseño de un filtro, se incluyen las respuestas teóricas; las cuales se comparan con las curvas de respuesta en frecuencia obtenidas con el prototipo antes descrito.*

4.1 Características de un filtro pasobajas Butterworth de segundo orden

En muchas aplicaciones de filtros pasobajas, es necesario que la ganancia del mismo esté tan próxima como sea posible a 1 dentro de la banda de paso. El filtro Butterworth es el más adecuado para este tipo de aplicaciones.

Se escogió un filtro Butterworth de segundo orden, para mostrar la operación del sistema ADRF, el filtro proporciona una atenuación de magnitud de –40dB / década, es decir que después de la frecuencia de corte la magnitud del voltaje de salida decrece 40dB en cuanto la frecuencia aumenta a 10 veces la frecuencia de corte. La siguiente figura muestra gráficamente el comportamiento de la respuesta de magnitud del filtro

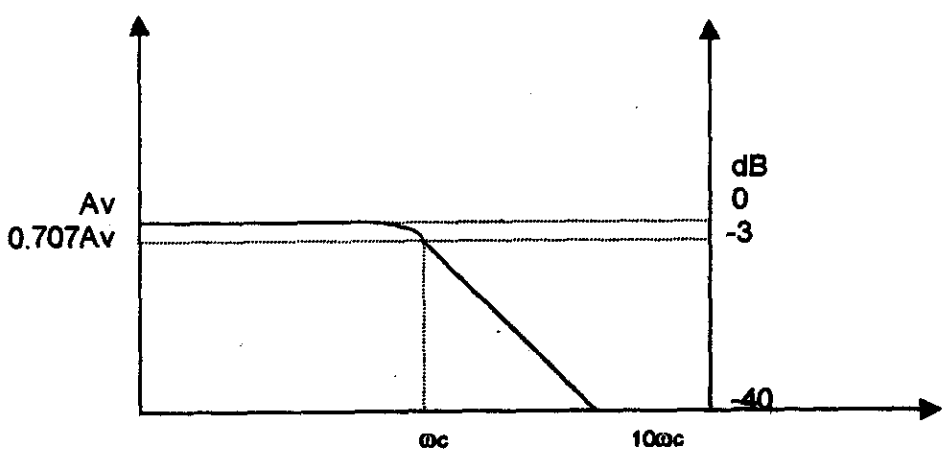


Figura 26. Respuesta de amplitud teórica del filtro Butterworth

La respuesta de fase de un filtro Butterworth paso bajas de segundo orden, se caracteriza porque en la frecuencia f =0 el defasamiento es 0 °. Para la frecuencia de corte f = fc, el ángulo de fase es de –90° , mientras que la frecuencia tiende a incrementarse el ángulo de fase converge a -180°. La curva de ángulo de fase es antisimétrica respecto al punto de inflexión, es decir el punto donde el defasamiento es –90°

La figura 27 muestra el comportamiento de la respuesta de fase del filtro Butterworth

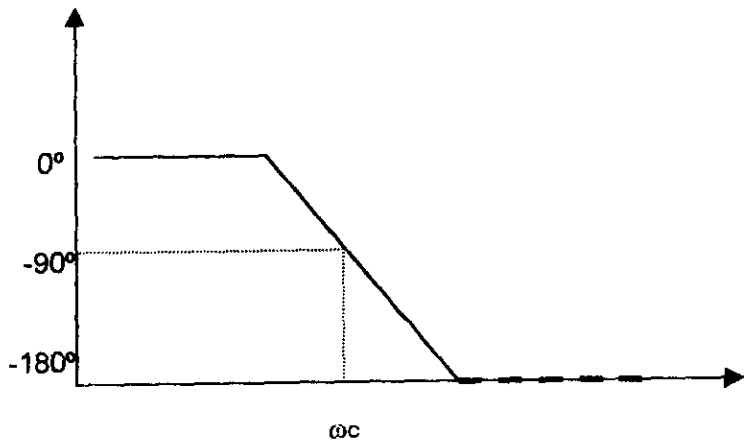


Figura 27. Respuesta de amplitud teórica del filtro Butterworth

Se escogió la configuración de retroalimentación múltiple para implementar dicho filtro. La figura 27 muestra los componentes requeridos para tal fin:

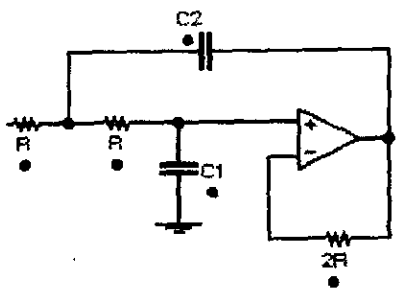


Figura 28. Configuración del filtro pasobajas.

4.2 Diseño de un filtro pasobajas Butterworth de segundo orden con frecuencia de corte en 1125 Hz

Para el diseño del filtro se siguieron los siguientes pasos:

1.- Se eligió la frecuencia de corte en fc = 1125 Hz . Este valor proporciona que los valores de los elementos discretos se aproximen mejor a un valor comercial.

2.- Se escogió un valor de 0.1 μF para C1, la relación con C2 es: C2 =2C1, por lo tanto C2 = 0.2 μF

3.- Se calcula $R = \frac{\sqrt{2}}{2 \cdot \pi \cdot f_c C_1}$

4.- Los valores obtenidos son:

R= 1kΩ , C1 = 0.1 μF, C2 = 0.2 μF, fc = 1125 Hz.

Recordando que la función de transferencia de un sistema de segundo orden es del tipo:

$$H(s) = \frac{\omega_n^{\,2}}{s^2 + 2\xi \cdot \omega_n s + \omega_n^{\,2}}$$

y dado que :

$\omega_n = 2\pi \cdot f_c$

$\xi = \frac{\sqrt{2}}{2}$

La función de transferencia del filtro puede expresarse como:

$$H(s) = \frac{[1125(2\pi)]^2}{s^2 + \sqrt{2}[1125(2\pi)]s + [1125(2\pi)]^2}$$

La figura 29 muestra la conexión realizada para obtener la medición con el ADRF, se omitió la computadora personal en la misma.

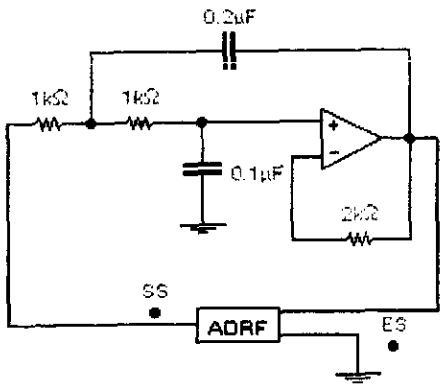


Figura 29. Conexión entre el ADRF y el filtro

A partir de la función de transferencia H(s), es posible determinar matemáticamente la respuesta en frecuencia utilizando la siguientes expresiones:

$$H(s) = H(j\omega) = \mathrm{Re}\{j\omega\} + j\,\mathrm{Im}\{j\omega\}$$
$$H(j\omega) = |H(\omega)|e^{j\Phi(j\omega)} = |H(\omega)|\angle\Phi(\omega)$$
$$\Phi(\omega) = \tan^{-1}\frac{\mathrm{Im}\{j\omega\}}{\mathrm{Re}\{j\omega\}}$$

Las mismas establecen que la función de transferencia en el dominio de s puede expresarse en términos de jω y a su vez puede descomponerse en dos términos, uno real y otro complejo; pudiendo ahora representar a la función en coordenadas polares con una magnitud y un ángulo, que respectivamente corresponden a las respuestas de magnitud y de fase respectivamente.

Recordando que $\omega = 2\pi f$ y $\xi = \frac{\sqrt{2}}{2}$ se determinó:

$$H(j\omega) = \frac{[1125\ (2\pi)]^2}{(j\omega)^2 + \sqrt{2}[1125\ (2\pi)]j\omega + [1125\ (2\pi)]^2}$$
$$|H(\omega)| = \frac{[1125\ (2\pi)]^2}{\sqrt{[1125\ (2\pi)]^4 + \omega^2}}$$
$$\Phi(\omega) = \tan^{-1}\left(\frac{2\xi\frac{\omega}{1125\ (2\pi)}}{1 - \frac{\omega}{1125\ (2\pi)}}\right)$$

La gráfica correspondiente a la magnitud se muestra en la figura 30. La misma se obtuvo con ayuda del programa MATLAB.

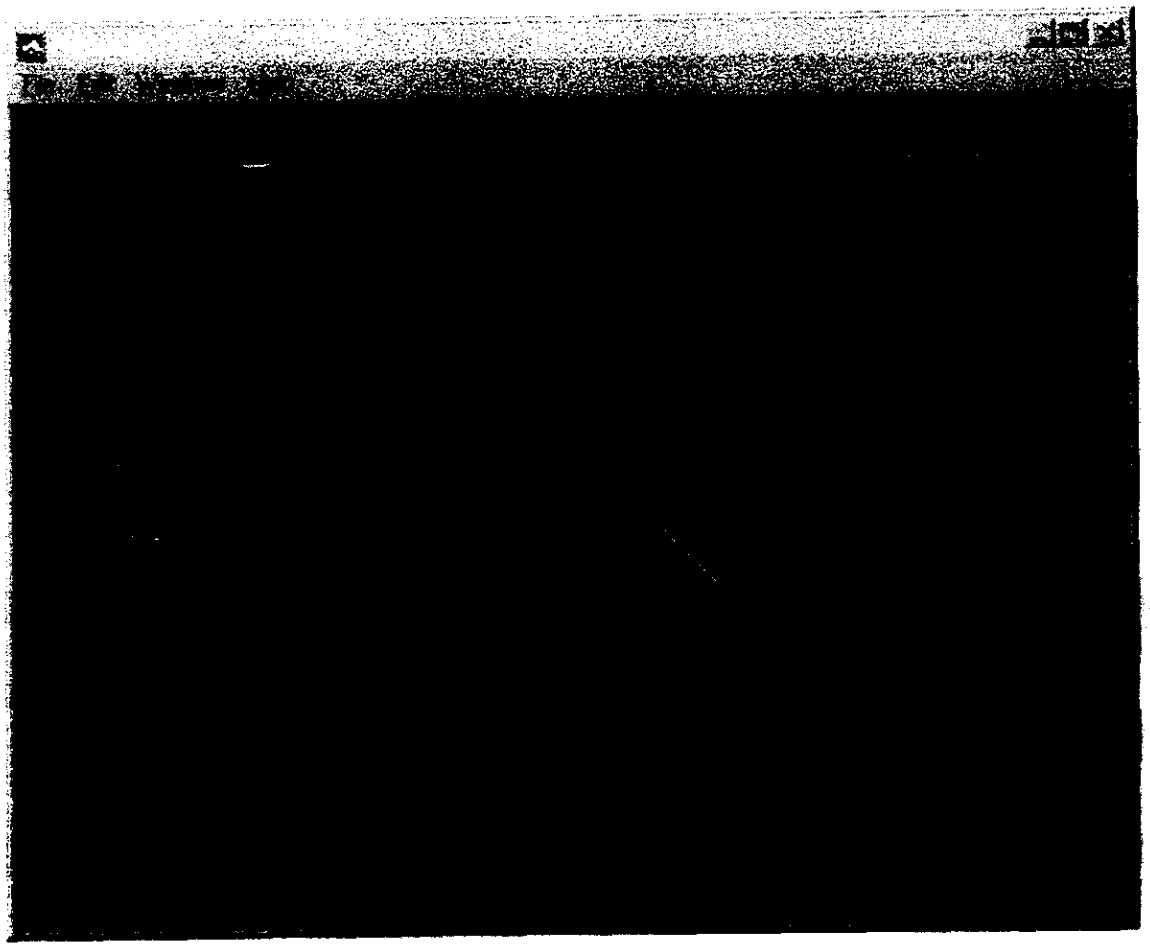


Figura 30. Gráfica de respuesta de amplitud obtenida con matlab.

El eje de las abscisas corresponde a la frecuencia en hertz mientras que el eje de las ordenadas representa la relación de amplitud o ganancia.

La figura 31 representa la gráfica de respuesta en fase obtenida con MATLAB.

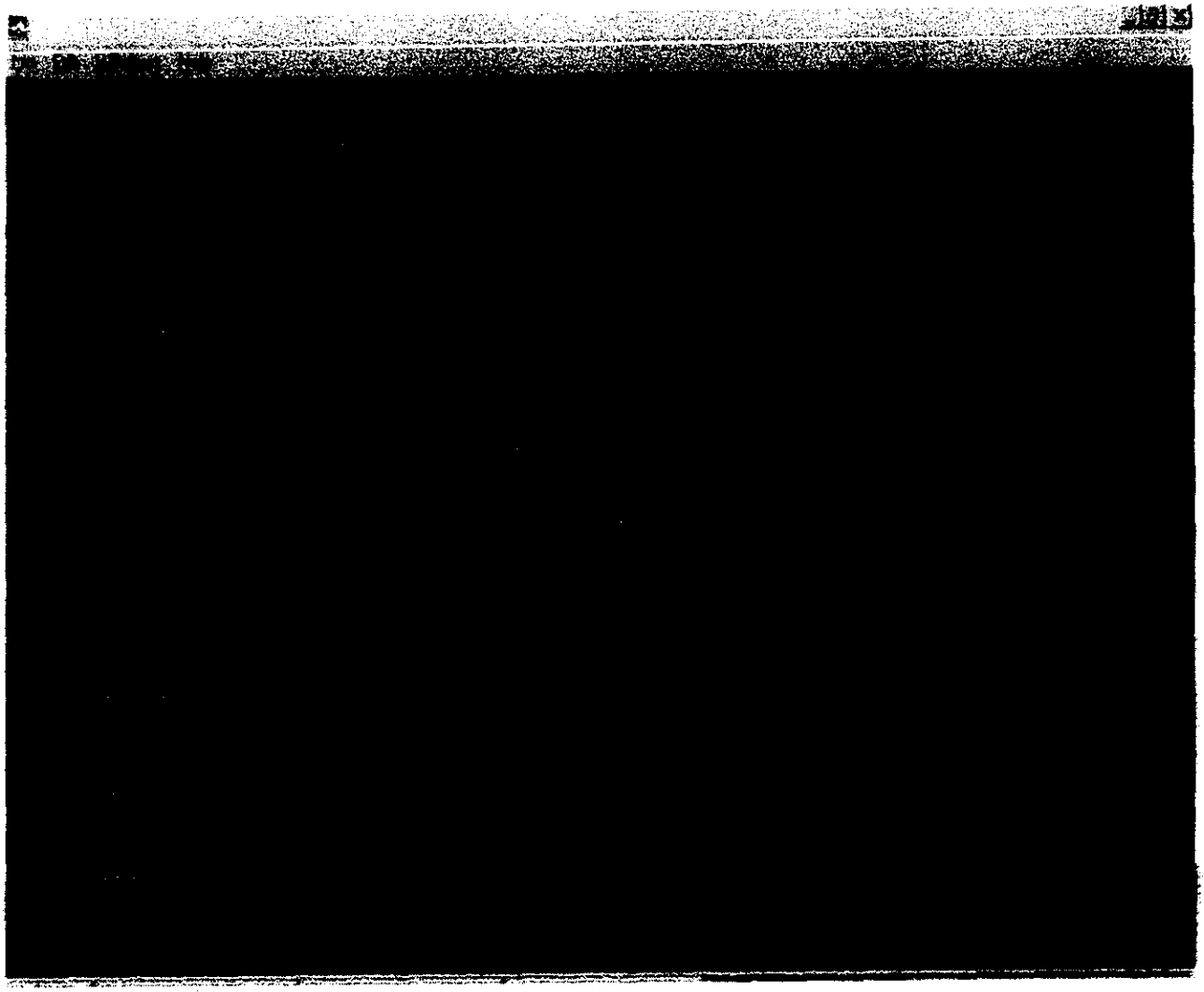


Figura 31. Gráfica de respuesta de fase obtenida con Matlab

La figura 32 muestra las gráficas obtenidas con el sistema ADRF, la gráfica superior representa la respuesta de amplitud, mientras que la gráfica bajo ésta corresponde a la respuesta de fase.

De acuerdo con el análisis experimental, empleando el ADRF, la frecuencia de corte es aproximadamente 1035 Hz. el valor teórico obtenido a partir de la función de transferencia del filtro, es 1125 Hz, la diferencia puede atribuirse a las tolerancias de los elementos comerciales empleados en la construcción del circuito.

La respuesta de fase concuerda con el perfil teórico, pues la transición de fase empieza en un valor cercano a cero, descendiendo a 90 grados en la frecuencia de corte, para que en la frecuencia de supresión se observe un comportamiento asintótico hacia 180°.

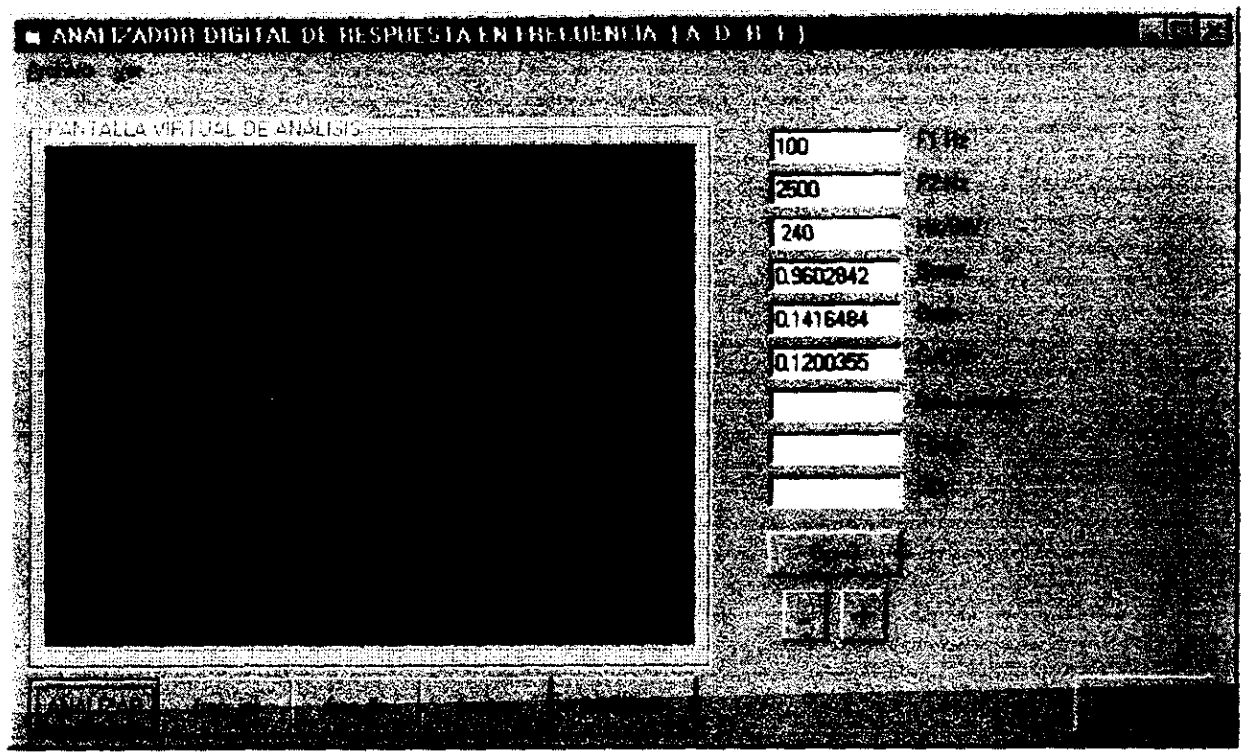


Figura 32. Gráficas obtenidas con el ADRF

En cuanto a la curva de magnitud se observa también que al rededor de 1000 hz existe una pequeña dispersión de los valores obtenidos, ésta se atribuye a que al obtener una relacion de 10 a 1 entre la frecuencia inicial y la frecuencia final, el circuito XR2206 presenta un comportamiento alineal. Ya sea por la introducción de elementos extra a las impedancias del capacitor y el resistor que deben conformar el oscilador del mismo, o por caracteristicas propias del circuito. Los elementos extra a los que se hce referencia, son los relevadores, los multiplexores analógicos y los cables de conexión.

Por otra parte en la curva de fase se observa que al inicio del análisis la fase no es cero, sino un valor aproximado a cero, lo cual puede deberse a los elementos utilizados para determinar la diferencia entre fases, pues al analizar la señal entregada por la compuerta XOR, los ciclos aunque cercanos, no alcanzan los extremos correspondientes al 0% ni al 100% del ciclo de trabajo.

Es indudable que las características reales de los componentes utilizados se ven reflejadas en los resultados obtenidos, pues el diseño se hizo a partir de las características ideales de dichos componentes, de alguna manera se compensaron algunas, utilizando el circuito registrador de frecuencia que determina la diferencia entre el valor teórico y el real de una frecuencia obtenida a partir de la elección de ciertos valores de las variables que intervienen al generarse una señal a una frecuencia determinada.

Conclusiones

El trabajo presentado corresponde a un sistema anfitrión - destino, diseñado de tal manera que un programa de computadora, recibe instrucciones del usuario, las interpreta y las envía a un sistema electrónico basado en un microprocesador 68HC11; el cuál ejecuta las órdenes y regresa a un estado de espera de la siguiente instrucción.

La finalidad del trabajo fue obtener las curvas de respuesta en frecuencia de un sistema eléctronico, aunque con la idea general, un sistema de este tipo tiene múltiples aplicaciones en procesos de automatización, instrumentación y control.

El trabajo presentado combina la programación en un lenguaje gráfico con el diseño electrónico, obteniéndose una herramienta capaz de ejecutar acciones de manera automática, incluyendo el registro de datos, medición de variables físicas y procesamiento de información; a partir de datos proporcionados por el usuario. El diseño del mismo se basa en la comunicación entre un dispositivo electrónico (microcontrolador) y un programa de computadora, que sirve de enlace con las instrucciones del usuario.

La parte más importante para establecer el enlace entre el microcontrolador y la computadora personal, es sin duda el programa de comunicación que reside en el microcontrolador y que a lo largo del presente documento se denominó "talker", este programa debe ser lo suficientemente compacto para ajustarse a las dimensiones de una memoria EPROM, pero lo sufucientemente completo para administrar las funciones de comunicación entre los dos dispositivos. La filosofía de el regreso al estado de espera, de la siguiente instrucción debe estar implícita en el desarrollo de dicho "talker".

El diseño del ADRF, siempre tuvo en mente la filosofía antes descrita, es decir que tras finalizar la ejecución de cada comando, el microcontrolador regresa a un estado de espera de la siguiente instrucción, lo que explica que después de cada rutina programada en lenguaje ensamblador, el flujo de ejecución se apunte a una dirección determinada en la RAM del dispositivo.

La mayoría de las funciones que ejecuta el ADRF, se implementan mediante la ejecución de un pequeño programa en lenguaje ensamblador. El talker recibe dichos programas mediante el puerto serial y al terminar de cumplir con las órdenes contenidas en estos, regresa a eperar el siguiente programa. Esta característica es muy importante, pues permite elaborar programas independientes para cada función, haciendo el proceso de prueba más sencillo. Además brinda la posibilidad de crecimiento de las funciones del dispositivo y ahorra espacio en memoria permanente, pues no es necesario que el programa que ejecuta todas la acciones del dispositivo, resida en una memoria EPROM. Brinda además un procedimiento estructurado y ordenado de ejecución, evitando los saltos a direcciones alejadas que suelen presentarse en programas de mayor extensión.

Del lado de la PC el programa desarrollado ofrece las ventajas de uso de un programa de ambiente Windows, su manejo es sencillo y ofrece gráficas de buena calidad visual, la parte más significativa del mismo lo constituye la administración del puerto serie para establecer comunicación con el microcontrolador, sin descartar las funciones y procedimientos necesarios para interpretar las órdenes ingresadas por el usuario en instrucciones para el microcontrolador.

Las funciones de procesamiento y almacenamiento de la información poseen múltiples alternativas, una vez adquiridos los datos; mediante el software se pueden aplicar diversas funciones de transformación, análisis, filtrado, etc. Además de contarse con la posibilidad de almacenarse en un archivo, o en una impresión. El poder contar con información digitalizada de un proceso físico permite un análisis completo y flexible cuando se hace con la ayuda de la computadora.

En cuanto al dispositivo electrónico, puede establecerse la importacia de combinar elementos digitales con analógicos, los problemas más comunes al combinarlos, se presentan en las etapas de acoplamiento de impedancias y conversion a niveles adecuados, de voltaje y corriente.

Del dispositivo presentado, puede comentarse que es el resultado de razonamientos sencillos en cuanto a la medición de respuesta en frecuencia, se utilizan dos subsistemas para la obtención de las curvas, el primero determina la relación de magnitudes entre las señales de entrada y salida del sistema por analizar, para finalmente ejecutar una división y obtener un valor puntual a medida que se efectua el análisis.

El segundo subsistema convierte las señales de entrada y salida del sistema por analizar, en señales digitales, que se operan mediante una compuerta TTL y después se filtran para obtener un valor de voltaje proporcional a la fase entre ambas señales.

Los resultados obtenidos son satisfactorios, pues las curvas obtenidas prácticamente confirman el diseño teórico del dispositivo, reflejando características reales de los dispositivos sujetos a analizar así como de los elementos constitutivos del sistema. El diseño y construcción de un dispositivo de estas características, permite observar el comportamiento real y práctico de los elementos electrónicos, bajo distintas condiciones, las conclusiones obtenidas a partir de la observación práctica del comportamiento de dichos elementos es sumamente valiosa y permite ampliar el conocimiento en cuanto al diseño con elementos reales.

Las mejoras al resultado final del trabajo pueden incluirlo siguiente: La calibración de diversos elementos con dispositivos de medición confiables, la modificación del programa de computadora para presentar al usuario diversas posibilidades de cálculo y despliegue, finalmente las modificaciones al sistema deben explorar la posibilidad de utilizar elementos eléctronicos diferentes, así como métodos alternativos para generar las señales de referencia .

Finalmente debe comentarse que el diseño del sistema se realizó con materiales utilizados en los laboratorios de las asignaturas de el área de electrónica, el costo de los mismos es relativamente reducido y facilmente se encuentra en las tiendas especializadas.

5. REFERENCIAS

[1] Motorola Inc, HC11 M68HC11 Reference manual (1991)

[2] Exar Corporation, XR2206 Monolithic Function Generator (1997)

[3] Cornell, Gary, "Visual Basic 4.0 para programadores"
McGraw - Hill, México (1993)

[4] Pérez S.L, Bañuelos M.A
Sisteme de medición de temperatura usando instrumentación virtual
Memorias SOMI XIII, Congreso de instrumentación .
Sociedad Mexicana de Instrumentación 1998

[5] Fuentes R.
Caracterización de los sensores inteligentes en el proceso de Automatización
Tesis de maestría
ITESM Morelos 1995

[6] National Semiconductor Corporation
National Data Acquisition Databook, 1995

[7] Coughlin F. Driscoll F.
Amplificadores Operacionales Y Circuitos Integrados Lineales
Prentice Hall 1993

[8] Savant C. Roden M. Carpenter G.
Diseño electrónico, circuitos y sistemas
Addison Wesley 1992.

[9] Ogata K
Ingeniería de Control Moderna
Prentice Hall 1993

Apéndice 1. Guia de usuario de la tarjeta SIMMP2

Tomado de :

"Guía de usuario del sistema PUMMA – SIMMP2"
Antonio Salvá Calleja
Documento de circulación gratuita
Facultad de Ingeniería UNAM.

CAPÍTULO 2

GUÍA DE USO DE LA COMPUTADORA MONOTABLILLA SIMMP-2.

La computadora monotablilla (CMT) SIMMP-2 es una arquitectura basada en el microcontrolador 68HC11F1 fabricado por MOTOROLA, la tarjeta incorpora un programador de memorias EPROM y puede ser operada en cualquiera de los cuatro modos de operación del 68HC11, aunque lo recomendable, dadas las características propias del 68HC11F1, es operar la tarjeta en los modos expandido o boot-strap.

Para operar el programador de memorias EPROM la tarjeta necesariamente deberá operar en modo boot-strap, mas adelante se explicará el funcionamiento del mismo.

Al operar la tarjeta SIMMP-2 en los modos expandido o TEST se pueden configurar diferentes mapas de memoria con sus respectivos submapas de puertos, dichas configuraciones se logran cambiando puentes (jumpers), así como también firmware residente en la tarjeta; mas adelante se hablará de como hacer estas configuraciones.

En la Figura 2.1 se muestra el diagrama a bloques de la CMT SIMMP-2, apreciándose que el usuario tiene acceso a casi todas las líneas asociadas ya sea con periféricos propios del microcontrolador o con líneas de control del mismo, el puerto G del microcontrolador no está disponible para el usuario ya que es empleado por la CMT SIMMP-2 para arbitrar la paginación de memoria y puertos y el programador de memorias EPROM; a continuación se describe genéricamente el funcionamiento de los bloques funcionales de la CMT SIMMP-2 que aparecen en la Figura 2.1.

MICROCONTROLADOR 68HC11F1.- Este es el circuito integrado número uno de la CMT SIMMP-2 y es una versión del 68HC11 que cuenta, con las siguientes características adicionales a las que se cuentan en otras versiones del 68HC11 y estas son entre otras las siguientes:

a) Bus de direcciones demultiplexado.
b) Puerto paralelo F de salida, disponible para el usuario sólo en los modos boot-strap o single-chip. Los pines que dan acceso a este puerto validan la parte baja del bus de direcciones al operar el microcontrolador en los modos expandido o TEST.
c) Puerto G bidireccional. Los cuatro bits más significativos de este puerto, pueden ser empleados para controlar la paginación de memoria y puertos cuando el microcontrolador opera en modo expandido o TEST, en cuyo caso quedarían disponibles para el usuario sólo los cuatro bits menos significativos de este puerto. En la tarjeta SIMMP-2 este puerto no está disponible para el usuario debido a que el mismo es empleado para arbitrar parte del funcionamiento de la arquitectura.

CONVERTIDOR TTL-RS 232.- Este bloque funcional está realizado con el chip MAX-232 muy popular en la industria para estos fines; la función del mismo consiste en cambiar los niveles lógicos de voltaje TTL a los niveles RS-232 (+12 volts y -12 volts) y viceversa.

LÓGICA M-I.- Este bloque genera señalización de control de periféricos INTEL para que los mismos puedan ser empleados por la CMT SIMMP-2, es importante aclarar aquí que el periférico INTEL que se deseara conectar no operará correctamente si la velocidad de operación del mismo no es compatible con la correspondiente a la tarjeta SIMMP-2. Las señales de control generadas son $\overline{WR}$ y $\overline{RD}$.

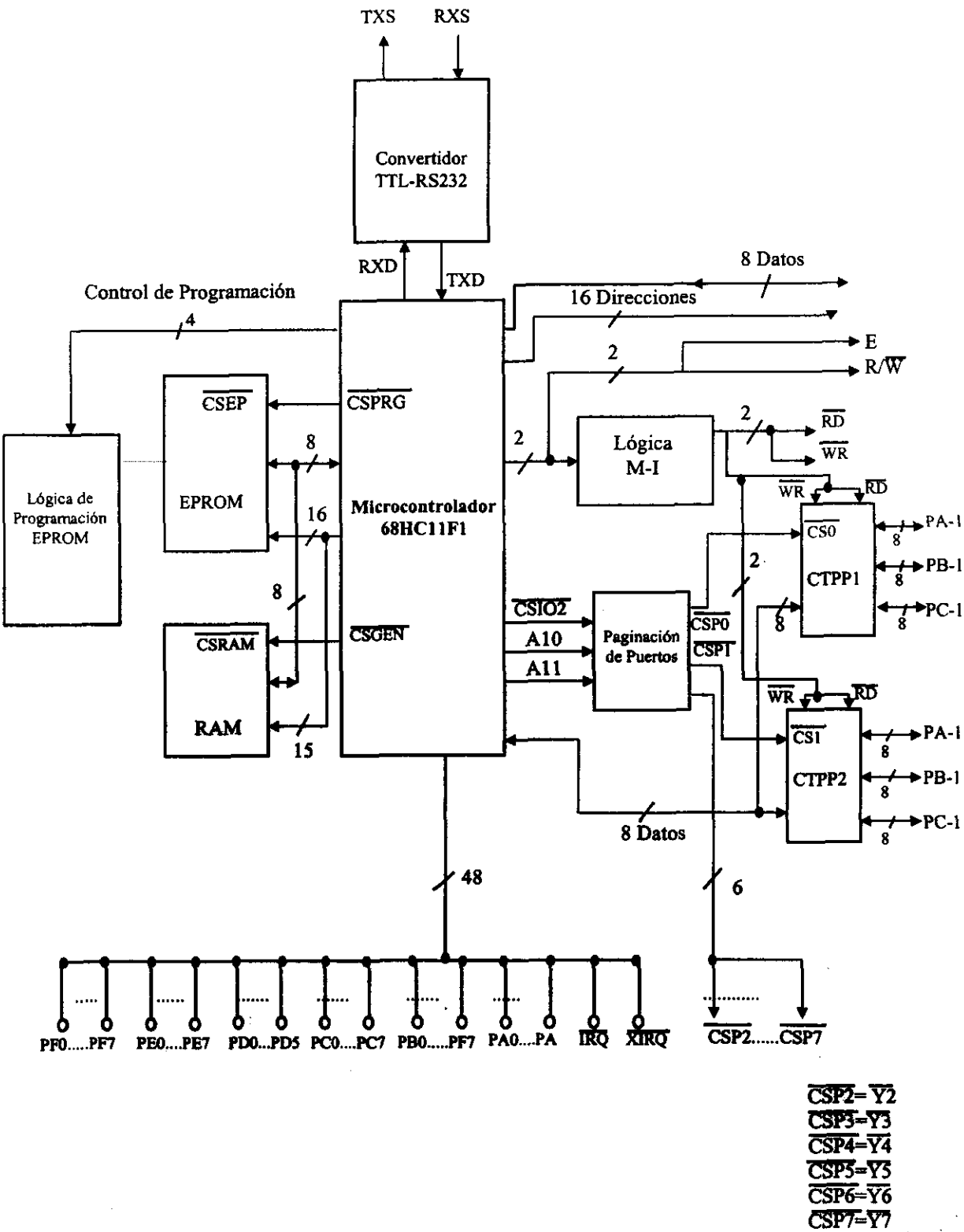


Figura 2.1.- Diagrama de Bloques de la CMT SIMMP-2.

PAGINADOR DE PUERTOS.-Este bloque está realizado con un decodificador de 3 a 8 74LS138 y el mismo genera 8 líneas de paginación de puertos denominadas como CSP0 a CSP7, validando cada una de estas un intervalo de 128 direcciones, en los conectores de la Tarjeta SIMMP-2 el usuario tiene acceso a seis de las ocho líneas mencionadas anteriormente (CSP2 a CSP7), siendo las mismas respectivamente denominadas como Y2 a Y7; a continuación se indica el intervalo de direcciones que verifica cada línea de paginación de puertos.

LÍNEA DE PAGINACIÓN DE PUERTO	LÍNEA EN CONECTORES DE SIMMP-2	INTERVALO DE DIRECCIONES
$\overline{CSP0}$	*	$1800 a $17FF
$\overline{CSP1}$	*	$1880 a $18FF
$\overline{CSP2}$	$\overline{Y2}$	$1900 a $197F
$\overline{CSP3}$	$\overline{Y3}$	$1980 a $19FF
$\overline{CSP4}$	$\overline{Y4}$	$1A00 a $1A7F
$\overline{CSP5}$	$\overline{Y5}$	$1A80 a $1AFF
$\overline{CSP6}$	$\overline{Y6}$	$1B00 a $1B7F
$\overline{CSP7}$	$\overline{Y7}$	$1B80 a $1BFF

* Estas líneas de paginación de puerto no están disponibles para el usuario ya que el sistema las usa para habilitar los bloques funcionales CTPP1 y CTPP2 de los cuales se hablará mas adelante.

CONJUNTO TRIPLE DE PUERTOS PARALELOS NÚMERO UNO (CTPP1).- Este bloque funcional está realizado por un chip INTEL 82C55 que es un conjunto de tres puertos paralelos programables denotados en la hoja de datos correspondiente como PA, PB, y PC, cada uno de estos puertos tienen un registro de datos asociado; en la notación de la tarjeta SIMMP-2 estos puertos son nombrados como PA-1, PB-1 y PC-1 para evitar confusiones con los puertos que son denotados con las letras A, B, y C en el microcontrolador 68HC11F1. El 82C55 tiene cuatro direcciones asociadas, tres corresponden con los tres puertos mencionados y la cuarta está asociada con un registro de control del chip mediante el cual se puede programar tanto el modo en que va a operar el chip como la naturaleza de entrada o salida de cada uno de los puertos del mismo; las direcciones asociadas en el mapa de puertos de la tarjeta SIMMP-2 son las siguientes:

PUERTO O REGISTRO EN CTPP1	DIRECCIÓN EN SIMMP-2
PA-1	$1800
PB-1	$1801
PC-1	$1802
REGISTRO DE CONTROL	$1803

Dado que el 82C55 usa sólo dos líneas de direccionamiento y cada línea de paginación de puertos en la CMT SIMMP-2 se verifica al invocarse un intervalo de 128 direcciones, las direcciones mencionadas anteriormente estarán repetidas 32 veces esto es: el puerto PA-1 estará en la direcciones 1800, 1804, 1808, hasta la 187C; el puerto PB-1 estará en las direcciones 1801,

1805, 1809, hasta la 187D; el puerto PC-1 estará en las direcciones 1802, 1806, 180A, hasta la 187E; el registro de control de este bloque CTPP1 estará localizado en las direcciones 1803, 1807, 180B, hasta la 187F.

CONJUNTO TRIPLE DE PUERTOS PARALELOS NÚMERO DOS (CTPP2).- Este bloque funcional está realizado por un chip INTEL 82C55 que es un conjunto de tres puertos paralelos programables denotados en la hoja de datos correspondiente como PA, PB, y PC, cada uno de estos puertos tienen un registro de datos asociado; en la notación asociada con la tarjeta SIMMP-2 estos puertos son nombrados como PA-2, PB-2 y PC-2 para evitar confusiones con los puertos que son denotados con las letras A, B, y C en el microcontrolador 68HC11F1. El 82C55 tiene cuatro direcciones asociadas, tres corresponden con los tres puertos mencionados y la cuarta está asociada con un registro de control del chip mediante el cual se puede programar tanto el modo en que va a operar el chip como la naturaleza de entrada o salida de cada uno de los puertos del mismo; las direcciones asociadas en el mapa de puertos de la tarjeta SIMMP-2 son las siguientes:

PUERTO O REGISTRO EN CTPP2	DIRECCIÓN EN SIMMP-2
PA-2	$1880
PB-2	$1881
PC-2	$1882
REGISTRO DE CONTROL	$1883

Dado que el 82C55 usa sólo dos líneas de direccionamiento y cada línea de paginación de puertos en la CMT SIMMP-2 se verifica al invocarse un intervalo de 128 direcciones, las direcciones mencionadas anteriormente estarán repetidas 32 veces esto es: el puerto PA-2 estará en la direcciones 1880, 1884, 1888, hasta la 18FC; el puerto PB-2 estará en las direcciones 1881, 1885, 1889, hasta la 18FD; el puerto PC-2 estará en las direcciones 1882, 1886, 188A, hasta la 18FE; el registro de control de este bloque CTPP2 estará localizado en las direcciones 1883, 1887, 188B, hasta la 18FF.

Para más detalles acerca del funcionamiento del 82C55 se puede consultar las especificaciones técnicas proporcionadas por INTEL.

Las versiones del 82C55 que al emplearse en la CMT SIMMP-2 han sido compatibles con la velocidad de operación de la tarjeta son la 82C55-A2 y la KS82C55A-8CP.

Los bloques CTPP1 y CTPP2 tienen sentido como parte de la arquitectura de la CMT SIMMP-2 cuando ésta opera en modo expandido o TEST, en otro caso no formarían parte del mapa de memoria.

En caso de que la tarjeta CMT SIMMP-2 no cuente con ninguno de los chips 82C55 que realizan los bloques CTPP1 y CTPP2 el usuario podrá emplear registros 74LS373 o 74LS374 o periféricos MOTOROLA para realizar puertos de entrada o salida, mas adelante se explicará como hacer esto para el caso de los chips TTL mencionados.

MEMORIA RAM.- Este bloque es la memoria RAM estática que se emplee en la CMT SIMMP-2 en un momento dado, para ello existe en la tarjeta una base de 28 pines que puede aceptar memorias RAM de 8 o 32 kb, dependiendo de el mapa de memoria en modo expandido que haya escogido el usuario.

MEMORIA EPROM.- Este bloque podrá ser una memoria EPROM de 2k a 64k para fines del programador de memorias EPROM contenido en la tarjeta, si la EPROM en cuestión está colocada para ser parte de el mapa de memoria de la CMT SIMMP-2 operando en modo expandido o TEST necesariamente su tamaño deberá ser de 8k o 32k, dependiendo del mapa de memoria escogido por el usuario.

LÓGICA DE PROGRAMACIÓN DE MEMORIAS EPROM.- Este bloque está integrado por circuitería analógica y digital que realiza físicamente la programación de memorias EPROM, efectuándose esto mediante el arbitrio del microcontrolador de la tarjeta SIMMP-2 ejecutando software bajado desde una computadora anfitriona vía puerto serie, empleando para ello al manejador hexadecimal PUMMA, para más detalles acerca del uso del programador de memorias EPROM consultar el tema sobre el particular que se expone mas adelante en este capítulo o bien leer lo referente a esto en el capítulo que habla sobre la operación del programa PUMMA.EXE.

LOCALIZACIÓN DE COMPONENTES EN LA CMT SIMMP-2

En la Figura 2.2 se muestra una vista de planta de la CMT SIMMP-2, en dicha figura se aprecia la localización de los diversos componentes que integran la arquitectura de la tarjeta.

En la parte central se aprecia la localización del microcontrolador 68HC11F1 que es el corazón de SIMMP-2.

A la izquierda de la tarjeta aparecen los bloques CTPP1 y CTPP2 observándose a la izquierda de los mismos la ubicación de dos conectores, denominados con las letras A y B respectivamente, dichos conectores dan acceso al usuario a los distintos puertos paralelos programables correspondientes con los bloques mencionados y a algunas líneas importantes del propio microcontrolador tales como $\overline{\text{XIRQ}}$, $\overline{\text{IRQ}}$ y las líneas asociadas del puerto D del 68HC11F1; en el conector A aparecen líneas de puerto del bloque CTPP1 y en el conector B aparecen líneas de puerto asociadas con el CTPP2.

A la derecha de la tarjeta aparecen otros dos conectores, uno es denominado como conector C y el otro es el conector auxiliar de expansión, en este último el usuario tiene acceso a líneas control, de selección de puerto y de polarización; en el conector C se tiene acceso a líneas de puerto del propio microcontrolador; en la Figura 2.3 se muestra en detalle las líneas asociadas con cada uno de los postes que integran los conectores antes mencionados.

En la parte inferior izquierda se encuentra parte del hardware relacionado con el programador de memorias EPROM.

A la izquierda del microcontrolador se puede observar el botón de restablecimiento manual.

En la parte central inferior de la tarjeta se observa un conector de polarización, el mismo es de cuatro terminales, adjunto al mismo se ve la leyenda(5 0 P 6), indicando esto los postes por donde se han de conectar a la tarjeta SIMMP-2 la fuente de cinco volts (poste con el número cinco), la fuente que proporcionaría el voltaje de programación de las memorias EPROM (poste con la letra P) y la línea común de tierra de ambas fuentes mencionadas (poste con el guarismo cero); al poste con el guarismo seis no se le conecta nada y su uso se reserva para posibles mejoras futuras del programador de memorias EPROM.

En la parte inferior izquierda, entre el circuito que valida al bloque CTPP2 y el circuito MAX-232, se encuentra un conector de tres postes por donde ha de conectarse cable de enlace serie entre SIMMP-2 y una computadora anfitriona, en la Figura 2.4 se muestra el conexionado de el cable de enlace serie.

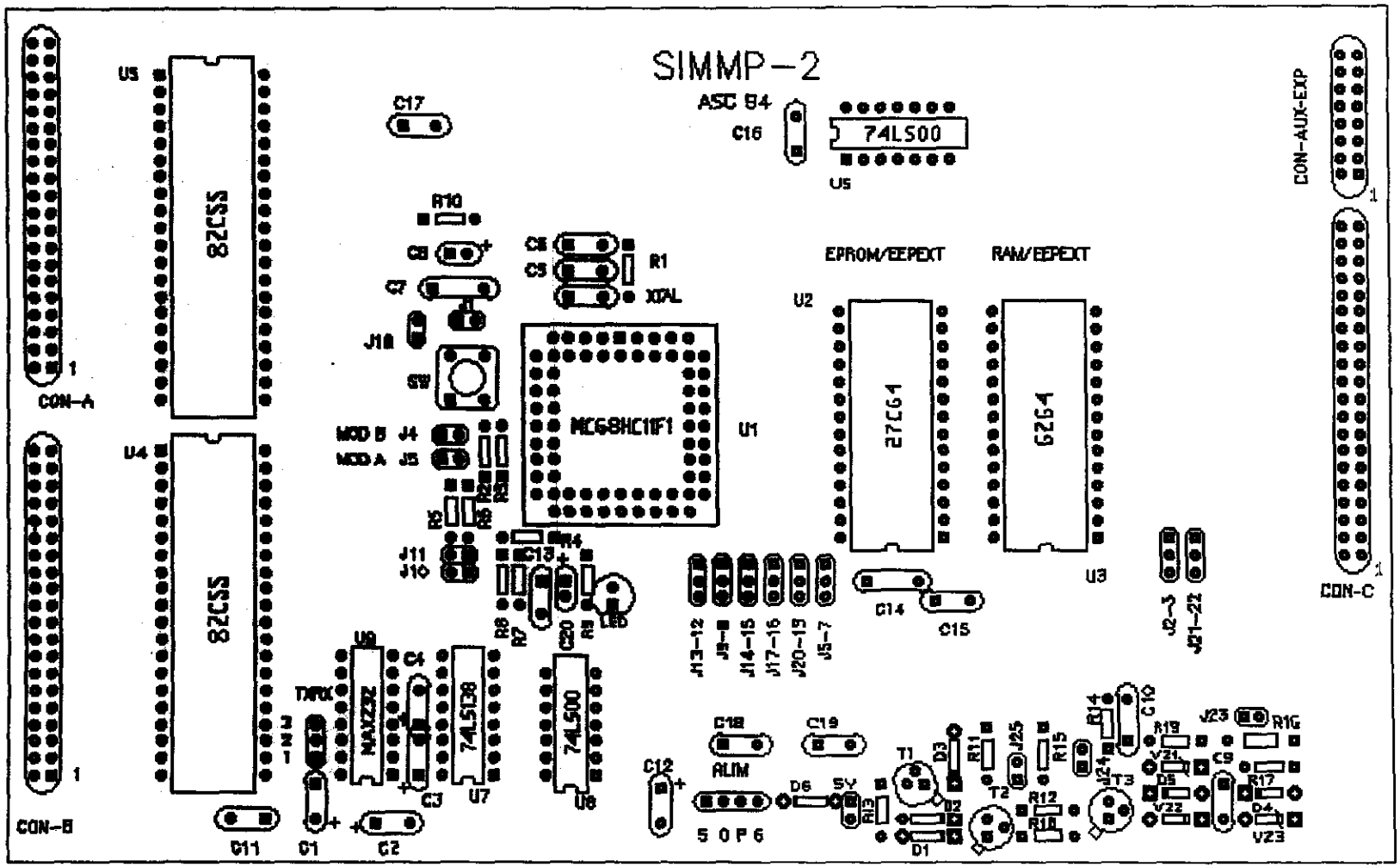


Figura 2.2.- Vista superior de la CMT SIMMP-2

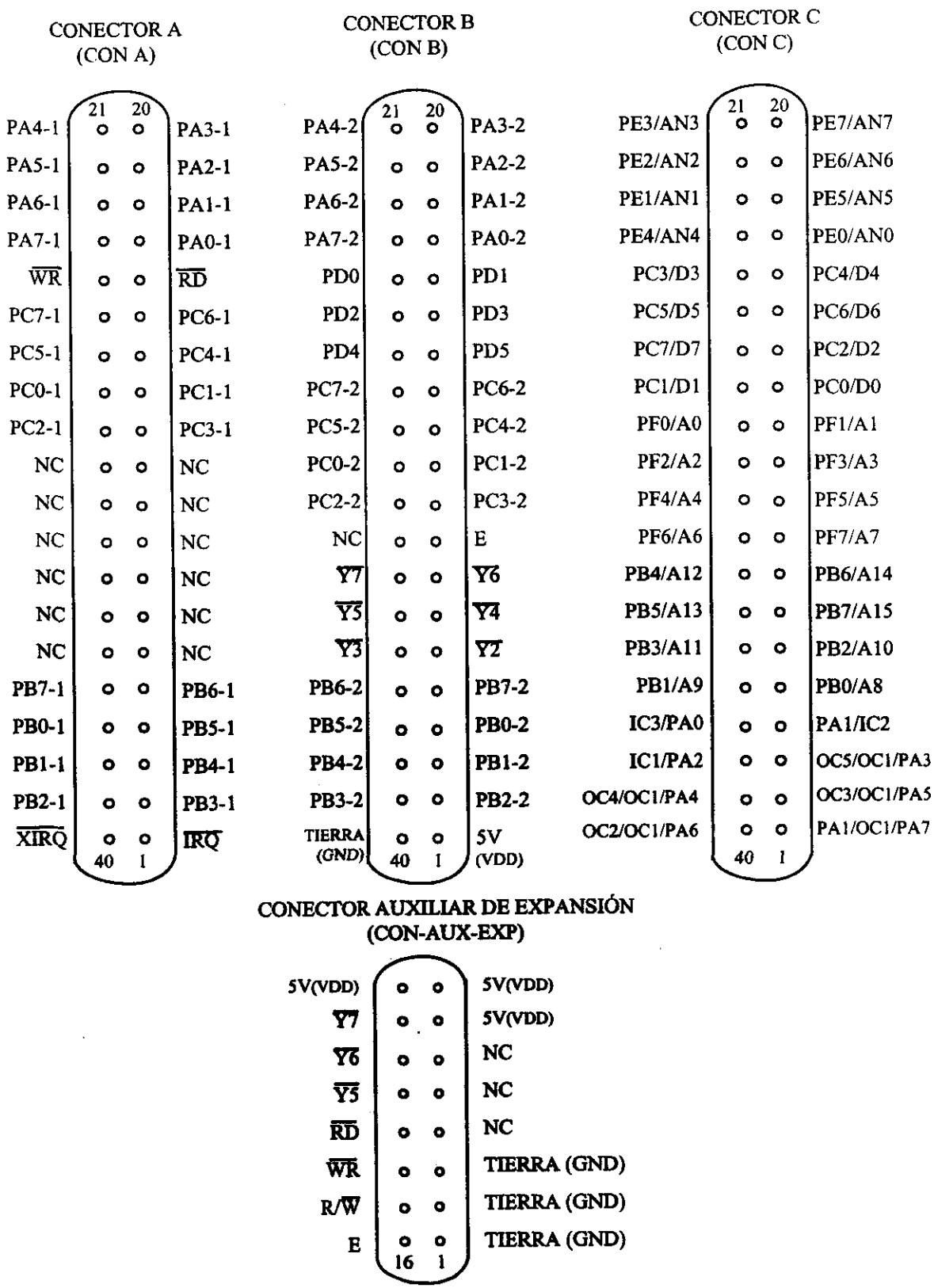


Figura 2.3.- Conectores de la CMT SIMMP-2.

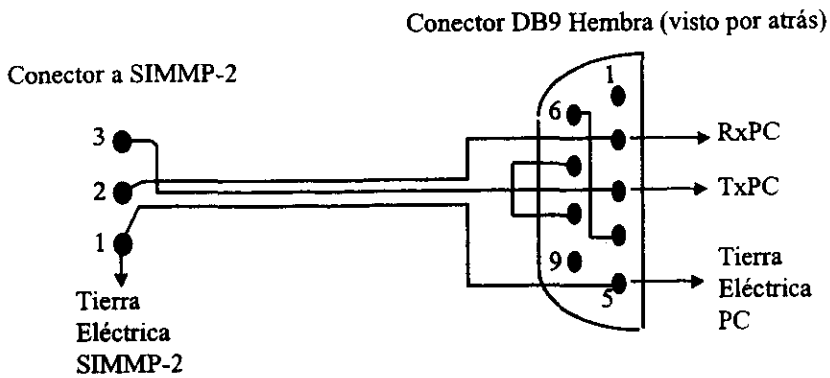


Figura 2.4.- Detalle de conexionado del cable de enlace serie.

Los puentes (jumpers) juegan un papel muy importante ya que con ellos se configuran distintas maneras de operar además de que con ellos se habilitan, para cada caso, las distintas posibilidades con que se cuenta para programar memorias EPROM de acuerdo con su tamaño. A continuación se describe en forma genérica la función de cada uno de los puentes y/o postes asociados a los mismos que existen en la tarjeta SIMMP-2.

a) Postes asociados con el puente J1. Uno de estos postes está conectado con la terminal de restablecimiento (RESET) del microcontrolador y el otro está conectado con el lado negativo (tierra) de la fuente de polarización que alimente a la tarjeta, por lo tanto, el puente J1 nunca deberá ser colocado y la presencia de los postes mencionados facilita al usuario conectar externamente a la tarjeta un botón de restablecimiento o algún circuito especial para estos fines.
b) Trío de Postes asociados con los puentes J2 y J3. Estos están colocados abajo a la derecha de la base de la memoria RAM y permiten la colocación de uno sólo de los puentes J2 o J3 que seleccionan si la memoria EPROM a programar es la 27C16, 27C32 o bien las memorias 27C64, 27C128, 27C256 o 27C512.
c) Postes para el puente J4. A estos postes ha de conectarse un puente, cuando se desea que sea bajo el nivel lógico asociado con la terminal MODB del microcontrolador, en otro caso no ha de conectarse el puente J4.
d) Postes para el puente J5. A estos postes ha de conectarse un puente, cuando se desea que sea bajo el nivel lógico asociado con la terminal MODA del microcontrolador, en otro caso no ha de conectarse el puente J5.
e) Trío de postes asociados con los puentes J6 y J7. A estos postes ha de conectarse ya sea el puente J6 o el puente J7, aunque para la versión actual del programador de memorias EPROM contenido en la tarjeta debe colocarse siempre el puente J6, posibles mejoras futuras del programador de memorias EPROM podrían llegar a requerir la colocación del puente J7 para este trío de postes.
f) Trío de postes asociados con los puentes J8 y J9. En estos postes se coloca el puente J9 para operación normal de lectura de la memoria EPROM, en caso de que se use el programador ha de conectarse a estos postes el puente J8.

g) Postes para el puente J10. Aquí debe conectarse el puente J10, para la versión actual del firmware residente en el microcontrolador de la tarjeta este puente no debe conectarse.

h) Postes para el puente J11. A estos postes se conecta el puente J11; el tener este puente conectado cuando el sistema se opere en modo expandido hace que el microcontrolador salte, después de un restablecimiento, a el origen de la memoria EPROM externa, en otro caso el microcontrolador pasaría a ejecutar firmware que le permite comunicarse con una computadora anfitriona que este ejecutando el manejador PUMMA, esto permite al usuario hacer desarrollo cuando se trabaje la tarjeta SIMMP-2 en modo expandido.

i) Trío de postes asociado con los puentes J12 y J13. A estos postes ha de conectarse el puente J12 o el puente J13; el puente J12 se pondría cuando se tuviera colocada en la base de la EPROM una memoria 27C16, 27C32, 27C64, o 27C128; en caso de que la memoria EPROM sea la 27C256 o la 27C512 el puente J13 es el que debe ser puesto.

j) Trío de postes asociado con los puentes J14 y J15. A estos postes se conecta el puente J14 o el puente J15; el puente J15 se conecta cuando en la base de la memoria EPROM está colocada para programación una memoria 27C512, en otro caso se debe colocar el puente J14.

k) Trío de postes asociado con los puentes J16 y J17. Aquí ha de conectarse el puente J17 cuando se este programando una EPROM 27C32 o 27C512, al programar otro tipo de EPROM se debe colocar aquí el puente J16.

a) Postes asociados con el puente J18. Este puente conecta a la terminal VRH del microcontrolador el mismo voltaje de cinco volts que polariza a la tarjeta; en caso de que el usuario desee una referencia de voltaje más precisa, deberá desconectar el puente J18 y conectar la terminal positiva de la fuente que proporcione la referencia de voltaje a el poste ligado, vía una red RC, con la entrada VRH del microcontrolador; la terminal negativa de la fuente de referencia deberá conectarse con la referencia de cero volts (tierra) de la tarjeta SIMMP-2, ya que en la arquitectura SIMMP-2, la entrada VRL del microcontrolador está conectada a tierra, ver la Figura 2.5. Cabe recordar aquí que el voltaje aplicado en la entrada VRH del microcontrolador, es empleado por el mismo, no sólo como referencia para el convertidor analógico digital sino también como voltaje de entrada para la lógica de programación de la memoria EEPROM interna, por lo tanto, si el usuario desea programar dicha memoria, el puente J18 deberá estar colocado en su lugar. El poste donde el usuario ha de conectar una referencia externa de voltaje a la entrada VRH, previa desconexión del puente J18, es el que se encuentra casi en línea con los postes para el puente J1, véase la Figura 2.6.

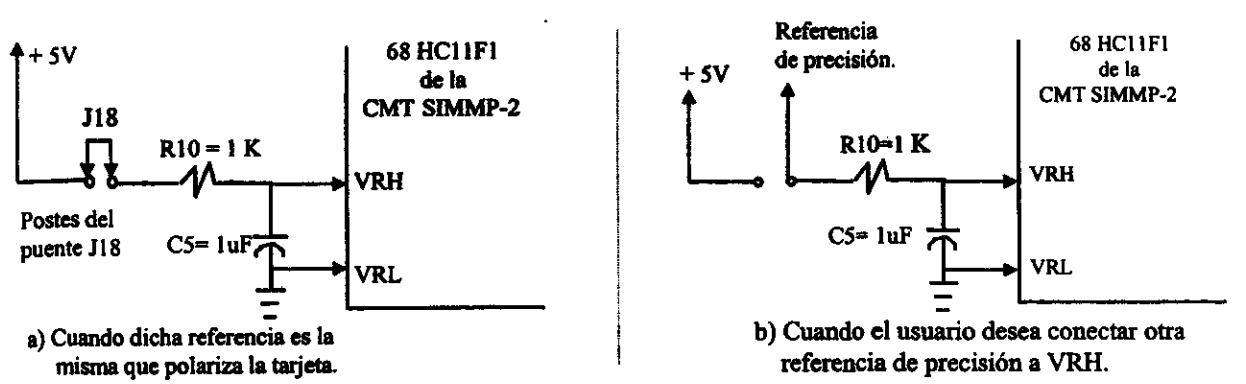


Figura 2.5.- Esquema eléctrico de conexión a la entrada VRH del microcontrolador:

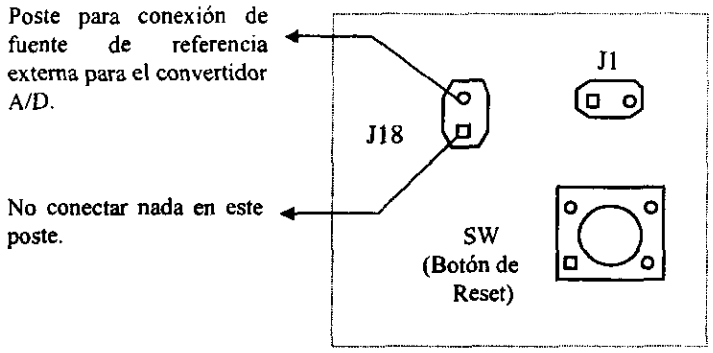


Figura 2.6.- Localización en la CMT SIMMP-2 del poste para conexión de fuente de referencia externa, cuando se desconecta el puente J18.

a) Trío de postes asociado con los puentes J19 y J20. Aquí debe colocarse el puente J19 cuando la memoria RAM colocada en la base correspondiente es de 32k, en caso de que la misma sea de 8k ha de colocarse el puente J20.
b) Trío de Postes asociados con los puentes J21 y J22. Estos están colocados abajo a la derecha de la base de la memoria RAM y permiten la colocación de uno sólo de los puentes J21 o J22 que seleccionan si la memoria EPROM a programar es la 27C16, 27C32 o bien las memorias 27C64, 27C128, 27C256 o 27C512.
c) Postes para el puente J23. El uso del puente J23 se reserva para posibles mejoras futuras del programador de memorias EPROM; por lo tanto, no es necesaria su conexión en la tarjeta SIMMP-2, los postes mencionados pueden ser empleados como puntos de prueba para verificar el funcionamiento del programador de memorias EPROM, ya que en ellos aparece el voltaje de programación cuando se habilita esta facilidad desde la computadora anfitriona que maneje la tarjeta, empleando el manejador PUMMA, en un momento dado.
d) Postes para el puente J24. En estos postes se coloca el puente J24 cuando se desee programar una EPROM cuyo voltaje de programación sea de 12.5 volts; en caso de que la memoria a programar requiera más voltaje para su programación el puente mencionado aquí no debe ser puesto.
e) Postes para el puente J25. Aquí se coloca el puente J25, que deberá estar siempre puesto, para una correcta operación del programador de memorias EPROM y de otras facilidades con las que cuenta la tarjeta SIMMP-2.

MAPAS DE MEMORIA CON LOS QUE PUEDE OPERAR LA CMT SIMMP-2

La CMT SIMMP-2 puede operar con diferentes mapas de memoria; cuando la CMT SIMMP-2 opera en modo single-chip o boot-strap, los mapas correspondientes son los naturales del microcontrolador 68HC11F1, al operar el mismo en los modos mencionados, véanse las Figuras 2.7 y 2.8.

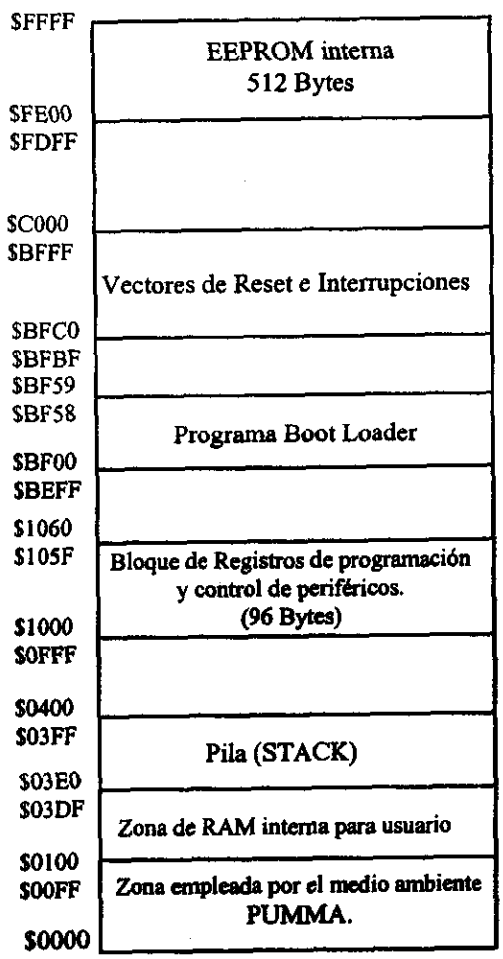


Figura 2.7.- Mapa de memoria de la CMT SIMMP-2 operando en modo Boot-Strap.

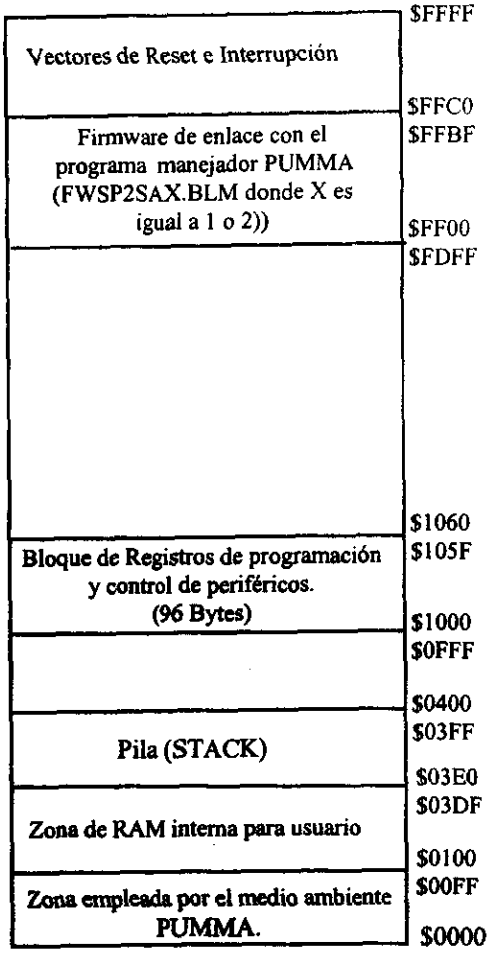


Figura 2.8.- Mapa de memoria de la CMT SIMMP-2 operando en modo Single-Chip.

Al operar en los modos expandido o TEST el microcontrolador 68HC11F1 cuenta con líneas de selección de memoria y puertos que pueden ser configurados por software, esto hace que en estos casos puedan ser configurados diversos mapas de memoria. Cada mapa es configurado por hardware (colocando puentes) y software (programando firmware residente en la tarjeta); para todos estos mapas, existe un submapa de puertos, si se desea conocer más detalles acerca del mismo, consultar el tema sobre este particular, en este mismo capítulo; a continuación se

describe la manera de configurar diferentes mapas de memoria cuando la tarjeta SIMMP-2 opere en los modos expandido o TEST; en la Tabla 2.1 se detalla en forma resumida la configuración de puentes y firmware residente requerido para cada uno de dichos mapas.

Mapa EA

Para este mapa se tiene el siguiente perfil: Operación en modo expandido con 8k de RAM externa, 1k de RAM interna, 7.5k de EPROM y 0.5k de EEPROM interna con firmware SP2EA residente. Para configurar este mapa se deben colocar los siguientes puentes: J3, J8 o J9, J12, J14, J16, J20 y J22, además de que uno de los archivos FWSP2EA1.BLM o FWSP2EA2.BLM deberá haber sido programado en la EEPROM interna a partir de su origen; para el caso del primer archivo mencionado el área de vectores de RESET e interrupción está programada únicamente con los vectores correspondientes a los tres tipos de RESET con los que cuenta el 68HC11 (apuntando estos al origen de la EEPROM interna), dejando al usuario la posibilidad de programar, empleando el manejador PUMMA, los vectores de interrupción que el mismo pudiera llegar a requerir al realizar una determinada aplicación; en lo que toca al segundo archivo mencionado el vector de RESET principal apunta al origen de la EEPROM interna y todos los demás vectores de RESET e interrupción apuntan a la dirección de la página cero, siendo éstas las mismas que las que corresponden a los vectores asociados con el modo boot-strap, lo anterior permitiría al usuario trabajar con el concepto de seudovector de interrupción al programar sus aplicaciones en modo expandido; en la Figura 2.9A se aprecia el mapa de memoria EA; los requerimientos en cuanto a chips de memoria para este mapa son: RAM estática de 8k en la base correspondiente, EPROM de 8k en su respectiva base. Cabe señalar que este mapa es el que viene configurado de origen en la CMT SIMMP-2 con el archivo FWSP2EA1.BLM en la EEPROM.

Mapa EB

Para este mapa se tiene el siguiente perfil: Operación en modo expandido con 28.90625k de RAM externa (29600 bytes), 1k de RAM interna, 31.5k de EPROM y .5k de EEPROM interna con firmware SP2EB residente. Para configurar este mapa se deben colocar los siguientes puentes: J3, J8 o J9, J13, J14, J16, J19 y J22, además de que uno de los archivos FWSP2EB1.BLM o FWSP2EB2.BLM deberá haber sido programado en la EEPROM interna a partir de su origen; para el caso del primer archivo mencionado el área de vectores de RESET e interrupción está programada únicamente con los vectores correspondientes a los tres tipos de RESET con los que cuenta el 68HC11 (apuntando estos al origen de la EEPROM interna), dejando al usuario la posibilidad de programar, empleando el manejador PUMMA, los vectores de interrupción que el mismo pudiera llegar a requerir al realizar una determinada aplicación; en lo que toca al segundo archivo mencionado el vector de RESET principal apunta al origen de la EEPROM interna y todos los demás vectores de RESET e interrupción apuntan a las direcciones de la página cero, siendo estas las mismas que las que corresponden a los vectores asociados con el modo boot-strap, lo anterior permitiría al usuario trabajar con el concepto de seudovector de interrupción al programar sus aplicaciones en modo expandido; en la Figura 2.9B se aprecia el mapa de memoria EB; los requerimientos en cuanto a chips de memoria para este mapa son: RAM estática de 32k en la base correspondiente, EPROM de 32k en su respectiva base.

Mapa EC

Para este mapa se tiene el siguiente perfil: Operación en modo expandido con 28.90625k de RAM externa (29600 bytes), 1k de RAM interna, 8k de EPROM (con direcciones que accesan la misma localidad separadas 8k <espejos>) y .5k de EEPROM interna con firmware SP2EB (el mismo que el correspondiente al mapa EB) residente. Para configurar este mapa se deben colocar los siguientes puentes: J3, J8 o J9, J12, J14, J16, J19 y J22, además de que uno de los archivos FWSP2EB1.BLM o FWSP2EB2.BLM deberá haber sido programado en la EEPROM interna a partir de su origen; para el caso del primer archivo mencionado el área de vectores de RESET e interrupción está programada únicamente con los vectores correspondientes a los tres tipos de RESET con los que cuenta el 68HC11 (apuntando estos al origen de la EEPROM interna), dejando al usuario la posibilidad de programar, empleando el manejador PUMMA, los vectores de interrupción que el mismo pudiera llegar a requerir al realizar una determinada aplicación; en lo que toca al segundo archivo mencionado el vector de RESET principal apunta al origen de la EEPROM interna y todos los demás vectores de RESET e interrupción apuntan a las direcciones de la página cero, siendo estas las mismas que las que corresponden a los vectores asociados con el modo boot-strap, lo anterior permitiría al usuario trabajar con el concepto de seudovector de interrupción al programar sus aplicaciones en modo expandido; en la Figura 2.9B se aprecia el mapa de memoria EB; los requerimientos en cuanto a chips de memoria para este mapa son: RAM estática de 32k en la base correspondiente, EPROM de 8k en su respectiva base.

Mapa TA

Para este mapa se tiene el siguiente perfil: Operación en modo TEST con 8k de RAM externa, 1k de RAM interna, 8k de EPROM y .5k de EEPROM interna con firmware SP2TA residente. Para configurar este mapa se deben colocar los siguientes puentes: J3, J8 o J9, J12, J14, J16, J20 y J22, además de que uno de los archivos FWSP2TA1.BLM o FWSP2TA2.BLM deberá haber sido programado en la EEPROM interna a partir de su origen; para el caso del primer archivo mencionado el área de vectores de RESET e interrupción está programada únicamente con los vectores correspondientes a los tres tipos de RESET con los que cuenta el 68HC11 (apuntando estos al origen de la EEPROM interna), dejando al usuario la posibilidad de programar, empleando el manejador PUMMA, los vectores de interrupción que el mismo pudiera llegar a requerir al realizar una determinada aplicación; en lo que toca al segundo archivo mencionado el vector de RESET principal apunta al origen de la EEPROM interna y todos los demás vectores de RESET e interrupción apuntan a las direcciones de la página cero, siendo estas las mismas que las que corresponden a los vectores asociados con el modo boot-strap, lo anterior permitiría al usuario trabajar con el concepto de seudovector de interrupción al operar el microcontrolador en modo TEST ; en la Figura 2.10A se aprecia el mapa de memoria TA; los requerimientos en cuanto a chips de memoria para este mapa son: RAM estática de 8k en la base correspondiente, EPROM de 8k en su respectiva base.

Para todos los mapas posibles se observa que la página cero (direcciones de la $0000 a la $00FF), está ocupada por lo que se denomina como ambiente PUMMA que es empleado como enlace con la computadora anfitriona cuando se trabaja con el sistema anfitrión-destino para

desarrollo; por lo tanto, en tal caso el usuario no deberá modificar localidades de memoria que se encuentren en la página cero antes de las direcciones correspondientes a los seudovectores de interrupción propios del modo boot-strap.
Para los mapas descritos en los párrafos anteriores es importante señalar que cuando se opere en modo TEST el nibble alto del registro CONFIG deberá estar programado con la palabra binaria 1011, en otro caso la dicha palabra deberá ser 1111, el registro CONFIG tiene asociada la dirección $103F y es una localidad EEPROM, para programarlo se puede usar el manejador PUMMA invocando la opción de programación de la EEPROM interna desde teclado, escogiendo como dirección a programar la mencionada anteriormente, para más detalles acerca del registro CONFIG se recomienda consultar el manual técnico sobre el 68HC11 editado por MOTOROLA, en lo que toca a el empleo de PUMMA para programación de la EEPROM interna puede consultarse el capítulo que describe este punto o la ayuda en línea del programa.

Tabla 2.1 Configuración de puentes y firmware residente para los mapas de memoria de la CMT SIMMP-2.

Mapa	Puentes en CMT SIMMP-2	Firmware Residente
EA	J3, J8 o J9, J12, J14, J16, J20 y J22	FWSP2EA1 o FWSP2EA2
EB	J3, J8 o J9, J13, J14, J16, J19 y J22	FWSP2EB1 o FWSP2EB2
EC	J3, J8 o J9, J12, J14, J16, J19 y J22	FWSP2EB1 o FWSP2EB2
TA	J3, J8 o J9, J12, J14, J16, J20 y J22	FWSP2TA1 o FWSP2TA2

Para cada uno de los mapas descritos anteriormente asociados con la operación en modo expandido o TEST, el firmware residente requerido es cargado en la EEPROM interna del microcontrolador, quedando en dicha memoria varias localidades libres para uso del usuario, para los mapas asociados con la operación en modo expandido las correspondientes direcciones van de la $FEA0 a la $FFBF, en lo que toca a el mapa para operación en modo TEST las direcciones de las localidades libres van de la $BEA0 a la $BFBF.
SUBMAPA DE PUERTOS.- El submapa de puertos está definido en un intervalo de 1k (de la dirección $1800 a la $1BFF) dividido en ocho subintervalos de 128 direcciones cada uno, al invocar una dirección de puerto en un subintervalo determinado se verifica en nivel bajo una de ocho líneas de habilitación de puerto, en la Figura 2.11 se muestra el submapa de puertos, para más detalles acerca de las habilitaciones y direcciones asociadas pueden consultarse en este mismo capítulo, los temas sobre el paginador de puertos y los bloques CTPP1 y CTPP2.

SUBMAPA DE PUERTOS ALTERNO.- En todos los mapas de memoria asociados con los modos TEST y expandido se aprecia la existencia de un submapa de puertos alterno, definido en un intervalo de 1k (direcciones de la $1C00 a la $1FFF), si el usuario lo requiriera podría conectar un 74LS138 externo que generaría en sus ocho salidas líneas de salida, habilitaciones de puerto asociadas cada una de ellas con sendos subintervalos de 128 direcciones cada uno, en la Figura 2.12 se muestra como hacer esto, así como también los subintervalos que verificarían cada línea de habilitación; en la práctica se ha visto que con las líneas de habilitación propias del submapa de puertos normal (direcciones de la $1800 a la 1BFF) pueden conectarse, mediante lógica de enlace, diversos puertos externos a la CMT SIMMP-2.

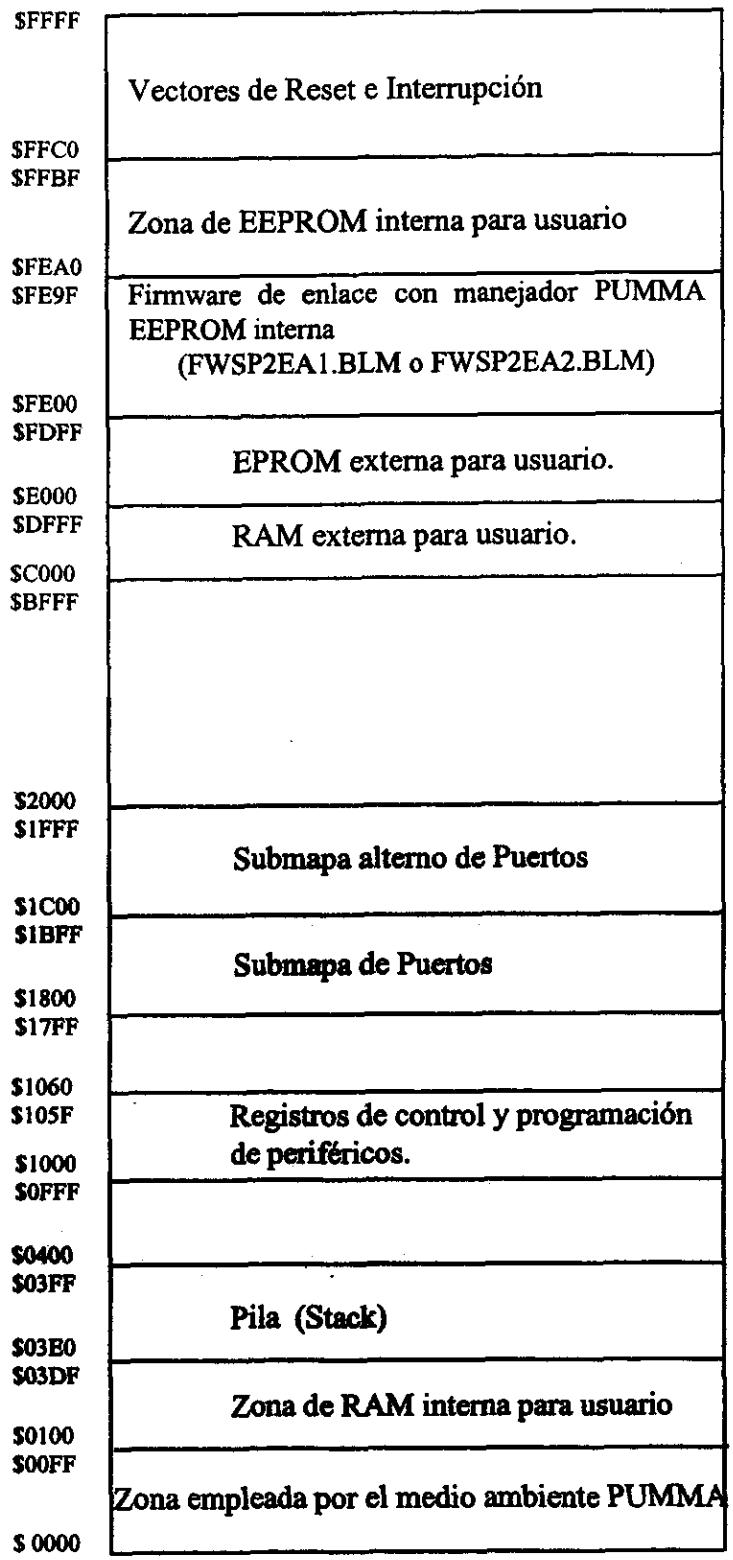


Figura 2.9A.- Mapa de memoria EA (operación en modo expandido)

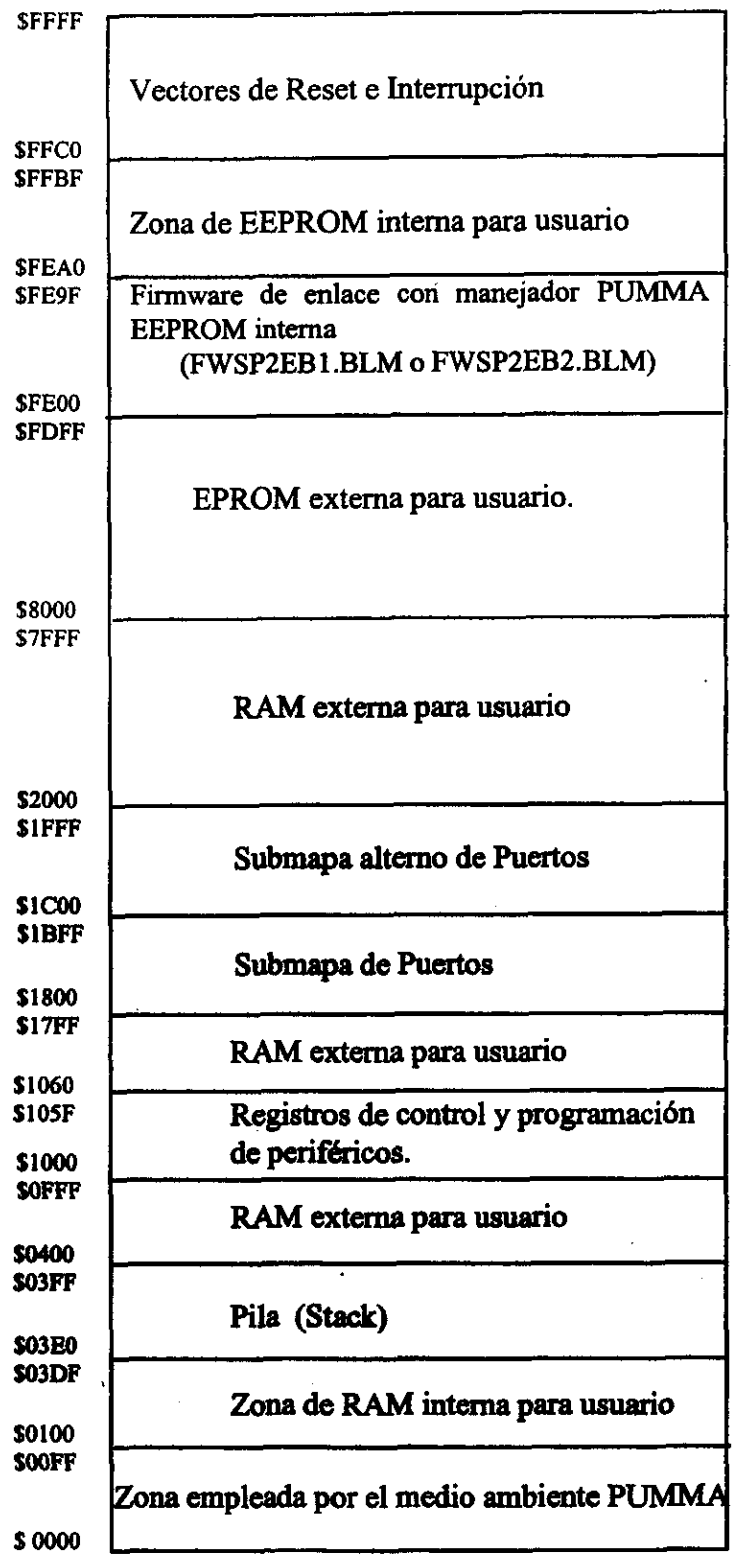


Figura 2.9B.- Mapa de memoria EB (operación en modo expandido)

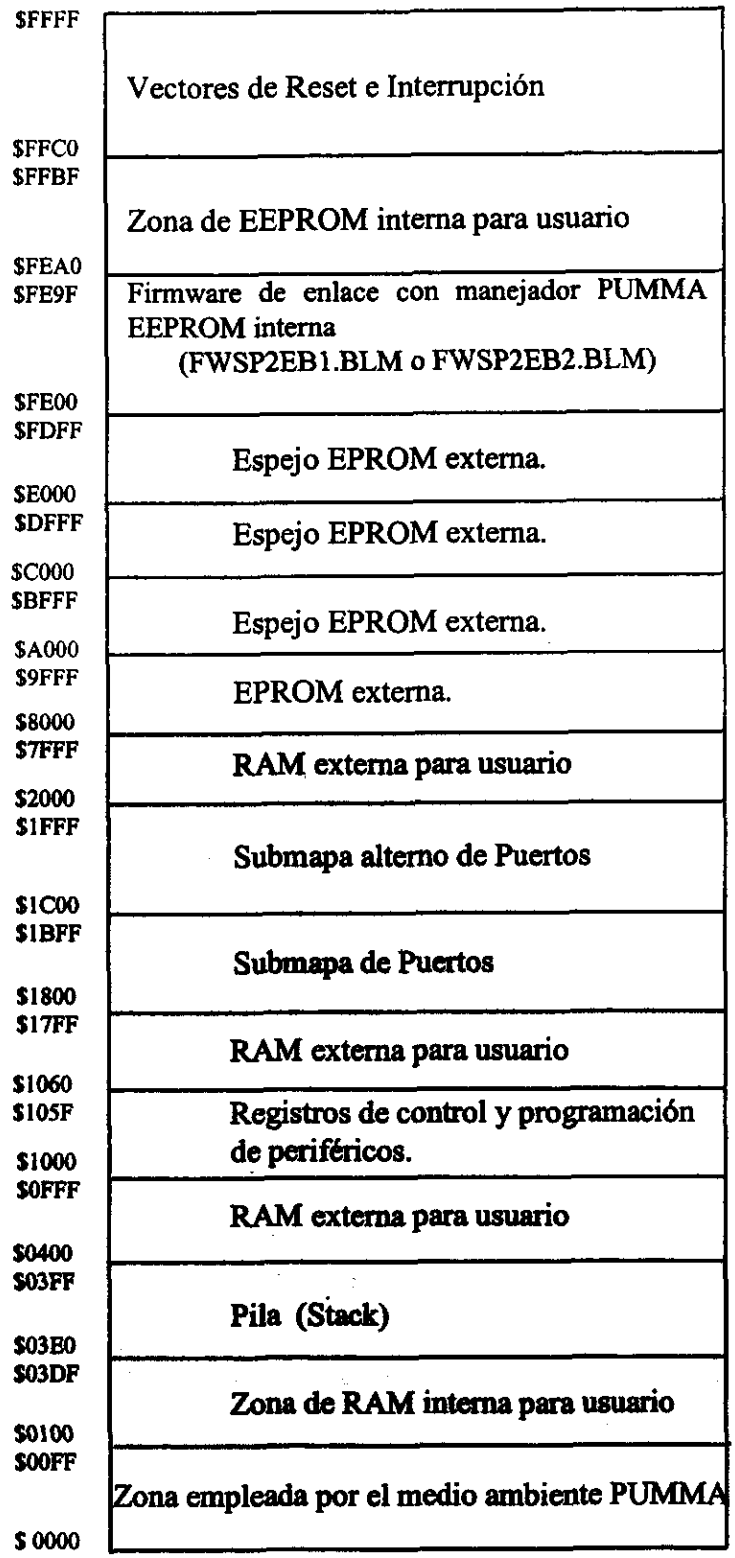


Figura 2.9C Mapa de memoria EC (operación en modo expandido)

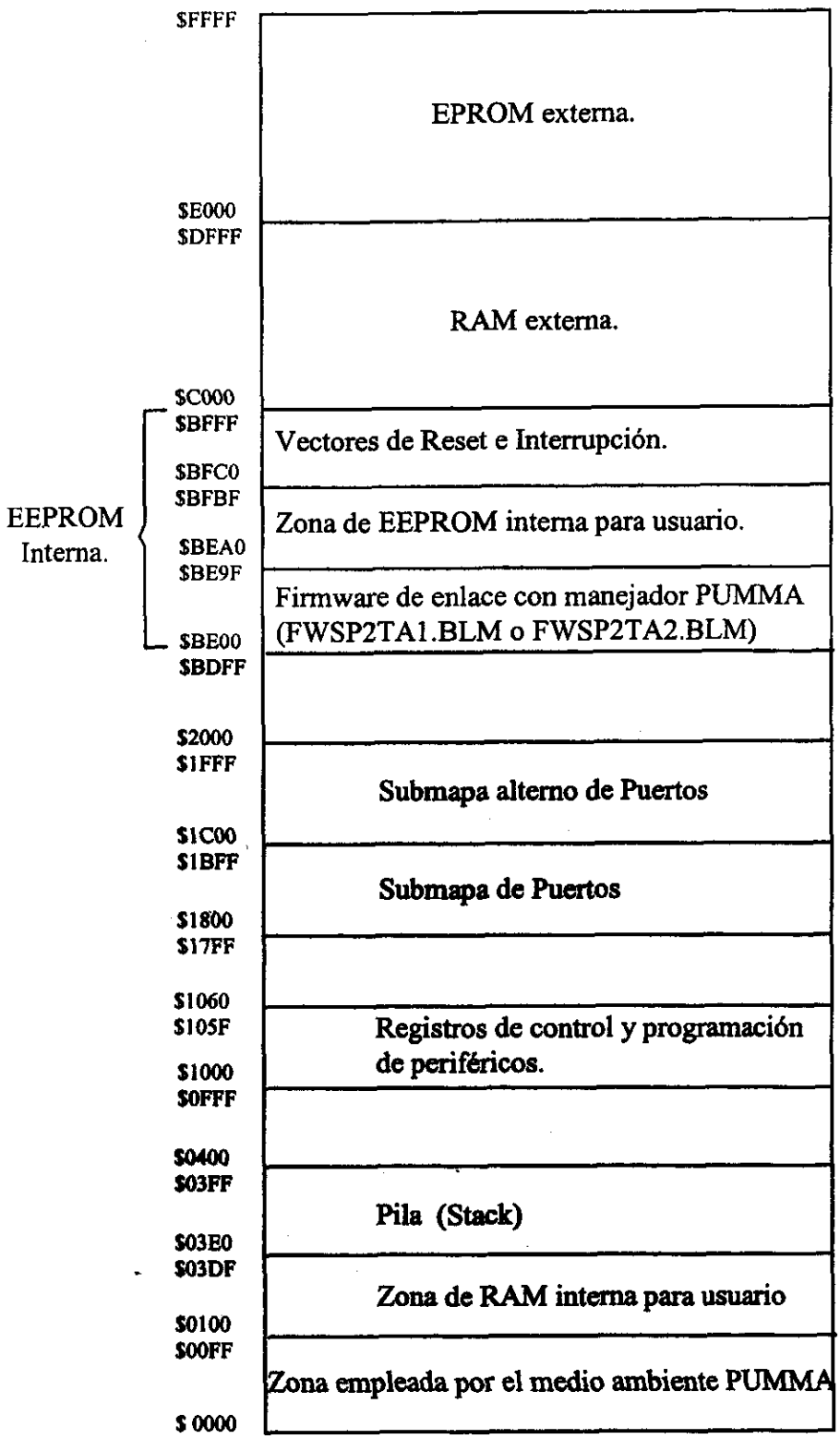


Figura 2.10.- Mapa de memoriaTA (operación en modo TEST).

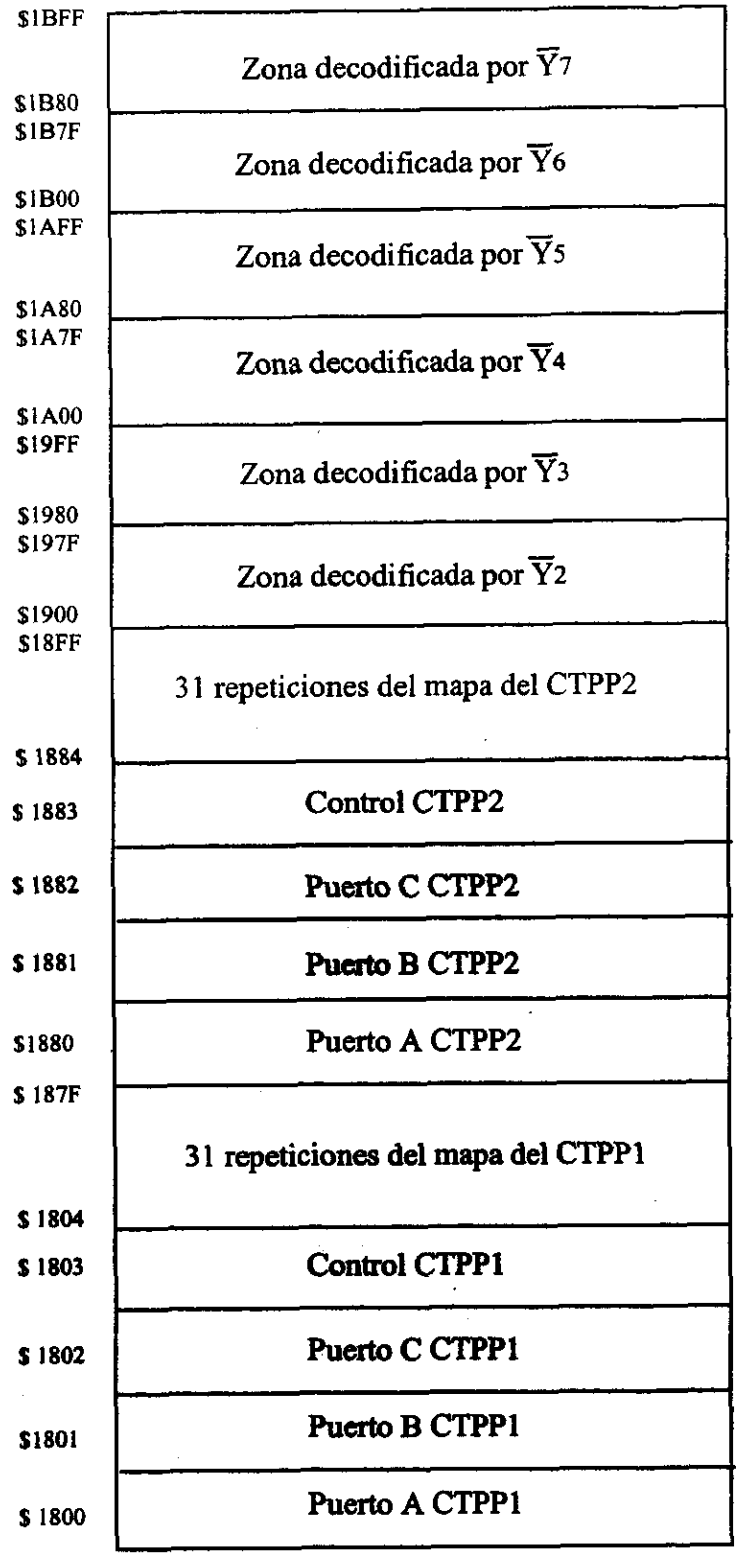


Figura 2.11.- Submapa de puertos de la CMT SIMMP-2 operando en modo expandido o TEST.

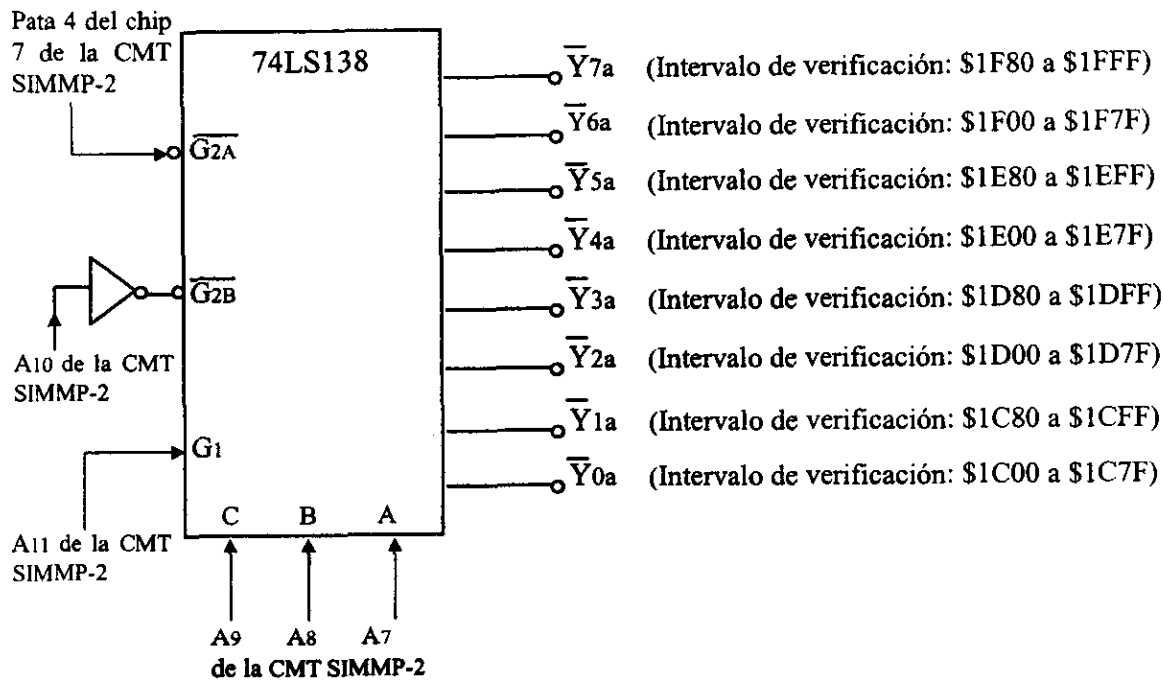


Figura 2.12.- Conexión de un decodificador 74LS138 externo para validar el submapa de puertos alterno con ocho subintervalos de verificación.

Si la tarjeta SIMMP-2 no contara con los chips que validan a los bloques CTPP1 y CTPP2 el usuario podría añadir puertos paralelos de salida cuando se opere en modo expandido o TEST, para ello se podrían emplear chips MOTOROLA que sirvieran para tal fin, tales como el PIA 6821, siempre y cuando los mismos fueran compatibles en velocidad con la CMT SIMMP-2; una manera económica de agregar puertos paralelos de entrada o salida a la arquitectura SIMMP-2 consiste en usar para tal fin a registros 74LS373 o 74LS374 a continuación se describe como hacer esto.

CONEXIÓN DE UN PUERTO DE SALIDA EXTERNO EMPLEANDO UN 74LS374.- En el caso de que la tarjeta SIMMP-2 no contara con los chips que validan a los bloques CTPP1 y CTPP2 el usuario podría añadir puertos paralelos de salida cuando se opere en modo expandido o TEST, para ello se podría emplear un registro 74LS374 que cuenta con señal de captura por flanco de subida (para más detalles consultar un manual de circuitos TTL), en la Figura 2.13 se muestra un esquema de como hacer esto, dado que la línea $\overline{Y7}$ de habilitación se verifica al invocarse direcciones que van de la $1B80 a la $1BFF, el puerto de salida mostrado en la Figura 2.13 tiene como dirección asociada a cualquiera de las contenidas en el intervalo antes mencionado; si se deseara reducir la redundancia de direcciones la señal de captura de dato para el 74LS374 se tendría que obtener de un circuito combinacional cuya entrada fuera la propia señal $\overline{Y7}$ y algunas líneas de dirección del microcontrolador, existen desde luego, de acuerdo

con la o las direcciones deseadas para el puerto de salida en cuestión, múltiples maneras de realizar tal lógica de paginación.

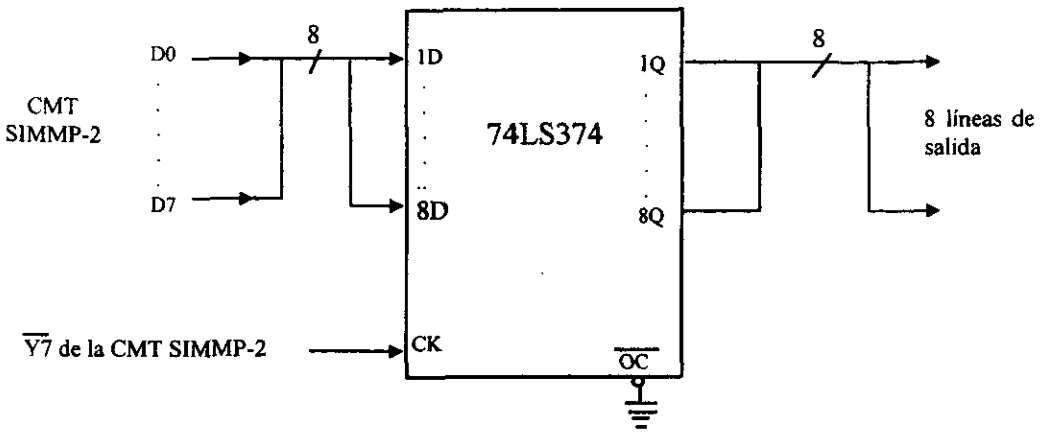


Figura 2.13.- Conexión de un puerto de salida externo a la tarjeta SIMMP-2, empleando un registro 74LS374, validado en cualquiera de las direcciones en el intervalo $1B80 a $1BFF.

CONEXIÓN DE UN PUERTO DE ENTRADA EXTERNO EMPLEANDO UN 74LS373.- En caso de que la tarjeta SIMMP-2 no contara con los chips que validan a los bloques CTPP1 y CTPP2 el usuario podría añadir puertos paralelos de entrada cuando se opere en modo expandido o TEST, para ello se podría emplear un registro 74LS373 que cuenta con señal de habilitación de tercer estado con nivel bajo de verificación (para más detalles consultar un manual de circuitos TTL), en la Figura 2.14 se muestra un esquema de como hacer esto, dado que la línea Y6 de habilitación se verifica al invocarse direcciones que van de la $1B00 a la $1B7F, el puerto de salida mostrado en la Figura 2.14 tiene como dirección asociada a cualquiera de las contenidas en el intervalo antes mencionado; si se deseara reducir la redundancia de direcciones la señal de habilitación de salida para el 74LS373 se tendría que obtener de un circuito combinacional cuya entrada fuera la propia señal Y6 y algunas líneas de dirección del microcontrolador, existen desde luego, de acuerdo con la o las direcciones deseadas para el puerto de salida en cuestión, múltiples maneras de realizar tal lógica de

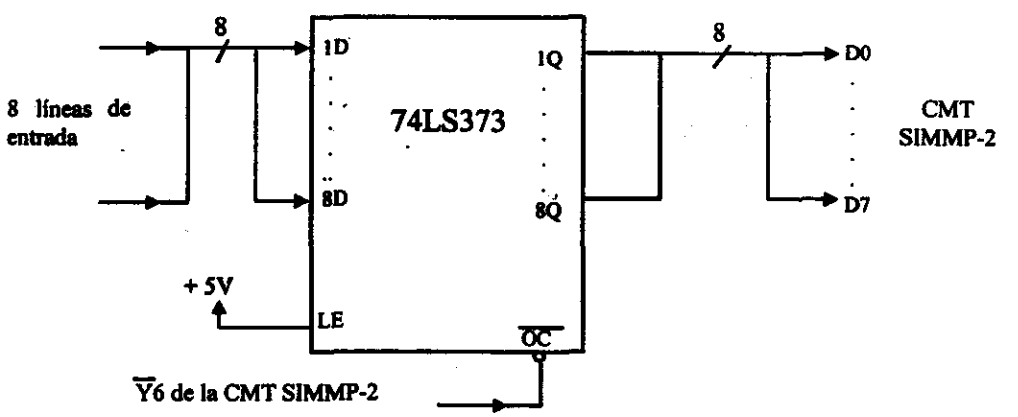


Figura 2.14.- Conexión de un puerto de entrada externo a la tarjeta SIMMP-2, empleando un registro 74LS373, validado en cualquiera de las direcciones en el intervalo $1B00 a $1B7F.

paginación.

PROGRAMADOR DE MEMORIAS EPROM CONTENIDO EN LA CMT SIMMP-2

La CMT SIMMP-2 incorpora en su arquitectura a un programador de memorias EPROM que requiere para su operación que la CMT SIMMP-2 esté ligada vía serie con una computadora anfitriona que esté ejecutando el manejador hexadecimal PUMMA; la memoria que se desee programar debe ser colocada en la base que para tal fin existe en la tarjeta SIMMP-2, que deberá estar configurada para operación en modo boot-strap, las EPROM que se pueden programar son las siguientes: 2716 o 27C16, 2732 o 27C32, 2764 o 27C64, 27128 o 27C128, 27256 o 27C256 y 27512 o 27C512; cuando se programen memorias 2716, 27C16, 2732 o 27C32 se deberán insertar en la base de modo que la pata uno de la memoria coincida con la pata tres de la base y la pata 12 de la memoria coincida con la pata 14 de la base véase la Figura 2.15.

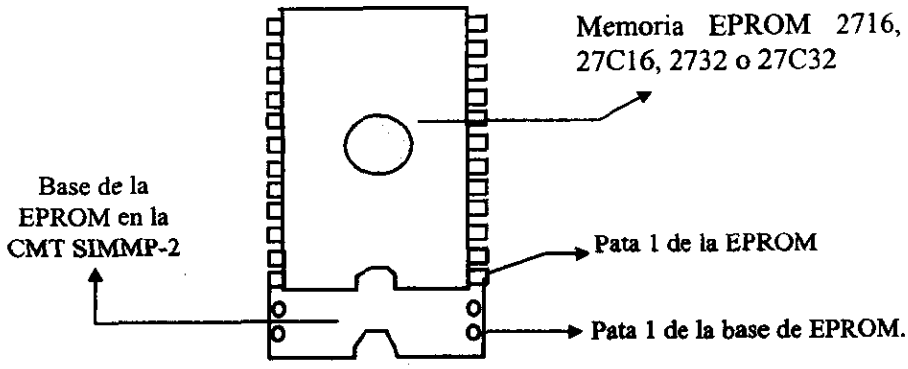


Figura 2.15.- Inserción de memorias EPROM 2716, 27C16, 2732 y 27C32 en la base respectiva de la CMT SIMMP-2.

El programador puede aceptar como origen de la información a colocar en la EPROM ya sea a una lista de bytes que el usuario introduzca desde el teclado de la computadora anfitriona o a la información contenida en un archivo de tipo BLM o LEM que son los que maneja de modo natural el manejador hexadecimal PUMMA (para más información consultar el capítulo referente al manejador PUMMA), si el usuario deseara programar un archivo S19 deberá antes transformarlo a un archivo BLM empleando para ello la opción dos del menú de manejo de disco del programa PUMMA, en caso de que el archivo a programar tenga el formato INTEL-HEX el usuario podrá pasarlo al formato S19, empleando para ello al programa HEXS19.EXE, para su posterior transformación a BLM como se ha descrito anteriormente y proceder a programarlo en la EPROM.

Para las memorias 2716, 27C16, 2764, 27C64, 27128, 27C128, 27256, y 27C256 cada byte es programado y verificado sucesivamente, en lo que toca a las memorias 2732, 27C32, 27512 y 27C512 se debe hacer la programación completa para después proceder a hacer la verificación empleando opciones del manejador PUMMA, esto se debe a características propias de las memorias mencionadas.

En caso de que la memoria sea de tipo 27C16 o 2716 PUMMA preguntará al usuario si la memoria es antigua o de tipo 27C16B ya que la mayoría de las EPROM de 2k antiguas se programan con un pulso con nivel de verificación alto, en caso de que el nivel de verificación

del pulso de programación sea bajo el usuario deberá indicar a PUMMA que desea programar una memoria 27C16B, para saber el tipo de pulso de programación de una memoria de este tipo se recomienda consultar la hoja de especificaciones técnicas de la misma.
En la actualidad el voltaje mas usual para programar memorias EPROM es 12.5 volts aunque para memorias fabricadas hace varios años tal voltaje podría ser mayor (de 21 a 24 volts), esto está contemplado en el diseño del programador de la tarjeta SIMMP-2, pudiendo ser configurado colocando o quitando puentes e indicándolo al manejador PUMMA cuando este lo requiera.
Antes de proceder a la programación de una EPROM el usuario deberá tener dispuesto lo siguiente:

1) Fuente que proporcionará el voltaje de programación de la EPROM, que deberá estar calibrada a un voltaje que exceda en aproximadamente 1.5 volts al voltaje nominal requerido.
2) Fuente de cinco volts para polarizar a la tarjeta SIMMP-2.
3) Si se trata de una EPROM 2716 o 27C16 se deberá saber de antemano el nivel de verificación del pulso de programación.
4) Si la programación ha de hacerse desde disco el archivo BLM correspondiente deberá estar ya generado.

Para cada tipo de EPROM de las manejadas por la CMT SIMMP-2 existe una configuración de puentes para que las mismas puedan ser leídas o programadas, en las Tablas 2.2 y 2.3 se muestran tales configuraciones.

Tabla 2.2.- Configuración de puentes para leer memorias EPROM en la CMT SIMMP-2.

Memoria	Puentes Colocados
27C16 2716	* J2, J9/8, J12, J14, J16, J21
27C32 2732	* J2, J9/8, J12, J14, J16, J22
27C64 2764	* J3, J9/8, J12, J14, J16, J22
27C128 27128	* J3, J9/8, J12,J14, J16, J22
27C256 27256	* J3, J9/8, J13,J14, J16, J22
27C512 27512	* J3, J9/8, J13,J15, J16, J22

*** Si está colocado J8, la terminal P de la CMT SIMMP-2 debe estar desconectada, preferentemente colocar J9.**

Tabla 2.3.-Configuración de puentes para programar memorias EPROM en la CMT SIMMP-2.

Memoria	Puentes Colocados
27C16 2716	J2, J8, J12, J14, J16, J21
27C32 2732	J2, J8, J12, J14, J16, J22
27C64 2764	J3, J8, J12, J14, J16, J22
27C128 27128	J3, J8, J12, J14, J16, J22
27C256 27256	J3, J8, J13, J14, J16, J22
27C512 27512	J3, J8, J13, J15, J17, J22

En la Figura 2.16 se muestra un arreglo para programar memorias EPROM empleando a la tarjeta SIMMP-2.

A continuación se describen los pasos a seguir para llevar a cabo la programación de una EPROM refiriéndose tanto al arreglo mostrado en la Figura 2.16 como a cosas propias del manejador PUMMA (consultar de ser necesario el capítulo sobre la guía de PUMMA), los pasos a seguir son los siguientes:

a) Configurar la CMT SIMMP-2 para operación en modo boot-strap (puentes J4 y J5 colocados).
b) Con la tarjeta desenergizada colocar en su base la memoria a programar.
c) Energizar con la fuente de cinco volts
d) Oprimir botón de RESET en SIMMP-2.

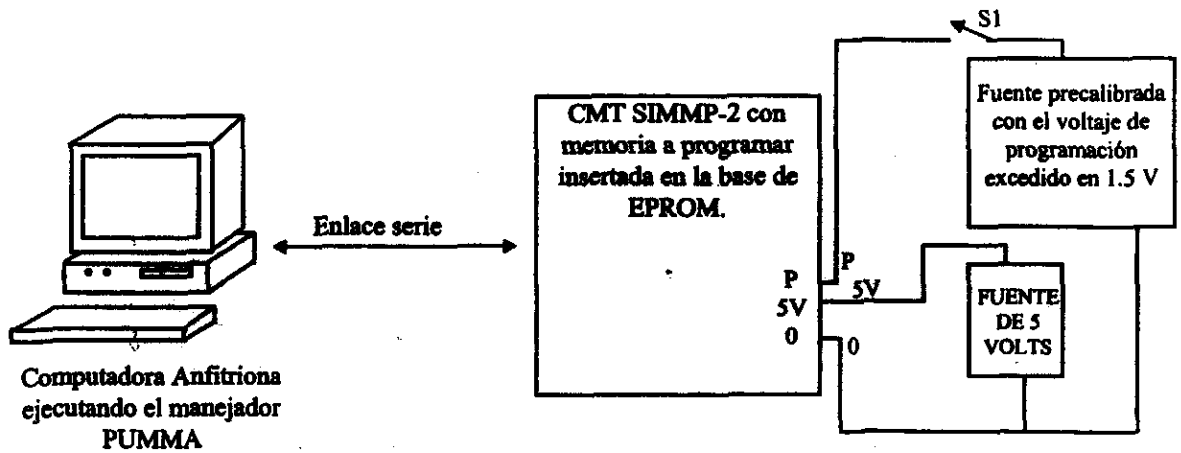


Figura 2.16.- Arreglo para programar memorias EPROM empleando a la CMT SIMMP-2.

e) Ejecutar en la computadora anfitriona el manejador PUMMA.
f) Una vez que PUMMA esté en su menú principal y el ambiente PUMMA esté ejecutándose en la tarjeta SIMMP-2 pasar a el menú de manejo de memoria.
g) Una vez en el menú de manejo de memoria pasar al menú de programación de la EPROM externa. Al entrar a esta opción PUMMA presentará un menú de cuatro opciones siendo estas las siguientes:
   1) Verificar que la EPROM esté completamente borrada.
   2) Pasar a programar la EPROM.
   3) Pasar a verificar lo programado en la EPROM.
   4) Pasar a leer la EPROM.
h) Verificar, antes de proceder a la programación, que la memoria esté completamente borrada de no ser así proceder a borrarla y repetir los pasos anteriores.
i) Invocar la opción (2) del menú de programación.
j) Indicar a PUMMA el origen de la información a programar (bytes introducidos desde el teclado de la PC o un archivo de disco), en caso de que la información a programar sean bytes escritos desde el teclado PUMMA procederá a pedirlos sucesivamente, en otro caso PUMMA pedirá el nombre del archivo BLM o LEM a programar.
l) Indicar a PUMMA la dirección incial de programación de los datos desde el punto de vista de la EPROM y no de su posible localización en un mapa de memoria de un sistema.
m) Indicar a PUMMA el tipo de memoria a programar tecleando los dígitos finales del número de parte de la EPROM, por ejemplo, si se va a programar una memoria 27128 el usuario deberá teclear 128 seguido de la opresión de la tecla return.
n) Indicar a PUMMA el tipo de voltaje de programación y verificar la correcta colocación de los puentes J24 y J25. En caso de que el voltaje de programación sea 12.5 volts J24 y J25 deberán estar colocados, en otro caso deberá estar puesto solamente el puente J25.
ñ) Cerrar el interruptor S1 de la Figura 2.15 y después oprimir cualquier tecla, en seguida a lo anterior PUMMA desplegara un letrero que dice: UN MOMENTO POR FAVOR ESTOY PROGRAMANDO. Después de terminar la programación PUMMA indicará el número de errores al programar debiendo éste ser cero en caso de que la programación haya sido totalmente exitosa, si la memoria que se programó es la 2732, 27C32, 27512 o 27C512 la verificación de la programación debe hacerse a posteriori, empleando para ello la opción tres del menú de programación de la EPROM.
o) Desconectar, abriendo el interruptor S1, el voltaje de programación cuando PUMMA lo indique.

Si el usuario deseara verificar el contenido de una EPROM contra información proporcionada desde teclado o bien contenida en un archivo BLM o LEM, al presentar PUMMA las opciones del menú de programación deberá optar por la opción (3), para seguir los pasos que PUMMA le indique, para más detalles sobre esto se puede consultar el capítulo que trata sobre el manejador PUMMA o la ayuda en línea que contiene el propio programa.

RESPUESTAS AL RESTABLECIMIENTO (RESET) DE LA CMT SIMMP-2

Las distintas formas de respuesta al restablecimiento (RESET) de la CMT SIMMP-2 dependen de características propias del sistema de RESET del 68HC11, del modo de operación del microcontrolador y de colocaciones de puentes en la tarjeta.

Existen para el 68HC11 cuatro posibles maneras en que un restablecimiento (RESET) puede ser invocado y estas son:

a) RESET invocado al detectarse el flanco de subida de la fuente de cinco volts que polariza la tarjeta (POR). Las direcciones de memoria que deben contener al vector correspondiente son la $FFFE y la $FFFF para los modos normales (single-chip y boot-strap), para los modos especiales (boot-strap y TEST) tales direcciones son la $BFFE y la $BFFF.

Si la fuente de alimentación que polariza la tarjeta tuviera un tiempo de levantamiento lento, pudiera llegar a suceder que al energizar la tarjeta no se llevara a cabo correctamente el restablecimiento, originando esto un comportamiento errático del microcontrolador haciéndose necesario el oprimir el botón de RESET para lograr un restablecimiento correcto; para evitar este tipo de problema sobre todo cuando la tarjeta SIMMP-2 vaya a ser parte de un equipo de instrumentación, se podría conectar a el poste de la tarjeta ligado con la terminal de RESET del microcontrolador el circuito de autorrestablecimiento mostrado en la Figura 2.17, para un mejor desempeño del mismo se sugiere desconectar el capacitor C7 de la tarjeta, el circuito mostrado sería exógeno a la tarjeta y parte del hardware del sistema que se estuviera basando en la tarjeta.

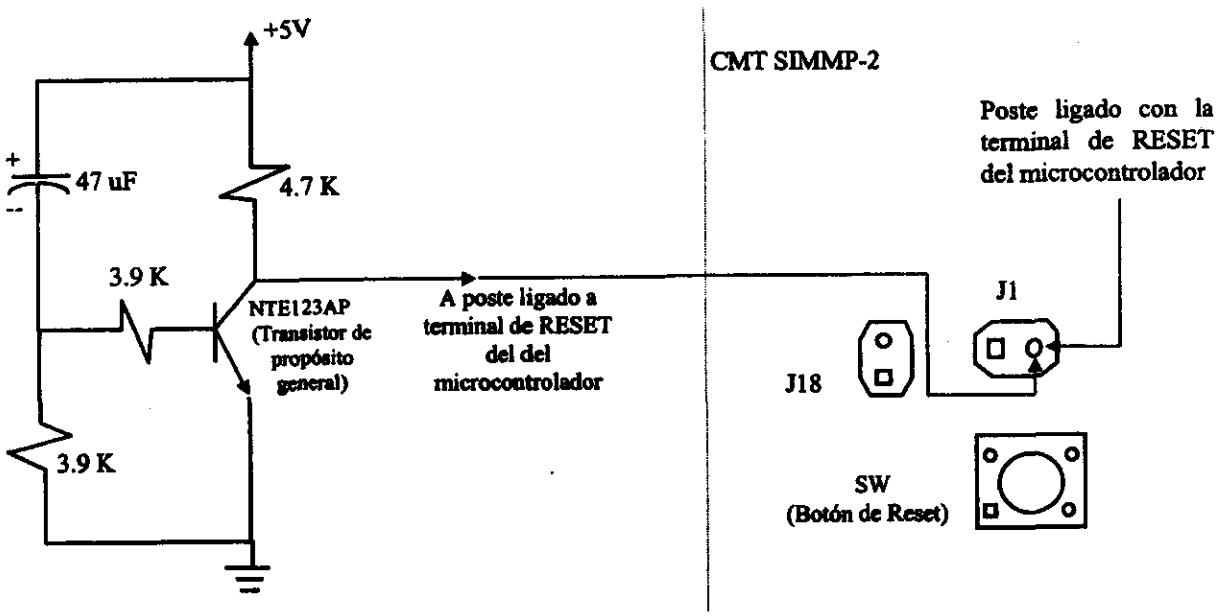


Figura 2.17.- Circuito de autorrestablecimiento que asegura un restablecimiento correcto de la CMT SIMMP-2 aun cuando el tiempo de levantamiento de la fuente de polarización fuera lento.

b) RESET invocado al detectarse una transición de bajo a alto en el pin marcado como RESET en el microcontrolador. Las direcciones de memoria que deben contener al vector

correspondiente son la $FFFE y la $FFFF para los modos normales (single-chip y boot-strap), para los modos especiales (boot-strap y TEST) tales direcciones son la $BFFE y la $BFFF.

c) RESET invocado al detectarse una falla en el funcionamiento de los circuitos de reloj. Las direcciones de memoria que deben contener al vector correspondiente son la $FFFC y la $FFFD para los modos normales (single-chip y boot-strap), para los modos especiales (boot-strap y TEST) tales direcciones son la $BFFC y la $BFFD.

d) RESET invocado al detectar el supervisor de operación correcta (watch-dog) una incorrecta secuencia de ejecución de instrucciones. Las direcciones de memoria que deben contener al vector correspondiente son la $FFFA y la $FFFB para los modos normales (single-chip y boot-strap), para los modos especiales (boot-strap y TEST) tales direcciones son la $BFFA y la $BFFB.

Para una información detallada acerca del funcionamiento de los distintos tipos de RESET mencionados en párrafos anteriores, se sugiere consultar las referencias dos y tres del capítulo que describe el funcionamiento del manejador PUMMA.

Es conveniente señalar aquí, que en caso de que el microcontrolador opere en un modo que no sea boot-strap, para el firmware residente en la CMT SIMMP-2 los tres vectores mencionados anteriormente apuntan a el origen de la EEPROM si tal firmware está definido por archivos BLM cuyo nombre sea terminado por el dígito uno, en caso de que el nombre del archivo definitorio termine en el dígito dos, el vector correspondiente a los RESET de tipo (a) y (b) apunta a el origen de la EEPROM interna y los otros dos (c) y (d) apuntan a direcciones de la página cero similares a las asociadas con tales vectores cuando el microcontrolador opere en modo boot-strap; de ser necesario el usuario podrá modificar esos vectores empleando el manejador PUMMA.

A continuación se describen las distintas respuestas al restablecimiento que puede presentar la tarjeta SIMMP-2, cuando el microcontrolador de la misma opera en cada uno de los cuatro modos posibles.

Respuesta al restablecimiento operando en modo boot-strap. En este caso al darse el RESET el microcontrolador salta a ejecutar un programa denominado por MOTOROLA como boot-loader, tal programa lleva a cabo las siguientes tareas:

1) Carga el apuntador de pila (stack pointer) con una dirección que apunta al tope superior de la RAM interna.
2) Inicializa el puerto serie interno (SCI) a un baudaje de 7812 bps no paridad y un bit de stop.
3) Genera una transición de uno a cero en la terminal transmisora del puerto serie (TXD).
4) Pasa a esperar la recepción de un byte enviado por una computadora anfitriona ligada vía serie al sistema 68HC11, en caso de que tal byte sea $00 se producirá un salto al origen de la memoria EEPROM interna.
5) Si ese primer byte recibido es diferente de $00 y de $FF el baudaje es cambiado a 1200 bps, si el mismo es igual a $FF no se efectúa el cambio mencionado.
6) Los siguientes bytes recibidos son cargados sucesivamente en la RAM interna a partir de la dirección $0000, al detectarse el fin de la transmisión por parte de la computadora anfitriona que esté enviando los bytes al sistema 68HC11, se producirá un salto al origen de la RAM interna autoejecutándose el programa recién cargado en la RAM interna, desde luego que la cadena de bytes recibida debe representar código coherente ya que de no ser así el comportamiento del sistema será impredecible, debiendo ser necesario restablecerlo.

Existen para las distintas versiones del 68HC11 algunas pequeñas variantes en la lógica de ejecución del programa boot-loader, la descrita anteriormente se apega a el caso del 68HC11F1; para más información acerca del programa boot-loader pueden consultarse las referencias (1) y (3) del capítulo sobre el manejador PUMMA.

El programa boot-loader está contenido en un ROM de 256 bytes (mapeado de la dirección $BF00 a la $BFFF) que contiene además los correspondientes vectores de RESET e interrupción propios del modo boot-strap, tal ROM es programado de fabrica y es visible en el mapa de memoria únicamente al operar en el modo boot-strap.

De lo explicado en los párrafos anteriores se deduce que al restablecer la CMT SIMMP-2 cuando el microcontrolador de la misma opere en modo boot-strap, se tendrán las siguientes dos posibilidades:

a) Se pasará a recibir vía serie un programa que será cargado en la RAM interna a partir de su origen, para ser autoejecutado de inmediato una vez que se detecte el fin de la transmisión.
b) Si se conecta un puente entre la terminal de transmisión y recepción de la CMT SIMMP-2, obviamente sin una computadora anfitriona conectada, se producirá un salto al origen de la EEPROM interna, lográndose con esto la autoejecución autónoma de un programa previamente cargado en tal memoria; el puente mencionado podría ser colocado entre los postes dos y tres de la terminal serie de la CMT SIMMP-2, véase la Figura 2.2.

Si la CMT SIMMP-2 opera en el modo boot-strap y el ambiente PUMMA está ejecutándose en la misma, al restablecerla se perderá el mismo por lo que deberá ser necesario reinstalarlo empleando para ello la opción ocho del menú principal del manejador PUMMA.

Respuesta al restablecimiento operando en modo single-chip.- Cuando el microcontrolador de la tarjeta SIMMP-2 opera en modo single-chip, existen dos posibles respuestas al restablecimiento, determinadas por el puente J11; el firmware correspondiente que deberá estar residente en la CMT SIMMP-2 puede ser el contenido ya sea en el archivo FWSP2SA1.BLM o en el FWSP2SA2.BLM; los dos accionamientos posibles son los siguientes:

a) Generación de un salto a ejecución de código compatible con el manejador PUMMA que se estuviera ejecutando en una computadora anfitriona ligada vía serie con la tarjeta, en caso que se detectara que el ambiente PUMMA está residente en la página cero de RAM se producirá un salto inmediato al mismo, en otro caso se pasara a código que puede recibirlo y autoejecutarlo (para más información sobre esto puede consultarse el capítulo referente a el manejador PUMMA). Para que se produzca este accionamiento se requerirá que el puente J11 no esté colocado.
b) Generación de un salto a la dirección $FEA0, donde el usuario previamente debería haber cargado un programa mediante la opción de programación de la EEPROM interna que es parte del manejador PUMMA; la dirección tope de ese programa no deberá exceder a $FFBF. Para este accionamiento se requiere que el puente J11 esté colocado.

Respuesta al restablecimiento operando en modo expandido.-Cuando el microcontrolador de la tarjeta SIMMP-2 opera en modo expandido, existen dos posibles respuestas al restablecimiento, determinadas por el puente J11; el firmware correspondiente que deberá estar residente en la CMT SIMMP-2 debe ser el adecuado a el mapa de memoria que se esté usando en un momento dado (véase la información referente a los mapas EA, EB y EC), los dos accionamientos posibles son los siguientes:

a) Generación de un salto a ejecución de código compatible con el manejador PUMMA que se estuviera ejecutando en una computadora anfitriona ligada vía serie con la tarjeta, en caso que

se detectara que el ambiente PUMMA está residente en la página cero de RAM se producirá un salto inmediato al mismo, en otro caso se pasará a código que puede recibirlo y autoejecutarlo (para más información sobre esto puede consultarse el capítulo referente a el manejador PUMMA). Para que se produzca este accionamiento se requerirá que el puente J11 no esté colocado.

b) Generación de un salto a la dirección origen de la EPROM, que dependerá del mapa escogido, donde el usuario previamente debería haber cargado un programa mediante la opción de programación de la EPROM externa, que es parte del manejador PUMMA. Para este accionamiento se requiere que el puente J11 esté colocado.

Respuesta al restablecimiento operando en modo TEST.-Cuando el microcontrolador de la tarjeta SIMMP-2 opera en modo TEST, existen dos posibles respuestas al restablecimiento, determinadas por el puente J11; el firmware correspondiente que deberá estar residente en la CMT SIMMP-2 puede ser el contenido ya sea en el archivo FWSP2TA1.BLM o en el FWSP2TA2.BLM; los dos accionamientos posibles son los siguientes:

a) Generación de un salto a ejecución de código compatible con el manejador PUMMA que se estuviera ejecutando en una computadora anfitriona ligada vía serie con la tarjeta, en caso que se detectara que el ambiente PUMMA está residente en la página cero de RAM se producirá un salto inmediato al mismo, en otro caso se pasará a código que puede recibirlo y autoejecutarlo (para más información sobre esto puede consultarse el capítulo referente a el manejador PUMMA). Para que se produzca este accionamiento se requerirá que el puente J11 no esté colocado.

b) Generación de un salto a la dirección origen de la EPROM externa, donde el usuario previamente debería haber cargado un programa mediante la opción de programación de la EPROM que es parte del manejador PUMMA; para este accionamiento se requiere que el puente J11 esté colocado.

En la Tabla 2.4 se resumen los accionamientos al restablecer la CMT SIMMP-2.

Ejemplo de uso del manejador PUMMA para cargar y ejecutar un programa en lenguaje de máquina en la CMT SIMMP-2.

Para el microcontrolador un programa a ejecutarse en el mismo no es sino una lista de palabras binarias de ocho bits, donde cada instrucción elemental estará codificada en uno o varios bytes, para que el programa se ejecute se requiere que la lista de bytes que lo conforma sea cargada en memoria a partir de una determinada dirección, para luego transferir el valor del contador de programa (PC) a la dirección donde haya quedado cargado el código de la primera instrucción; lo anterior puede hacerse empleando el manejador PUMMA, a continuación se ilustra esto por medio de un programa ejemplo sencillo, para una mejor comprensión de este ejemplo se recomienda consultar paralelamente lo referente a las opciones uno, seis y siete del menú principal del manejador PUMMA en el capítulo referente a esto o en la ayuda en línea del manejador.

Modo de Operación	Puente colocado entre las terminales Tx y Rx de SIMMP-2 (ver Figura 2.2)	Puente no colocado entre las terminales Tx y Rx de la SIMMP-2 (ver Figura 2.2)	Puente J11 colocado	Puente J11 no colocado
Boot-Strap	Se genera un salto al origen de la EEPROM interna	Se pasa a firmware de recepción de un programa a colocarse en la página cero.	------	------
Single Chip	Posibilidad no permitida.	----------	Se genera un salto a dirección $FEA0 dentro de la EEPROM interna.	Se pasa a ejecutar firmware compatible con el manejador PUMMA.
Test	Posibilidad no permitida.	----------	Se genera salto al origen de la EPROM externa.	Se pasa a ejecutar firmware compatible con el manejador PUMMA.
Expandido	Posibilidad no permitida.	----------	Se genera salto al origen de la EPROM externa.	Se pasa a ejecutar firmware compatible con el manejador PUMMA.

Tabla 2.4 .- Accionamientos al restablecer la CMT SIMMP-2.

El programa ejemplo, que se denominara como PROGE1 y supuestamente ha de ser cargado a partir de la dirección $0100, es el siguiente:

Programa PROGE1

```
DIREC.  CÓDIGO  ETIQ     INSTRUC  OPER.    COMENTARIOS
0100    8608            LDAA     #$08     ;Carga de A con$08
0102    8D06    VUELTA: BSR      RET      ;Invoca retardo de .25 s
0104    4A              DECA              ;Decrementa A
0105    26FB            BNE      VUELTA   ;Saltar si A<>0
0107    7E0000          JMP      $0000    ;Salta a ambiente PUMMA
010A    3C      RET:    PSHX              ;Registro X a pila
010B    CEFFFF          LDX      #$FFFF   ;Carga X con $FFFF
010E    01      REP:    NOP
010F    09              DEX               ;Decrementa X
0110    26FC            BNE      REP      ;Salta a REP si X<>0
0112    38              PULX              ;Reinstala X de pila
0113    39              RTS               ;Retorna de subrutina
```

El programa anterior es una espera de un poco más de dos segundos después de la cual se retorna de inmediato al medio ambiente PUMMA, al ejecutarse este código en la CMT SIMMP-

2 se deberá observar que el LED testigo del ambiente PUMMA, deja de centellear por un tiempo un poco mayor a dos segundos; el programa para el microcontrolador es una lista de bytes que para este ejemplo sería la siguiente:

DIR	CÓDIGO HEX
0100	86
0101	08
0102	8D
0103	06
0104	4A
0105	26
0106	FB
0107	7E
0108	00
0109	00
010A	3C
010B	CE
010C	FF
010D	FF
010E	01
011F	09
0110	26
0111	FC
0112	38
0113	39

La lista anterior es el programa del ejemplo en lenguaje de máquina (código), para cargarlo en la CMT SIMMP-2 el mismo debe ser primero validado en la computadora anfitriona empleando la opción uno del menú principal de PUMMA, después de esto para ejecutarlo en la CMT SIMMP-2 se emplearía la opción seis del mismo menú. Si se desea guardar en disco este programa se puede invocar la opción seis del menú de manejo de disco de PUMMA.

Ejemplo de uso del manejador PUMMA, para cargar y ejecutar en la CMT SIMMP-2, un programa cuyo código haya sido generado por un ensamblador o compilador.

Generalmente en las aplicaciones prácticas, el código requerido es generado a partir de un programa fuente que podría ser una secuencia de instrucciones indicadas mediante cadenas de caracteres, que recuerden la función efectuada por la instrucción, seguidos estos por caracteres que indiquen, si es el caso, la manera como se va a accesar uno de los operandos que requiriera la operación a efectuar, antes de los caracteres que denotan la instrucción podría haber caracteres que etiquetaran la posición de la instrucción para poder hacer transferencias a esa posición desde otras partes del programa si esto fuera necesario; cuando la secuencia de instrucciones que contiene un programa es expresado de la manera descrita anteriormente, el mismo está escrito en lenguaje ensamblador; de acuerdo a lo expresado, cada instrucción del programa es declarada usando un sólo renglón, distinguiéndose en el mismo cuatro campos a saber:

a) Campo de etiqueta. En este campo se coloca una cadena de caracteres que normalmente se usa para marcar la posición que tiene una determinada instrucción dentro del programa, normalmente el primer caracter de la etiqueta deberá estar colocado en el primera columna de la página de texto donde se estén escribiendo las instrucciones.
b) Campo de instrucción. En este campo se coloca una cadena de caracteres que recuerdan lo que efectúa la instrucción.
c) Campo de operandos. Aquí se colocan, en caso de ser esto necesario, una cadena de caracteres que denotan en alguna forma la manera en que uno de los operandos que emplee la instrucción es accesado.
d) Campo de comentarios. Este campo se inicia siempre con el caracter (;), todo lo que se coloque a la derecha de este caracter no es tomado en cuenta por el programa que traduce el programa fuente a código de máquina del microcontrolador; normalmente este campo es empleado para poner comentarios que pudieran servir de orientación para el programador o a otra persona en la depuración o mejora del propio programa.

Además de lo anterior, el programa fuente en ensamblador contendrá lo que se denomina como directivas que no son propiamente instrucciones del propio microcontrolador o microprocesador en cuestión, sino, indicaciones entre otras como la asignación de valores numéricos además de caracteres o indicaciones de a partir de que dirección de memoria ha de generarse el código de todo el programa o de un tramo del mismo, para mayor información sobre ensambladores para el 68HC11 se puede consultar la referencia uno de este capítulo.

De acuerdo con lo comentado anteriormente la traducción a código de máquina de un programa fuente la hace un programa que se denomina programa ensamblador o simplemente ensamblador, si el mismo es ejecutado en una computadora cuya unidad central de proceso no sea la misma que la correspondiente al procesador del microcontrolador o microprocesador cuyo código de máquina ha de generarse, el mismo es denominado como ensamblador cruzado (cross assembler), tal es el caso de los ensambladores para el 68HC11 que corren en una computadora de tipo PC; existen en el mercado diversos ensambladores para este microcontrolador, habiendo variantes de uno a otro, sin embargo, todos ellos presentan como mínimo las siguientes facilidades:

a) Generación de un archivo con extensión LST, en donde para cada renglón del mismo se coloca del lado izquierdo la dirección a partir de la cual ha de cargarse el código correspondiente a la instrucción seguido por el propio código en hexadecimal, colocándose después de esto la propia instrucción con sus cuatro campos. Por lo general el archivo contendrá también información referente a las direcciones de memoria correspondientes a las distintas etiquetas que se usaran en el programa.
b) Generación de un archivo S19, este tipo de archivo es un estandard en la presentación del código de máquina a ejecutarse en el 68HC11, el mismo consiste de una secuencia de renglones cuyo primer caracter es siempre la letra S seguida por un caracter que podría ser el guarismo uno o nueve, en este último caso esto testifica al último renglón del archivo, después de estos dos caracteres aparece en el renglón una cadena de bytes expresados en notación hexadecimal empleándose dos caracteres ASCII por cada byte, la información contenida en la cadena mencionada es la siguiente:

1) Los primeros dos caracteres denotan un byte que representa el numero (N) de bytes de información que hay en el renglón sin contar a este indicador.

2) Los siguientes cuatro caracteres, denotan dos bytes que representan la dirección a partir de la cual, han de cargarse los bytes que sigan a estos dos mencionados aquí.
3) Los siguientes 2(N-3) caracteres, representan a los N-3 bytes que han de cargarse a partir de la dirección especificada en el párrafo anterior.
4) El último par de caracteres representan a un byte verificador, que es el complemento a unos del byte menos significativo de la suma de todos los bytes del renglón.

A continuación se indican los pasos a seguir para ejecutar en la CMT SIMMP-2 un programa escrito originalmente en lenguaje ensamblador, para esta explicación se usará como ejemplo el mismo programa que se empleo anteriormente para explicar la carga y ejecución de un programa en lenguaje de máquina.

Paso 1) Se escribe en un editor de texto el programa fuente, frecuentemente se usa la extensión ASM para el archivo ASCII correpondiente, aunque se podría usar otra diferente.

Para el programa PROG1 se tendría que generar un archivo PROG1.ASM cuyo texto podría ser el siguiente:

```
         ORG   $0100     ;Directiva que denota dirección inicial del programa.
         LDAA  #$08      ;Carga de A con$08
VUELTA:  BSR   RET       ;Invoca retardo de .25 s
         DECA            ;Decrementa A
         BNE   VUELTA    ;Saltar si A<>0
         JMP   $0000     ;Salta a ambiente PUMMA
RET:     PSHX            ;Registro X a pila
         LDX   #$FFFF    ;Carga X con $FFFF
REP:     NOP
         DEX             ;Decrementa X
         BNE   REP       ;Salta a REP si X<>0
         PULX            ;Reinstala X de pila
         RTS             ;Retorna de subrutina
```

Paso 2) Se invoca el ensamblador cruzado a emplear, indicándole al mismo si se desea o no generar los archivos LST y S19 correspondientes. Por ejemplo, si se usara el ensamblador cruzado AS11NEW.EXE distribuido por MOTOROLA y que puede ser bajado de la red (freeware), para ensamblar el programa PROG1.ASM con generación de los correspondientes archivos LST y S19, el usuario deberá teclear después del requerimiento de comando del DOS (promt) lo siguiente:

```
AS11NEW PROG1.ASM -L > PROG1.LST
```

El archivo PROG1.LST tendría el siguiente aspecto:

```
0100   8608          LDAA  #$08     ;Carga de A con$08
0102   8D06  VUELTA:  BSR   RET      ;Invoca retardo de .25 s
0104   4A            DECA           ;Decrementa A
0105   26FB          BNE   VUELTA   ;Saltar si A<>0
0107   7E0000        JMP   $0000    ;Salta a ambiente PUMMA
010A   3C      RET:  PSHX           ;Registro X a pila
```

```
010B    CEFFFF          LDX   #$FFFF    ;Carga X con $FFFF
010E    01      REP:    NOP
010F    09              DEX             ;Decrementa X
0110    26FC            BNE   REP       ;Salta a REP si X<>0
0112    38              PULX            ;Reinstala X de pila
0113    39              RTS             ;Retorna de subrutina
```

En caso de que el ensamblador detectara algún error, indicaría esto en el archivo LST para que el usuario al revisarlo vea en que línea o líneas del programa se presentó el problema y de esta manera poder hacer las correcciones pertinentes en el archivo ASM correspondiente, para más detalles acerca del funcionamiento de este ensamblador se recomienda consultar su archivo de documentación que puede ser también bajado de la red; el archivo PROG1.S19 correspondiente a este ejemplo tendría el siguiente aspecto:

```
S113010086088D064A26FB7E00003CCEFFFF0109CF
S107011026FC383954
S9030000FC
```

Para ejecutar PROG1 en la CMT SIMMP-2 se deben seguir los siguientes pasos:

1) Inicializar el sistema A-D para desarrollo, con la computadora PC ejecutando el manejador PUMMA y enlazada vía serie con la CMT SIMMP-2 (para más detalles acerca de esto, puede consultarse el capítulo que habla de la guía de PUMMA).
2) Una vez en el menú principal de PUMMA pasar al menú de manejo de disco.
3) Desde el menú de manejo de disco invocar la opción de lectura de un archivo S19, proporcionando aquí el nombre correspondiente sin extensión, para el caso de este ejemplo el usuario deberá teclear PROG1 seguido de la opresión de la tecla <RETURN>.
4) En seguida a lo anterior, PUMMA pide el nombre de el archivo BLM que va a contener el programa en forma de lista (lenguaje de máquina), si el usuario teclea aquí <RETURN> sin escribir ningún nombre el archivo BLM a generar tendrá el mismo nombre que el correspondiente S19, (PROG1.BLM).
5) Después de un tiempo PUMMA desplegará en pantalla los intervalos de direcciones que conforman los distintos bloques que conforman el programa, indicando al final de esto la dirección correspondiente al récord S9, aquí el usuario deberá oprimir cualquier tecla para que PUMMA retorne al menú de manejo de disco.
6) Retornar al menú principal de PUMMA y una vez aquí invocar la opción seis del mismo para que la lista hexadecimal que corresponde al programa sea bajada, colocada en memoria del microcontrolador y ejecutada; al hacer esto el usuario deberá observar el funcionamiento del programa como se describió anteriormente al explicar la metodología para cargar y ejecutar el mismo programa en lenguaje de máquina.

En caso de que el programa fuente haya sido originalmente escrito en lenguaje C, el usuario deberá emplear un compilador cruzado que le genere el correspondiente archivo S19, para la ejecución del programa en la CMT SIMMP-2 podrían seguirse los seis pasos descritos anteriormente para el caso de que el programa fuente haya sido escrito en lenguaje ensamblador; sin embargo, dado que el código que genera un compilador puede ser de un tamaño considerablemente mayor al que se generaría para el caso de un ensamblador, los pasos recomendables a seguir para bajar y ejecutar el programa serían los siguientes:

1) Inicializar el sistema A-D para desarrollo, con la computadora PC ejecutando el manejador PUMMA y enlazada vía serie con la CMT SIMMP-2 (para más detalles acerca de esto, puede consultarse el capítulo que habla de la guía de PUMMA).
2) Una vez en el menú principal de PUMMA pasar al menú de manejo de disco.
3) Desde el menú de manejo de disco invocar la opción de carga en memoria RAM de un archivo S19, una vez que PUMMA haya hecho esta tarea, pasar al menú principal del manejador para invocar el menú de manejo de memoria.
4) Desde el menú de manejo de memoria invocar la opción de ejecución a partir de una dirección de memoria, aquí el usuario deberá especificar la dirección inicial del código objeto del programa (esta información de alguna manera la debe proporcionar el compilador), una vez que PUMMA baja a la CMT SIMMP-2 la correspondiente orden el programa deberá iniciar su ejecución.

Cabe señalar aquí que dado el tamaño del código que genera un compilador, es recomendable que la CMT SIMMP-2 opere en modo expandido, además algunos compiladores suponen que los vectores de interrupción son los mismos que los correspondientes al modo boot-strap, por lo que el usuario deberá asegurarse de que el firmware correspondiente así lo consigne, (véase la información referente a los distintos mapas de memoria con los que puede operar la CMT SIMMP-2 en este mismo capítulo); desde luego que deberá haber memoria RAM física en las direcciones que así lo requiera el programa.

En caso de que se requiera que el programa opere en la CMT SIMMP-2 de manera autónoma, el mismo con las asignaciones de direcciones pertinentes (opciones del compilador), deberá ser programado en la EPROM de la tarjeta a partir del origen de la misma, para la ejecución autónoma del programa deberá restablecerse la CMT SIMMP-2 con el puente J11 colocado.

En caso de observarse dificultades al pasar a EPROM el programa que ya ha sido ejecutado correctamente en RAM, podría hacerse lo siguiente:

1) Desde la opción dos del menú de manejo de disco de PUMMA leer el correspondiente archivo S19, para que se genere el archivo BLM respectivo.
2) Programar en la EPROM el archivo BLM generado en el punto anterior, 32 bytes adelante del origen de esta memoria.
3) A partir del origen de la EPROM programar un código que copie de EPROM a RAM la totalidad del código del programa, para después generar un salto a la dirección de RAM donde originalmente se iniciaba el programa cuando el mismo se ejecutaba desde RAM, de esta manera al restablecer la CMT SIMMP-2 con el puente J11 colocado, se copiaría a RAM el programa para su ejecución inmediata en la misma.

Desde luego que las consideraciones, descritas en párrafos anteriores, para ejecución autónoma de un programa escrito originalmente en lenguaje C, serían también aplicables en el caso de que el mismo haya sido escrito en lenguaje ensamblador.

REFERENCIAS:

1) Driscoll. Coughlin. Villanucci.
Data Acquisition and Process Control with the M68HC11 Microcontroller.
Merrill.
1994.

2) Peter Spasov
Microcontroller Technology. The 68HC11.
Prentice Hall.
1993.

XR-2206

Monolithic
Function Generator

EXAR ...the analog plus company™

June 1997-3

FEATURES

- Low-Sine Wave Distortion, 0.5%, Typical
- Excellent Temperature Stability, 20ppm/°C, Typ.
- Wide Sweep Range, 2000:1, Typical
- Low-Supply Sensitivity, 0.01%V, Typ.
- Linear Amplitude Modulation
- TTL Compatible FSK Controls
- Wide Supply Range, 10V to 26V
- Adjustable Duty Cycle, 1% TO 99%

APPLICATIONS

- Waveform Generation
- Sweep Generation
- AM/FM Generation
- V/F Conversion
- FSK Generation
- Phase-Locked Loops (VCO)

GENERAL DESCRIPTION

The XR-2206 is a monolithic function generator integrated circuit capable of producing high quality sine, square, triangle, ramp, and pulse waveforms of high-stability and accuracy. The output waveforms can be both amplitude and frequency modulated by an external voltage. Frequency of operation can be selected externally over a range of 0.01Hz to more than 1MHz.

The circuit is ideally suited for communications, instrumentation, and function generator applications requiring sinusoidal tone, AM, FM, or FSK generation. It has a typical drift specification of 20ppm/°C. The oscillator frequency can be linearly swept over a 2000:1 frequency range with an external control voltage, while maintaining low distortion.

ORDERING INFORMATION

Part No.	Package	Operating Temperature Range
XR-2206M	16 Lead 300 Mil CDIP	-55°C to +125°C
XR-2206P	16 Lead 300 Mil PDIP	–40°C to +85°C
XR-2206CP	16 Lead 300 Mil PDIP	0°C to +70°C
XR-2206D	16 Lead 300 Mil JEDEC SOIC	0°C to +70°C

TQM™

EXAR Corporation, 48720 Kato Road, Fremont, CA 94538 ◆ (510) 668-7000 ◆ (510) 668-7017

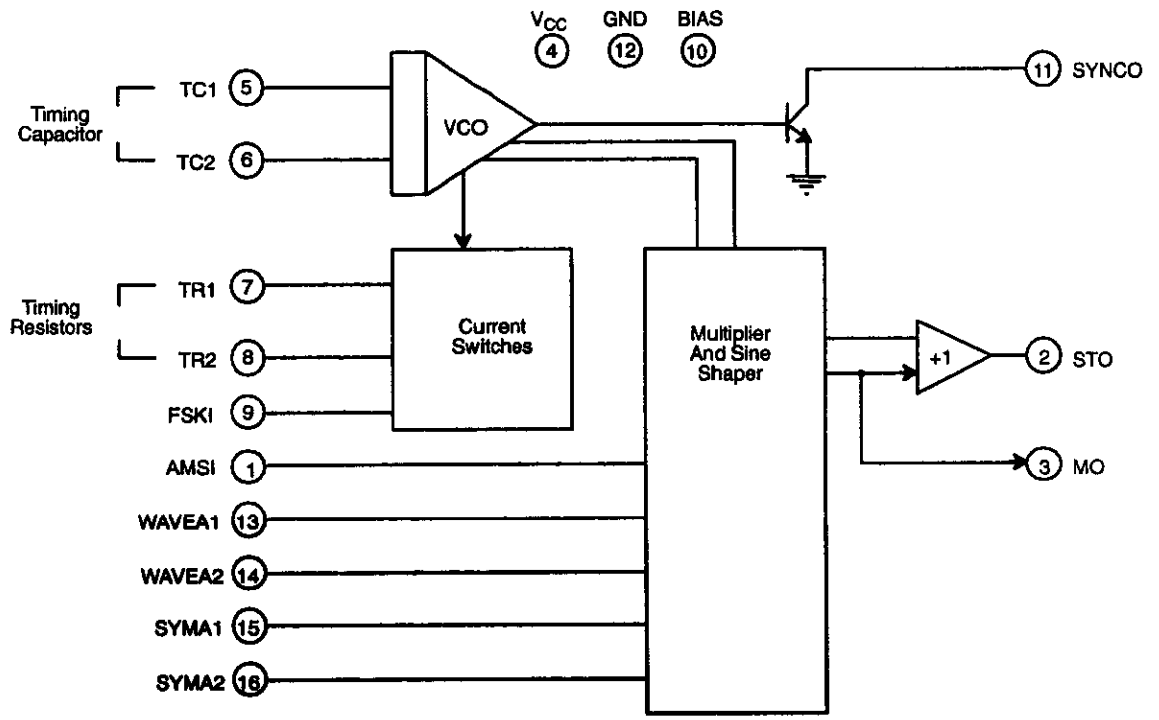
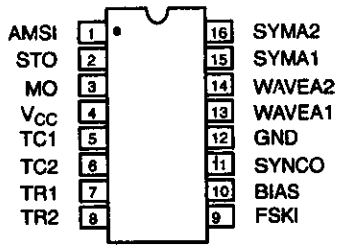
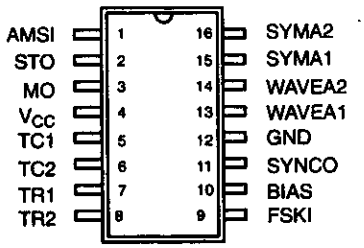


Figure 1. XR-2206 Block Diagram



16 Lead PDIP, CDIP (0.300")



16 Lead SOIC (Jedec, 0.300")

PIN DESCRIPTION

Pin #	Symbol	Type	Description
1	AMSI	I	**Amplitude Modulating Signal Input.**
2	STO	O	**Sine or Triangle Wave Output.**
3	MO	O	**Multiplier Output.**
4	V_{CC}		**Positive Power Supply.**
5	TC1	I	**Timing Capacitor Input.**
6	TC2	I	**Timing Capacitor Input.**
7	TR1	O	**Timing Resistor 1 Output.**
8	TR2	O	**Timing Resistor 2 Output.**
9	FSKI	I	**Frequency Shift Keying Input.**
10	BIAS	O	**Internal Voltage Reference.**
11	SYNCO	O	**Sync Output.** This output is a open collector and needs a pull up resistor to V_{CC}.
12	GND		**Ground pin.**
13	WAVEA1	I	**Wave Form Adjust Input 1.**
14	WAVEA2	I	**Wave Form Adjust Input 2.**
15	SYMA1	I	**Wave Symetry Adjust 1.**
16	SYMA2	I	**Wave Symetry Adjust 2.**

DC ELECTRICAL CHARACTERISTICS

Test Conditions: Test Circuit of *Figure 2* **Vcc = 12V, T_A = 25°C, C = 0.01µF, R_1 = 100kΩ, R_2 = 10kΩ, R_3 = 25kΩ** Unless Otherwise Specified. S_1 open for triangle, closed for sine wave.

Parameters	XR-2206M/P			XR-2206CP/D			Units	Conditions
	Min.	Typ.	Max.	Min.	Typ.	Max.		
General Characteristics								
Single Supply Voltage	10		26	10		26	V	
Split-Supply Voltage	±5		±13	±5		±13	V	
Supply Current		12	17		14	20	mA	$R_1 \geq 10k\Omega$
Oscillator Section								
Max. Operating Frequency	0.5	1		0.5	1		MHz	C = 1000pF, R_1 = 1kΩ
Lowest Practical Frequency		0.01			0.01		Hz	C = 50µF, R_1 = 2MΩ
Frequency Accuracy		±1	±4		±2		% of f_o	$f_o = 1/R_1C$
Temperature Stability Frequency		±10	±50		±20		ppm/°C	0°C ≤ T_A ≤ 70°C $R_1 = R_2$ = 20kΩ
Sine Wave Amplitude Stability[2]		4800			4800		ppm/°C	
Supply Sensitivity		0.01	0.1		0.01		%/V	V_{LOW} = 10V, V_{HIGH} = 20V, $R_1 = R_2$ = 20kΩ
Sweep Range	1000:1	2000:1			2000:1		$f_H = f_L$	f_H @ R_1 = 1kΩ f_L @ R_1 = 2MΩ
Sweep Linearity								
10:1 Sweep		2			2		%	f_L = 1kHz, f_H = 10kHz
1000:1 Sweep		8			8		%	f_L = 100Hz, f_H = 100kHz
FM Distortion		0.1			0.1		%	±10% Deviation
Recommended Timing Components								
Timing Capacitor: C	0.001		100	0.001		100	µF	*Figure 5*
Timing Resistors: R_1 & R_2	1		2000	1		2000	kΩ	
Triangle Sine Wave Output[1]								*Figure 3*
Triangle Amplitude		160			160		mV/kΩ	*Figure 2*, S_1 Open
Sine Wave Amplitude	40	60	80		60		mV/kΩ	*Figure 2*, S_1 Closed
Max. Output Swing		6			6		Vp-p	
Output Impedance		600			600		Ω	
Triangle Linearity		1			1		%	
Amplitude Stability		0.5			0.5		dB	For 1000:1 Sweep
Sine Wave Distortion								
Without Adjustment		2.5			2.5		%	R_1 = 30kΩ
With Adjustment		0.4	1.0		0.5	1.5	%	See *Figure 7* and *Figure 8*

Notes

[1] Output amplitude is directly proportional to the resistance, R_3, on Pin 3. See Figure 3.

[2] For maximum amplitude stability, R_3 should be a positive temperature coefficient resistor.

***Bold face parameters** are covered by production test and guaranteed over operating temperature range.*

DC ELECTRICAL CHARACTERISTICS (CONT'D)

Parameters	XR-2206M/P			XR-2206CP/D			Units	Conditions
	Min.	Typ.	Max.	Min.	Typ.	Max.		
Amplitude Modulation								
Input Impedance	50	100		50	100		kΩ	
Modulation Range		100			100		%	
Carrier Suppression		55			55		dB	
Linearity		2			2		%	For 95% modulation
Square-Wave Output								
Amplitude		12			12		Vp-p	Measured at Pin 11.
Rise Time		250			250		ns	C_L = 10pF
Fall Time		50			50		ns	C_L = 10pF
Saturation Voltage		0.2	**0.4**		0.2	0.6	V	I_L = 2mA
Leakage Current		0.1	**20**		0.1	100	µA	V_{CC} = 26V
FSK Keying Level (Pin 9)	0.8	1.4	**2.4**	0.8	1.4	2.4	V	See section on circuit controls
Reference Bypass Voltage	2.9	3.1	**3.3**	2.5	3	3.5	V	Measured at Pin 10.

Notes
[1] Output amplitude is directly proportional to the resistance, R_9, on Pin 3. See Figure 3.
[2] For maximum amplitude stability, R_9 should be a positive temperature coefficient resistor.
***Bold face parameters** are covered by production test and guaranteed over operating temperature range.*

Specifications are subject to change without notice

ABSOLUTE MAXIMUM RATINGS

Power Supply 26V
Power Dissipation 750mW
Derate Above 25°C ..... 5mW/°C
Total Timing Current 6mA
Storage Temperature ..... -65°C to +150°C

SYSTEM DESCRIPTION

The XR-2206 is comprised of four functional blocks; a voltage-controlled oscillator (VCO), an analog multiplier and sine-shaper; a unity gain buffer amplifier; and a set of current switches.

The VCO produces an output frequency proportional to an input current, which is set by a resistor from the timing terminals to ground. With two timing pins, two discrete output frequencies can be independently produced for FSK generation applications by using the FSK input control pin. This input controls the current switches which select one of the timing resistor currents, and routes it to the VCO.

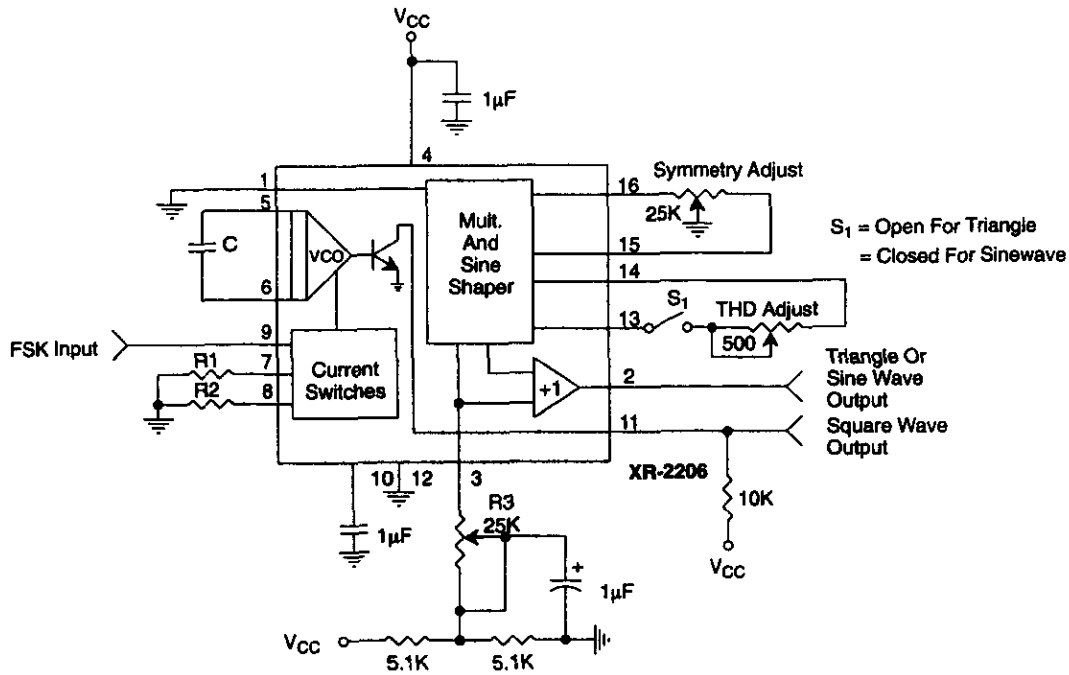


Figure 2. Basic Test Circuit

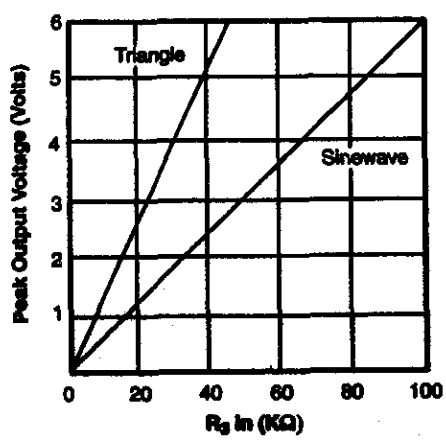


Figure 3. Output Amplitude as a Function of the Resistor, R3, at Pin 3

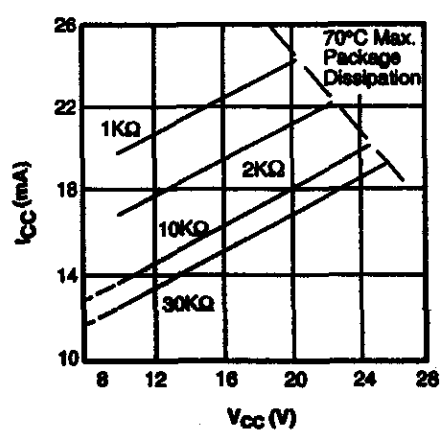


Figure 4. Supply Current vs Supply Voltage, Timing, R

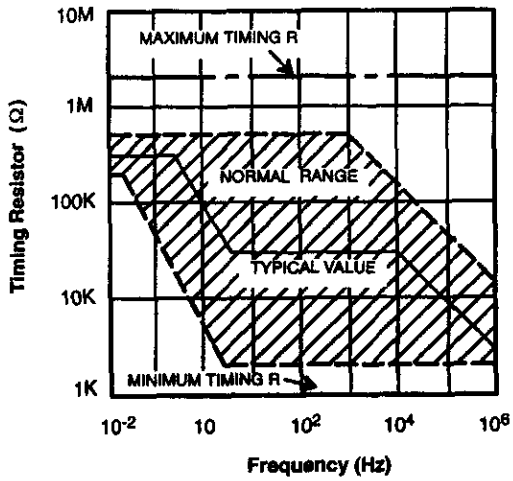


Figure 5. R versus Oscillation Frequency.

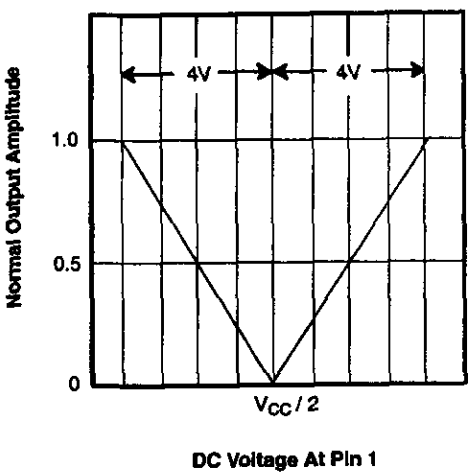


Figure 6. Normalized Output Amplitude versus DC Bias at AM Input (Pin 1)

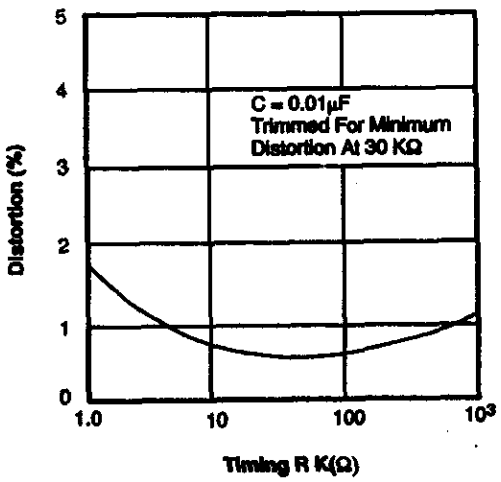


Figure 7. Trimmed Distortion versus Timing Resistor.

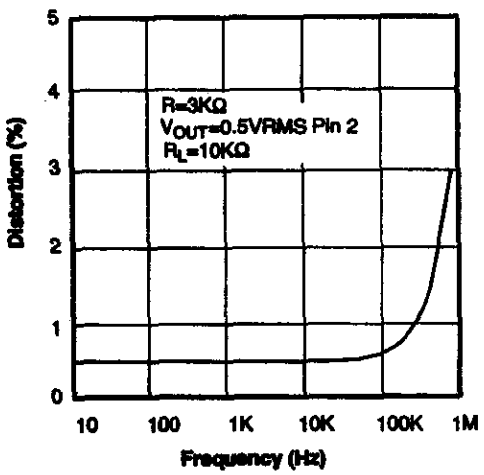


Figure 8. Sine Wave Distortion versus Operating Frequency with Timing Capacitors Varied.

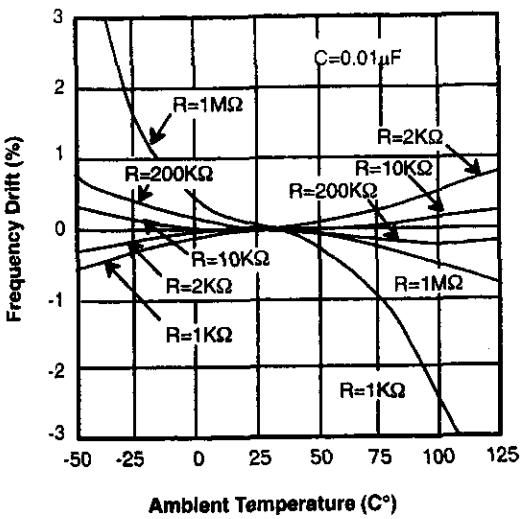


Figure 9. Frequency Drift versus Temperature.

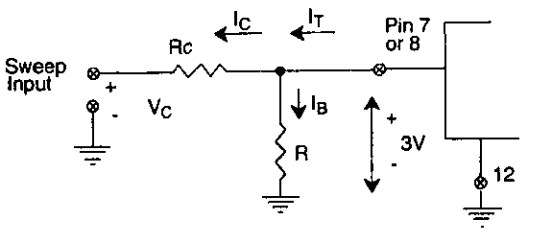


Figure 10. Circuit Connection for Frequency Sweep.

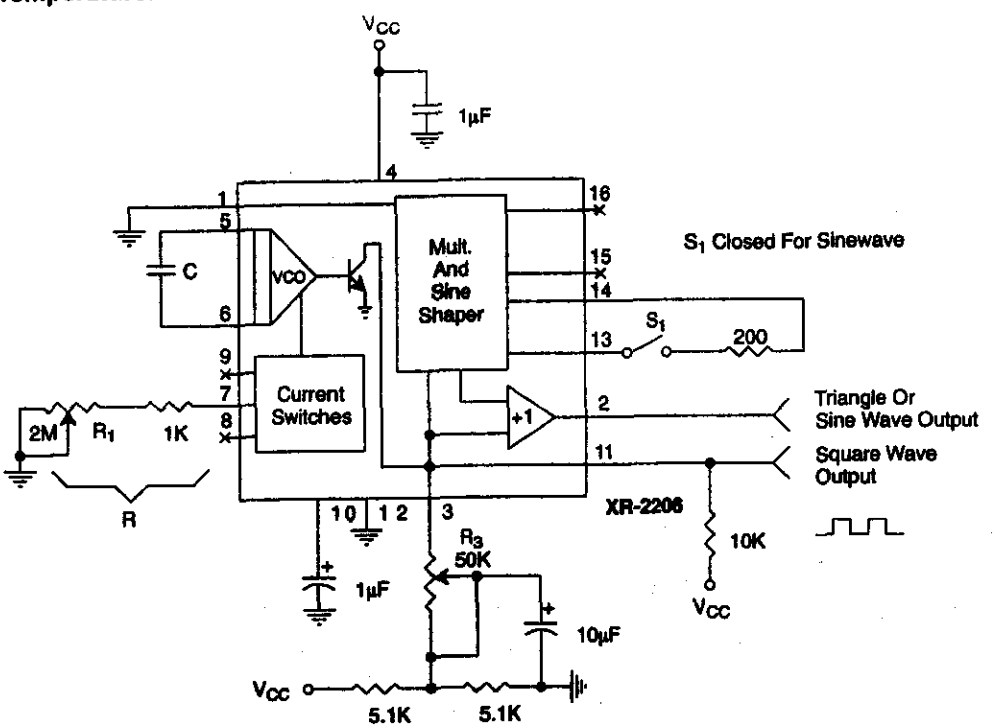


Figure 11. Circuit tor Sine Wave Generation without External Adjustment. (See *Figure 3* for Choice of R_3)

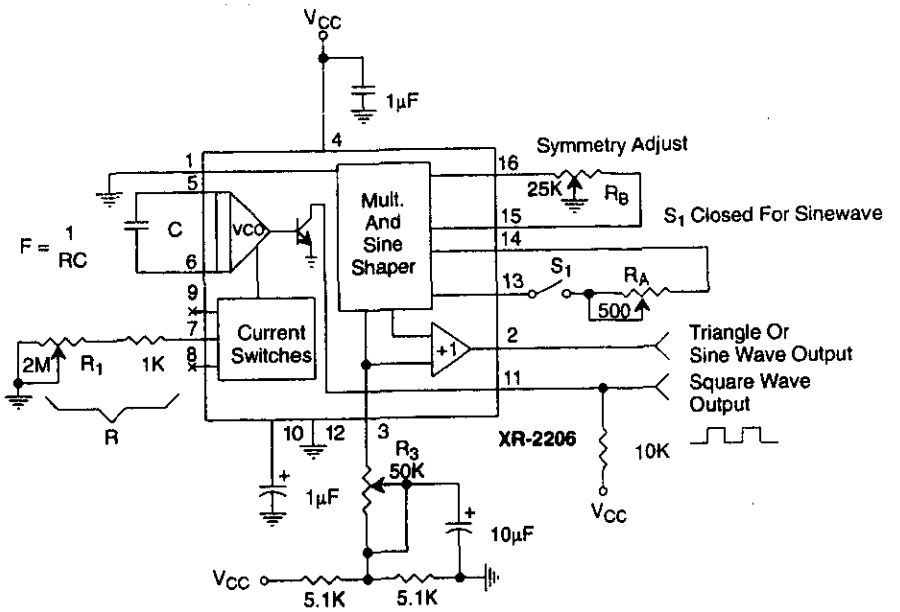


Figure 12. Circuit for Sine Wave Generation with Minimum Harmonic Distortion. (R_3 Determines Output Swing - See *Figure 3*)

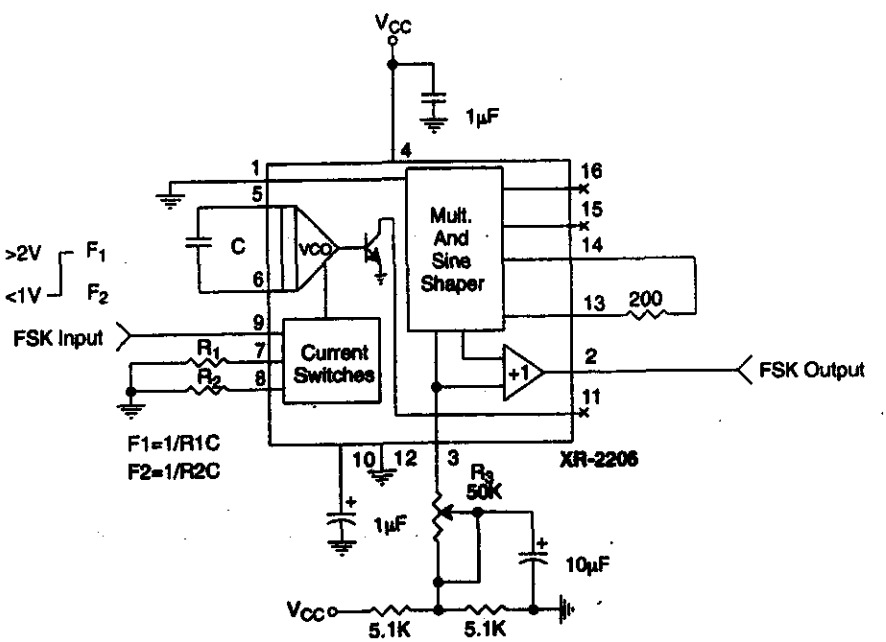


Figure 13. Sinusoidal FSK Generator

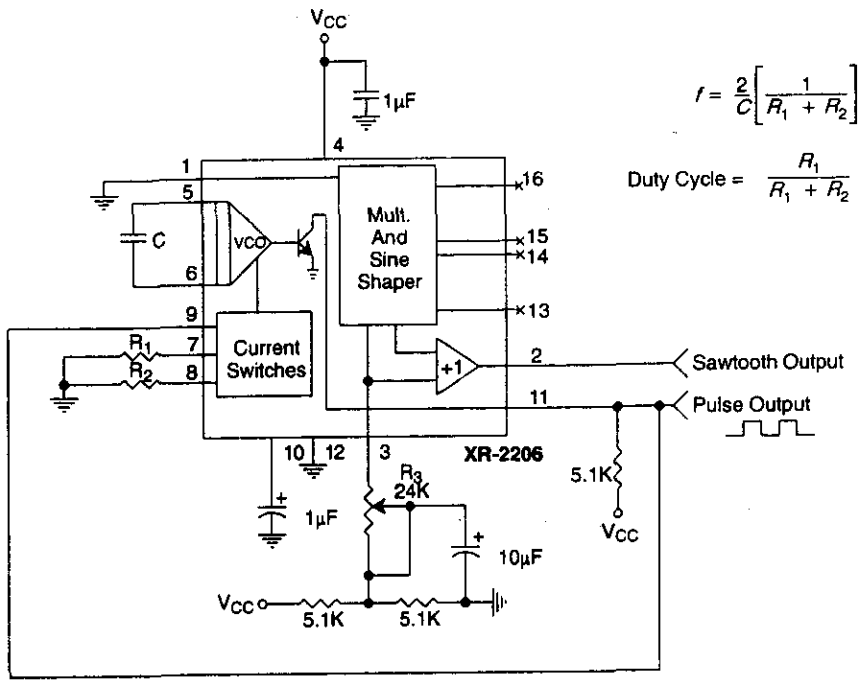


Figure 14. Circuit for Pulse and Ramp Generation.

Frequency-Shift Keying

The XR-2206 can be operated with two separate timing resistors, R_1 and R_2, connected to the timing Pin 7 and 8, respectively, as shown in *Figure 13*. Depending on the polarity of the logic signal at Pin 9, either one or the other of these timing resistors is activated. If Pin 9 is open-circuited or connected to a bias voltage $\geq$ 2V, only R_1 is activated. Similarly, if the voltage level at Pin 9 is $\leq$1V, only R_2 is activated. Thus, the output frequency can be keyed between two levels. f_1 and f_2, as:

$f_1 = 1/R_1C$ and $f_2 = 1/R_2C$

For split-supply operation, the keying voltage at Pin 9 is referenced to V⁻.

Output DC Level Control

The dc level at the output (Pin 2) is approximately the same as the dc bias at Pin 3. In *Figure 11*, *Figure 12* and *Figure 13*, Pin 3 is biased midway between V+ and ground, to give an output dc level of ≈ V⁺/2.

APPLICATIONS INFORMATION

Sine Wave Generation

Without External Adjustment

Figure 11 shows the circuit connection for generating a sinusoidal output from the XR-2206. The potentiometer, R_1 at Pin 7, provides the desired frequency tuning. The maximum output swing is greater than V⁺/2, and the typical distortion (THD) is < 2.5%. If lower sine wave distortion is desired, additional adjustments can be provided as described in the following section.

The circuit of *Figure 11* can be converted to split-supply operation, simply by replacing all ground connections with V⁻. For split-supply operation, R_3 can be directly connected to ground.

With External Adjustment:

The harmonic content of sinusoidal output can be reduced to -0.5% by additional adjustments as shown in *Figure 12.* The potentiometer, R_A, adjusts the sine-shaping resistor, and R_B provides the fine adjustment for the waveform symmetry. The adjustment procedure is as follows:

1. Set R_B at midpoint and adjust R_A for minimum distortion.
2. With R_A set as above, adjust R_B to further reduce distortion.

Triangle Wave Generation

The circuits of *Figure 11* and *Figure 12* can be converted to triangle wave generation, by simply open-circuiting Pin 13 and 14 (i.e., S_1 open). Amplitude of the triangle is approximately twice the sine wave output.

FSK Generation

Figure 13 shows the circuit connection for sinusoidal FSK signal operation. Mark and space frequencies can be independently adjusted by the choice of timing resistors, R_1 and R_2; the output is phase-continuous during transitions. The keying signal is applied to Pin 9. The circuit can be converted to split-supply operation by simply replacing ground with V^-.

Pulse and Ramp Generation

Figure 14 shows the circuit for pulse and ramp waveform generation. In this mode of operation, the FSK keying terminal (Pin 9) is shorted to the square-wave output (Pin 11), and the circuit automatically frequency-shift keys itself between two separate frequencies during the positive-going and negative-going output waveforms. The pulse width and duty cycle can be adjusted from 1% to 99% by the choice of R_1 and R_2. The values of R_1 and R_2 should be in the range of 1kΩ to 2MΩ.

PRINCIPLES OF OPERATION

Description of Controls

Frequency of Operation:

The frequency of oscillation, f_o, is determined by the external timing capacitor, C, across Pin 5 and 6, and by the timing resistor, R, connected to either Pin 7 or 8. The frequency is given as:

$$f_o = \frac{1}{RC}\ Hz$$

and can be adjusted by varying either R or C. The recommended values of R, for a given frequency range, as shown in *Figure 5.* Temperature stability is optimum for 4kΩ < R < 200kΩ. Recommended values of C are from 1000pF to 100μF.

Frequency Sweep and Modulation:

Frequency of oscillation is proportional to the total timing current, I_T, drawn from Pin 7 or 8:

$$f = \frac{320 I_T(mA)}{C(\mu F)}\ Hz$$

Timing terminals (Pin 7 or 8) are low-impedance points, and are internally biased at +3V, with respect to Pin 12. Frequency varies linearly with IT, over a wide range of current values, from 1μA to 3mA. The frequency can be controlled by applying a control voltage, V_C, to the activated timing pin as shown in *Figure 10.* The frequency of oscillation is related to VC as:

$$f = \frac{1}{RC}\left(1 + \frac{R}{R_c}\left(1 - \frac{V_c}{3}\right)\right) Hz$$

where V_C is in volts. The voltage-to-frequency conversion gain, K, is given as:

$$K = \partial f / \partial V_C = -\frac{0.32}{R_c C}\ \ Hz/V$$

CAUTION: *For safety operation of the circuit, I_T should be limited to ≤ 3mA.*

Output Amplitude:

Maximum output amplitude is inversely proportional to the external resistor, R_3, connected to Pin 3 (see *Figure 3*). For sine wave output, amplitude is approximately 60mV peak per kΩ of R_3; for triangle, the peak amplitude is approximately 160mV peak per kΩ of R_3. Thus, for example, R_3 = 50kΩ would produce approximately 13V sinusoidal output amplitude.

Amplitude Modulation:

Output amplitude can be modulated by applying a dc bias and a modulating signal to Pin 1. The internal impedance at Pin 1 is approximately 100kΩ. Output amplitude varies linearly with the applied voltage at Pin 1, for values of dc bias at this pin, within 14 volts of $V_{CC}/2$ as shown in *Figure 6*. As this bias level approaches $V_{CC}/2$, the phase of the output signal is reversed, and the amplitude goes through zero. This property is suitable for phase-shift keying and suppressed-carrier AM generation. Total dynamic range of amplitude modulation is approximately 55dB.

CAUTION: *AM control must be used in conjunction with a well-regulated supply, since the output amplitude now becomes a function of* V_{CC}.

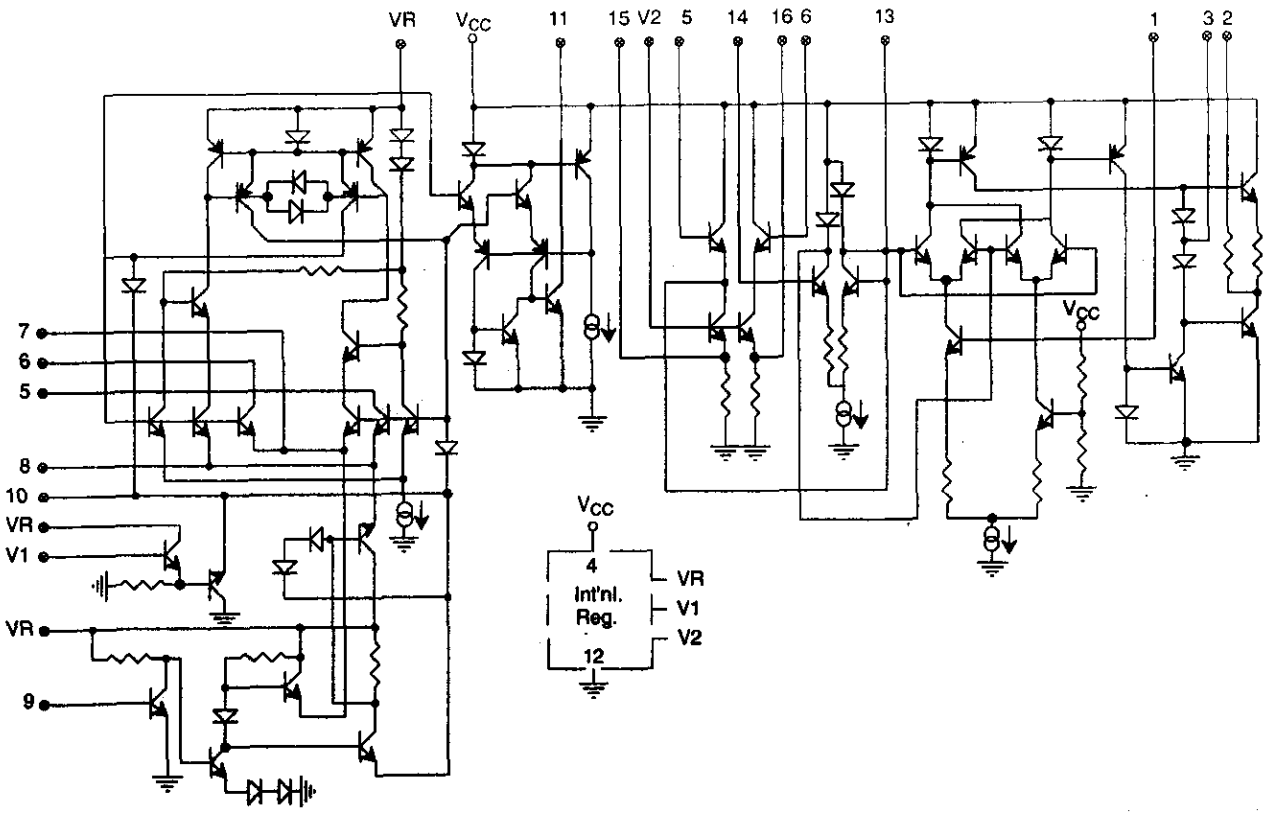
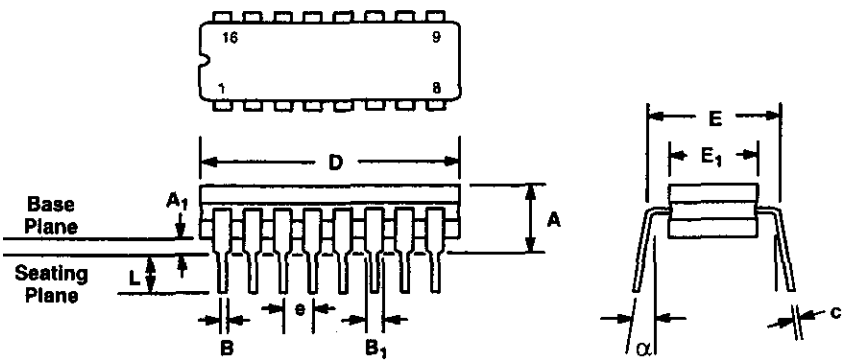


Figure 15. Equivalent Schematic Diagram

16 LEAD CERAMIC DUAL-IN-LINE (300 MIL CDIP)

Rev. 1.00

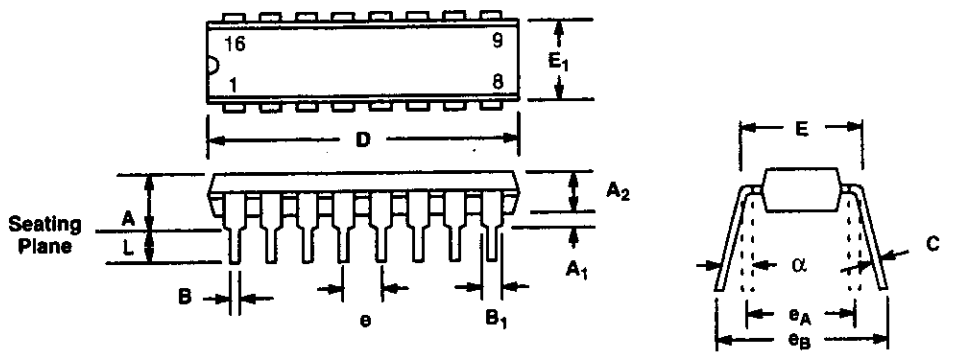


SYMBOL	INCHES		MILLIMETERS	
	MIN	MAX	MIN	MAX
A	0.100	0.200	2.54	5.08
A_1	0.015	0.060	0.38	1.52
B	0.014	0.026	0.36	0.66
B_1	0.045	0.065	1.14	1.65
c	0.008	0.018	0.20	0.46
D	0.740	0.840	18.80	21.34
E_1	0.250	0.310	6.35	7.87
E	0.300 BSC		7.62 BSC	
e	0.100 BSC		2.54 BSC	
L	0.125	0.200	3.18	5.08
α	0°	15°	0°	15°

Note: The control dimension is the inch column

16 LEAD PLASTIC DUAL-IN-LINE (300 MIL PDIP)

Rev. 1.00

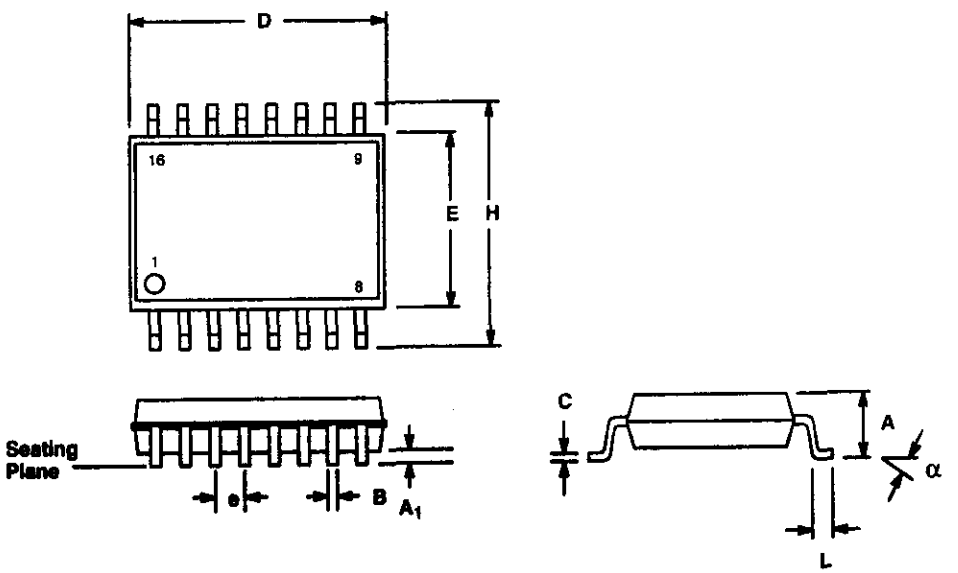


SYMBOL	INCHES		MILLIMETERS	
	MIN	MAX	MIN	MAX
A	0.145	0.210	3.68	5.33
A_1	0.015	0.070	0.38	1.78
A_2	0.115	0.195	2.92	4.95
B	0.014	0.024	0.36	0.56
B_1	0.030	0.070	0.76	1.78
C	0.008	0.014	0.20	0.38
D	0.745	0.840	18.92	21.34
E	0.300	0.325	7.62	8.26
E_1	0.240	0.280	6.10	7.11
e	0.100 BSC		2.54 BSC	
e_A	0.300 BSC		7.62 BSC	
e_B	0.310	0.430	7.87	10.92
L	0.115	0.160	2.92	4.06
α	0°	15°	0°	15°

Note: The control dimension is the inch column

16 LEAD SMALL OUTLINE (300 MIL JEDEC SOIC)

Rev. 1.00



SYMBOL	INCHES		MILLIMETERS	
	MIN	MAX	MIN	MAX
A	0.093	0.104	2.35	2.65
A_1	0.004	0.012	0.10	0.30
B	0.013	0.020	0.33	0.51
C	0.009	0.013	0.23	0.32
D	0.396	0.413	10.10	10.50
E	0.291	0.299	7.40	7.60
e	0.050 BSC		1.27 BSC	
H	0.394	0.419	10.00	10.65
L	0.016	0.050	0.40	1.27
α	0°	8°	0°	8°

Note: The control dimension is the millimeter column

NOTICE

EXAR Corporation reserves the right to make changes to the products contained in this publication in order to improve design, performance or reliability. EXAR Corporation assumes no responsibility for the use of any circuits described herein, conveys no license under any patent or other right, and makes no representation that the circuits are free of patent infringement. Charts and schedules contained here in are only for illustration purposes and may vary depending upon a user's specific application. While the information in this publication has been carefully checked; no responsibility, however, is assumed for inaccuracies.

EXAR Corporation does not recommend the use of any of its products in life support applications where the failure or malfunction of the product can reasonably be expected to cause failure of the life support system or to significantly affect its safety or effectiveness. Products are not authorized for use in such applications unless EXAR Corporation receives, in writing, assurances to its satisfaction that: (a) the risk of injury or damage has been minimized; (b) the user assumes all such risks; (c) potential liability of EXAR Corporation is adequately protected under the circumstances.

Datasheet June 1997

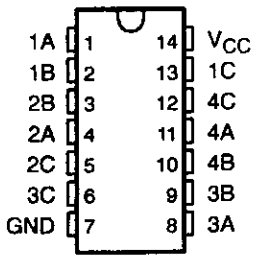
QUADRUPLE BILATERAL ANALOG SWITCH

SN 74 HC 4066

SCLS325B – MARCH 1996 – REVISED MAY 1997

- High Degree of Linearity
- High On-Off Output Voltage Ratio
- Low Crosstalk Between Switches
- Low On-State Impedance — Typically, 50 Ω at V_{CC} = 6 V
- Individual Switch Controls
- Extremely Low Input Current
- Package Options Include Plastic Small-Outline (D), Plastic Shrink Small-Outline (DB), and Thin Shrink Small-Outline (PW) Packages, and Standard Plastic (N) 300-mil DIPs

D, DB, PW, OR N PACKAGE
(TOP VIEW)



description

The SN74HC4066 is a silicon-gate CMOS quadruple analog switch designed to handle both analog and digital signals. Each switch permits signals with amplitudes of up to 6 V (peak) to be transmitted in either direction.

Each switch section has its own enable input control (C). A high-level voltage applied to C turns on the associated switch section.

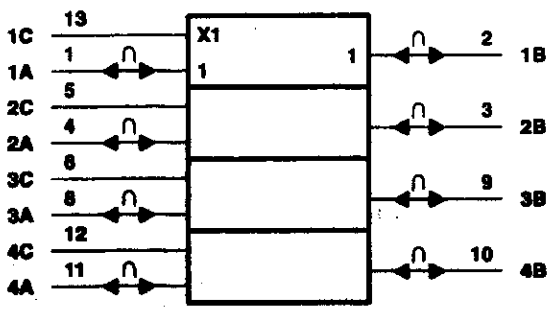
Applications include signal gating, chopping, modulation or demodulation (modem), and signal multiplexing for analog-to-digital and digital-to-analog conversion systems.

The SN74HC4066 is characterized for operation from –40°C to 85°C.

FUNCTION TABLE
(each switch)

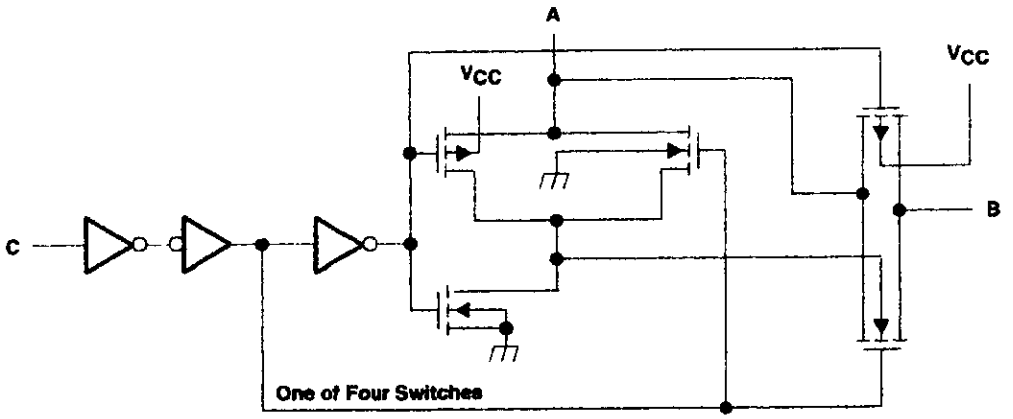
INPUT CONTROL (C)	SWITCH
L	OFF
H	ON

logic symbol†



† This symbol is in accordance with ANSI/IEEE Std 91-1984 and IEC Publication 617-12.

logic diagram, each switch (positive logic)



absolute maximum ratings over operating free-air temperature range (unless otherwise noted)†

Supply voltage range, V_{CC} (see Note 1)	–0.5 V to 7 V
Control-input diode current, I_I ($V_I < 0$ or $V_I > V_{CC}$)	±20 mA
I/O port diode current, I_I ($V_I < 0$ or $V_{I/O} < V_{CC}$)	±20 mA
On-state switch current ($V_{I/O} = 0$ to V_{CC})	±25 mA
Continuous current through V_{CC} or GND	±50 mA
Package thermal impedance, θ_{JA} (see Note 2): D package	127°C/W
DB package	158°C/W
N package	78°C/W
PW package	170°C/W
Storage temperature range, T_{stg}	–65°C to 150°C

† Stresses beyond those listed under "absolute maximum ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under "recommended operating conditions" is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTES: 1. All voltages are with respect to ground unless otherwise specified.
2. The package thermal impedance is calculated in accordance with JESD 51, except for through-hole packages, which use a trace length of zero.

recommended operating conditions

			MIN	NOM	MAX	UNIT
V_{CC}	Supply voltage		2†	5	6	V
$V_{I/O}$	I/O port voltage		0		V_{CC}	V
V_{IH}	High-level input voltage, control inputs	V_{CC} = 2 V	1.5		V_{CC}	V
		V_{CC} = 4.5 V	3.15		V_{CC}	
		V_{CC} = 6 V	4.2		V_{CC}	
V_{IL}	Low-level input voltage, control inputs	V_{CC} = 2 V	0		0.3	V
		V_{CC} = 4.5 V	0		0.9	
		V_{CC} = 6 V	0		1.2	
t_I	Input rise/fall time	V_{CC} = 2 V			1000	ns
		V_{CC} = 4.5 V			500	
		V_{CC} = 6 V			400	
T_A	Operating free-air temperature		–40		85	°C

† With supply voltages at or near 2 V, the analog switch on-state resistance becomes very nonlinear. It is recommended that only digital signals be transmitted at these low supply voltages.

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER			TEST CONDITIONS	V_{CC}	T_A = 25°C			MIN	MAX	UNIT
					MIN	TYP	MAX			
R_{on}	On-state switch resistance		I_T = –1 mA, V_I = 0 to V_{CC}, V_C = V_{IH}, (see Figure 1)	2 V		150				Ω
				4.5 V		50	85		106	
				6 V		30				
$R_{on(p)}$	Peak on resistance		V_I = V_{CC} or GND, V_C = V_{IH}, I_T = –1 mA	2 V		320				Ω
				4.5 V		70	170		215	
				6 V		50				
I_I	Control input current		V_C = 0 or V_{CC}	6 V		±0.1	±100		±1000	nA
I_{soff}	Off-state switch leakage current		V_I = V_{CC} or 0, V_O = V_{CC} or 0, V_C = V_{IL}, (see Figure 2)	6 V			±0.1		±5	µA
I_{son}	On-state switch leakage current		V_I = V_{CC} or 0, V_C = V_{IH}, (see Figure 3)	6 V			±0.1		±5	µA
I_{CC}	Supply current		V_I = 0 or V_{CC}, I_O = 0	6 V			2		20	µA
C_i	Input capacitance	A or B		5 V		9				pF
		C				3	10		10	
C_f	Feedthrough capacitance	A to B	V_I = 0			0.5				pF
C_o	Output capacitance	A or B		5 V		9				pF

switching characteristics over recommended operating free-air temperature range

PARAMETER		FROM (INPUT)	TO (OUTPUT)	TEST CONDITIONS	V_{CC}	T_A = 25°C			MIN	MAX	UNIT
						MIN	TYP	MAX			
t_{PLH}, t_{PHL}	Propagation delay time	A or B	B or A	C_L = 50 pF, (see Figure 4)	2 V		10	60		75	ns
					4.5 V		4	12		15	
					6 V		3	10		13	
t_{PZH}, t_{PZL}	Switch turn-on time	C	A or B	R_L = 1 kΩ, C_L = 50 pF, (see Figure 5)	2 V		70	180		225	ns
					4.5 V		21	36		45	
					6 V		18	31		38	
t_{PLZ}, t_{PHZ}	Switch turn-off time	C	A or B	R_L = 1 kΩ, C_L = 50 pF, (see Figure 5)	2 V		50	200		250	ns
					4.5 V		25	40		50	
					6 V		22	34		43	
f_I	Control input frequency	C	A or B	C_L = 15 pF, R_L = 1 kΩ, V_C = V_{CC} or GND, V_O = V_{CC}/2, (see Figure 6)	2 V		15				MHz
					4.5 V		30				
					6 V		30				
	Control feedthrough noise	C	A or B	C_L = 50 pF, R_{in} = R_L = 600 Ω, V_C = V_{CC} or GND, f_{in} = 1 MHz, (see Figure 7)	4.5 V		15				mV (rms)
					6 V		20				

operating characteristics, V_{CC} = 4.5 V, T_A = 25°C

PARAMETER		TEST CONDITIONS		TYP	UNIT
C_{pd}	Power dissipation capacitance per gate	C_L = 50 pF,	f = 1 MHz	45	pF
	Minimum through bandwidth, A to B or B to A† [20 log (V_O/V_I)] = –3 dB	C_L = 50 pF, V_C = V_{CC},	R_L = 600 Ω, (see Figure 8)	30	MHz
	Crosstalk between any switches‡	C_L = 10 pF, f_{in} = 1 MHz,	R_L = 50 Ω, (see Figure 9)	45	dB
	Feedthrough, switch off, A to B or B to A‡	C_L = 50 pF, f_{in} = 1 MHz,	R_L = 600 Ω, (see Figure 10)	42	dB
	Amplitude distortion rate, A to B or B to A	C_L = 50 pF, f_{in} = 1 kHz,	R_L = 10 kΩ, (see Figure 11)	0.05%	

† Adjust the input amplitude for output = 0 dBm at f = 10 kHz. Input signal must be a sine wave.
‡ Adjust the input amplitude for output = 0 dBm at f = 1 MHz. Input signal must be a sine wave.

PARAMETER MEASUREMENT INFORMATION

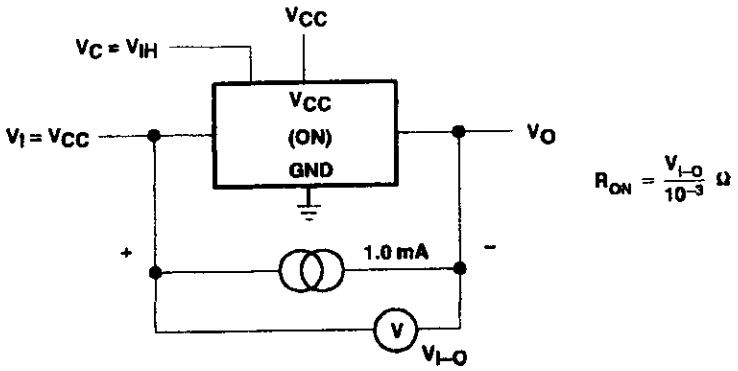
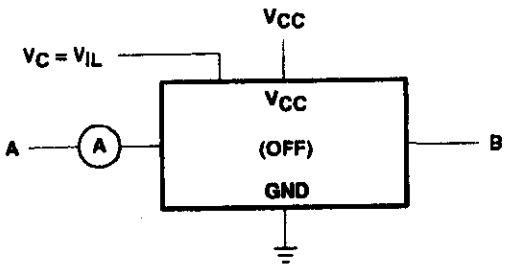


Figure 1. On-State Resistance Test Circuit



$V_S = V_A - V_B$
CONDITION 1: $V_A = 0$, $V_B = V_{CC}$
CONDITION 2: $V_A = V_{CC}$, $V_B = 0$

Figure 2. Off-State Switch Leakage Current Test Circuit

PARAMETER MEASUREMENT INFORMATION

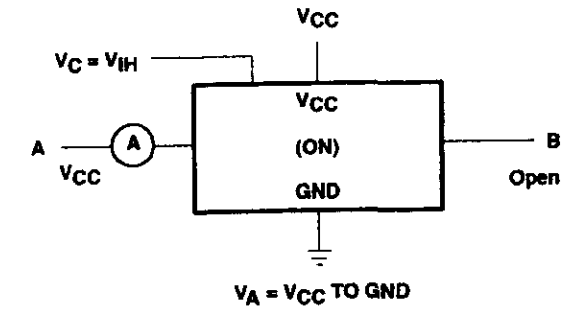


Figure 3. On-State Leakage Current Test Circuit

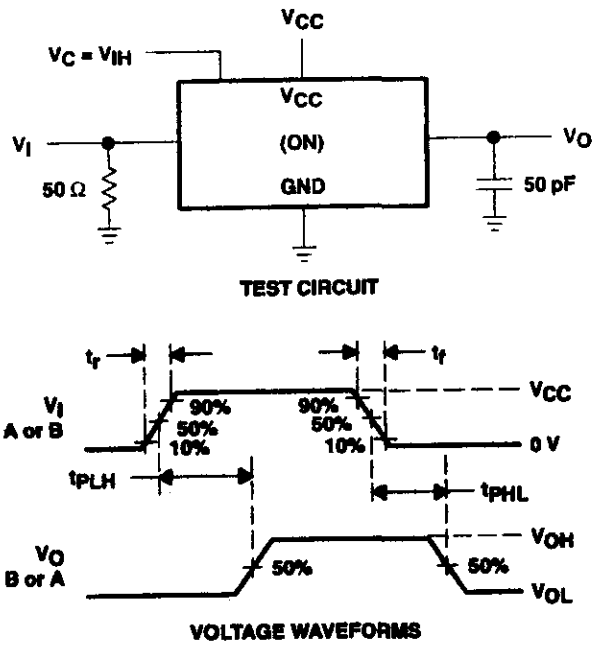
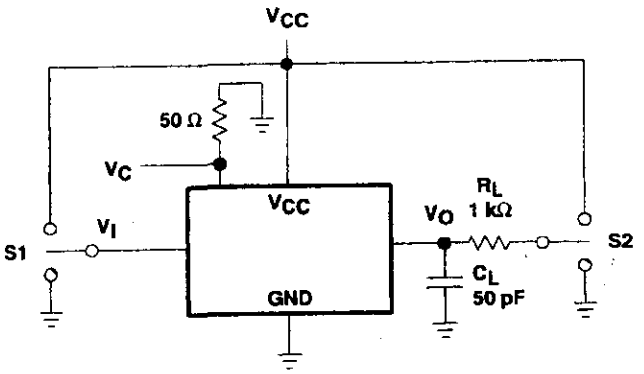


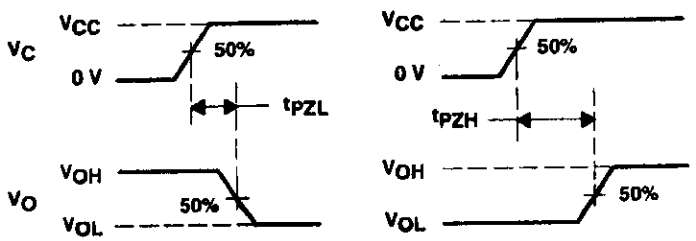
Figure 4. Propagation Delay Time, Signal Input to Signal Output

PARAMETER MEASUREMENT INFORMATION

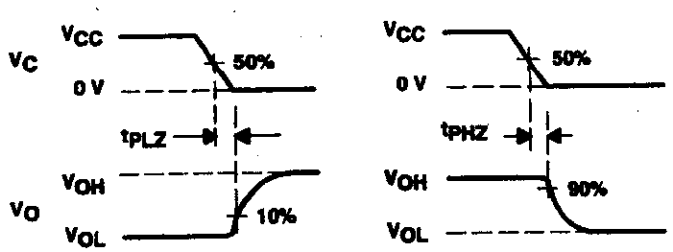


TEST	S1	S2
t_{PZL}	GND	V_{CC}
t_{PZH}	V_{CC}	GND
t_{PLZ}	GND	V_{CC}
t_{PHZ}	V_{CC}	GND

TEST CIRCUIT



(t_{PZL}, t_{PZH})



(t_{PLZ}, t_{PHZ})

VOLTAGE WAVEFORMS

Figure 5. Switching Time (t_{PZL}, t_{PLZ}, t_{PZH}, t_{PHZ}), Control to Signal Output

PARAMETER MEASUREMENT INFORMATION

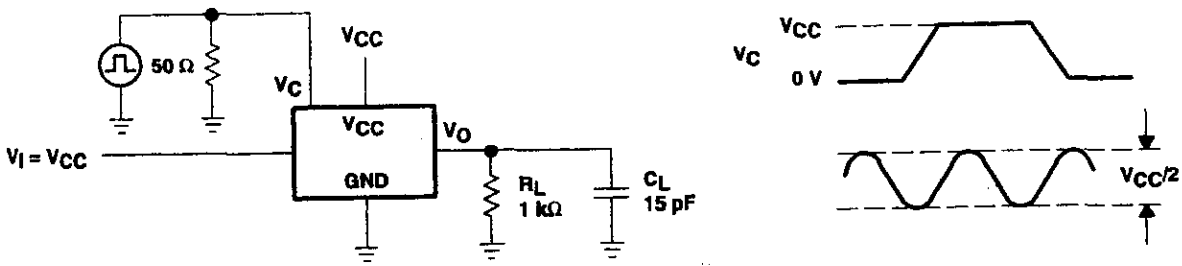


Figure 6. Control Input Frequency

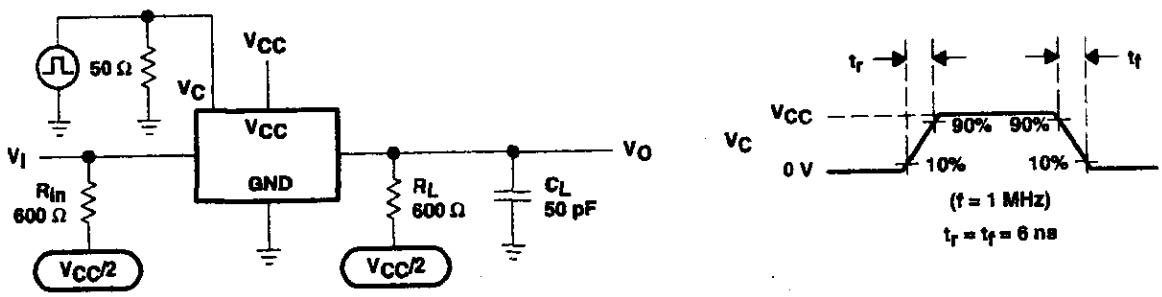


Figure 7. Control Feedthrough Noise

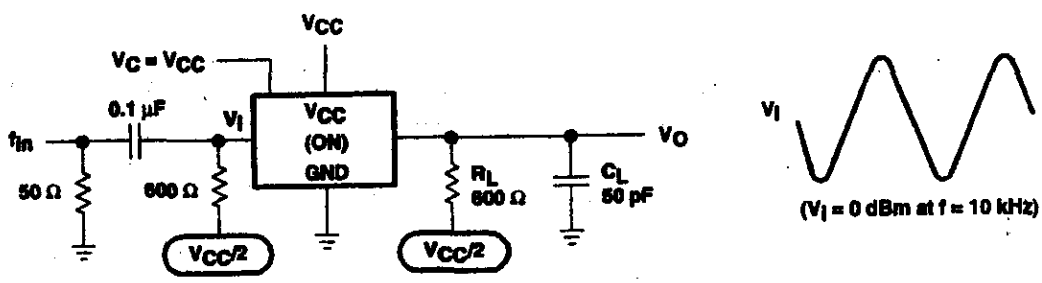


Figure 8. Minimum Through Bandwidth

TEXAS INSTRUMENTS
POST OFFICE BOX 655303 • DALLAS, TEXAS 75265

PARAMETER MEASUREMENT INFORMATION

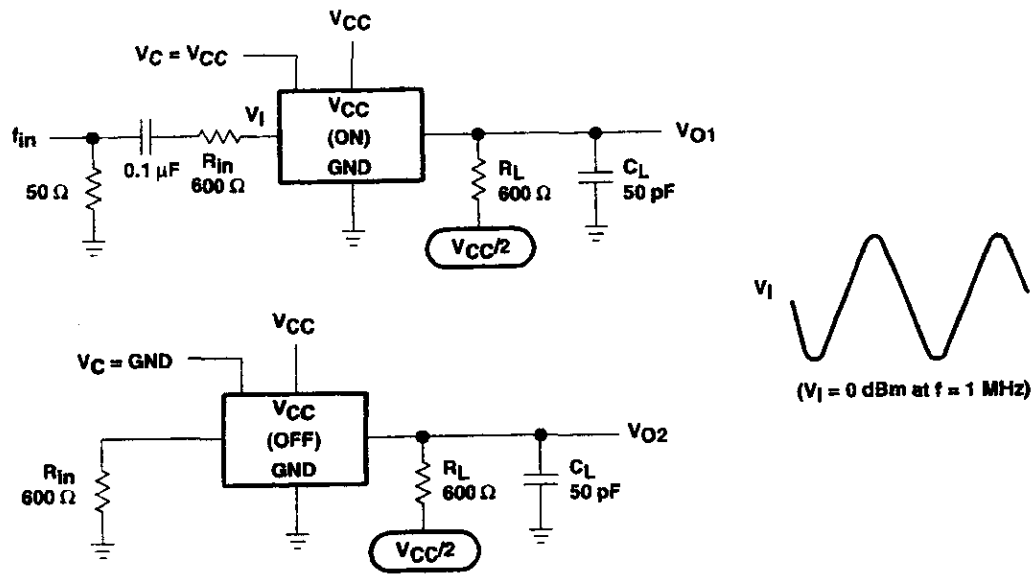


Figure 9. Crosstalk Between Any Two Switches

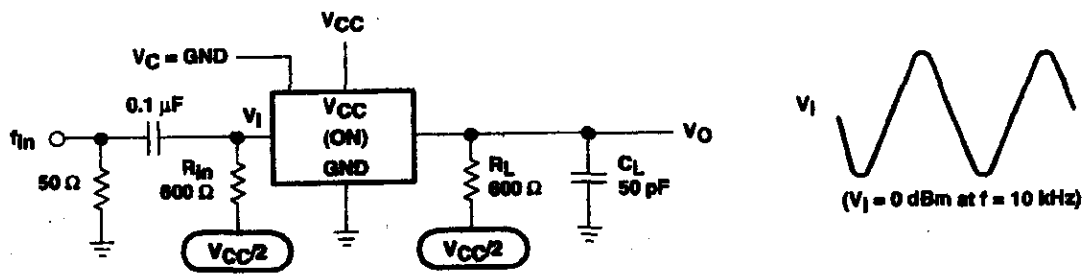


Figure 10. Feedthrough, Switch Off

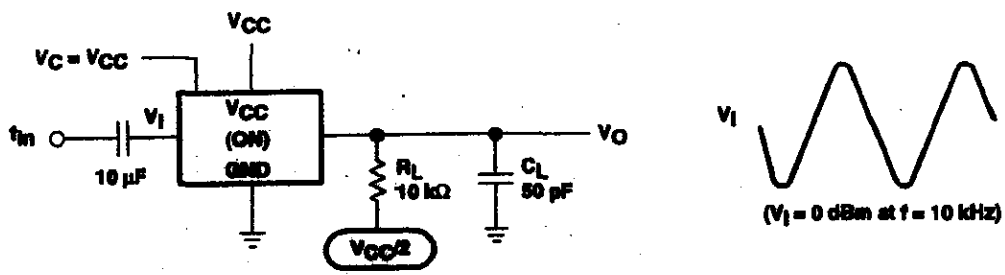


Figure 11. Amplitude Distortion Rate

IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgement, including those pertaining to warranty, patent infringement, and limitation of liability.

TI warrants performance of its semiconductor products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

CERTAIN APPLICATIONS USING SEMICONDUCTOR PRODUCTS MAY INVOLVE POTENTIAL RISKS OF DEATH, PERSONAL INJURY, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE ("CRITICAL APPLICATIONS"). TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS. INCLUSION OF TI PRODUCTS IN SUCH APPLICATIONS IS UNDERSTOOD TO BE FULLY AT THE CUSTOMER'S RISK.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance or customer product design. TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used. TI's publication of information regarding any third party's products or services does not constitute TI's approval, warranty or endorsement thereof.

National Semiconductor

January 1995

LF13508 8-Channel Analog Multiplexer
LF13509 4-Channel Differential Analog Multiplexer

General Description

The LF13508 is an 8-channel analog multiplexer which connects the output to 1 of the 8 analog inputs depending on the state of a 3-bit binary address. An enable control allows disconnecting the output, thereby providing a package select function.

This device is fabricated with National's BI-FET technology which provides ion-implanted JFETs for the analog switch on the same chip as the bipolar decode and switch drive circuitry. This technology makes possible low constant "ON" resistance with analog input voltage variations. This device does not suffer from latch-up problems or static charge blow-out problems associated with similar CMOS parts. The digital inputs are designed to operate from both TTL and CMOS levels while always providing a definite break-before-make action.

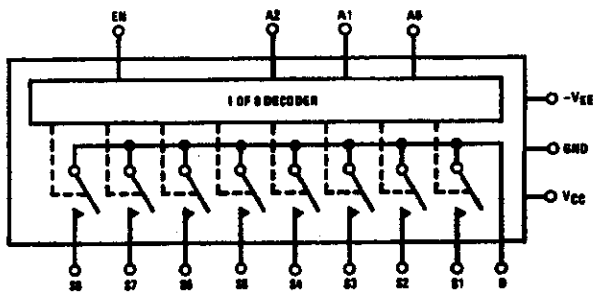
The LF13509 is a 4-channel differential analog multiplexer. A 2-bit binary address will connect a pair of independent analog inputs to one of any 4 pairs of independent analog outputs. The device has all the features of the LF13508 series and should be used whenever differential analog inputs are required.

Features

- JFET switches rather than CMOS
- No static discharge blow-out problem
- No SCR latch-up problems
- Analog signal range 11V, −15V
- Constant "ON" resistance for analog signals between −11V and 11V
- "ON" resistance 380 Ω typ
- Digital inputs compatible with TTL and CMOS
- Output enable control
- Break-before-make action: $t_{OFF} = 0.2\ \mu s$; $t_{ON} = 2\ \mu s$ typ
- Lower leakage devices available

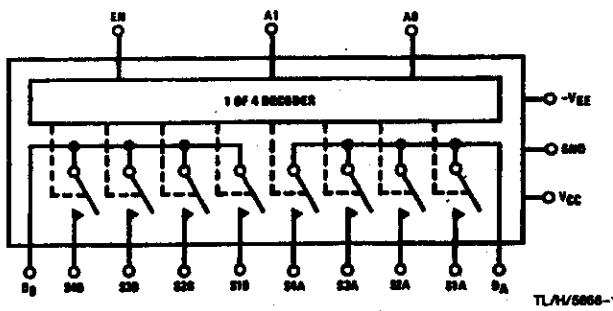
Functional Diagrams and Truth Tables

LF13508



EN	A2	A1	A0	SWITCH ON
H	L	L	L	S1
H	L	L	H	S2
H	L	H	L	S3
H	L	H	H	S4
H	H	L	L	S5
H	H	L	H	S6
H	H	H	L	S7
H	H	H	H	S8
L	X	X	X	NONE

LF13509



EN	A1	A0	SWITCH PAIR ON
L	X	X	None
H	L	L	S1
H	L	H	S2
H	H	L	S3
H	H	H	S4

TL/H/5666–1

RRD-B30M115/Printed in U. S. A.

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Positive Supply − Negative Supply ($V_{CC}-V_{EE}$)	36V
Positive Analog Input Voltage (Note 1)	V_{CC}
Negative Analog Input Voltage (Note 1)	$-V_{EE}$
Positive Digital Input Voltage	V_{CC}
Negative Digital Input Voltage	−5V
Analog Switch Current	$\|I_S\| < 10$ mA

Power Dissipation (P_D at 25°C) (Notes 2 & 7)	
Molded DIP (N) P_D	500 mW
Cavity DIP (D) P_D	900 mW
Small Outline (SO) P_D	500 mW
Maximum Junction Temperature (T_{jMAX})	100°C
Operating Temperature Range	$0°C \le T_A \le +70°C$
Storage Temperature Range	−65°C to +150°C
Lead Temperature	
D Package (Soldering, 10 seconds)	300°C
N Package (Soldering, 10 seconds)	260°C
Surface Mount Package (SO)	
Vapor Phase (60 seconds)	215°C
Infrared (15 seconds)	220°C

Electrical Characteristics (Note 3)

Symbol	Parameter	Conditions		LF13508 LF13509			Units
				Min	Typ	Max	
R_{ON}	"ON" Resistance	$V_{OUT}=0V$, $I_S=100\ \mu A$	$T_A=25°C$		380	650	Ω
					500	850	Ω
ΔR_{ON}	ΔR_{ON} with Analog Voltage Swing	$-10V \le V_{OUT} \le +10V$, $I_S=100\ \mu A$	$T_A=25°C$		0.01	1	%
R_{ON} Match	R_{ON} Match Between Switches	$V_{OUT}=0V$, $I_S=100\ \mu A$	$T_A=25°C$		20	150	Ω
$I_{S(OFF)}$	Source Current in "OFF" Condition	Switch "OFF", $V_S=11$, $V_D=-11$, (Note 4)	$T_A=25°C$			5	nA
					0.09	50	nA
$I_{D(OFF)}$	Drain Current in "OFF" Condition	Switch "OFF", $V_S=11$, $V_D=-11$, (Note 4)	$T_A=25°C$			20	nA
					0.6	500	nA
$I_{D(ON)}$	Leakage Current in "ON" Condition	Switch "ON" $V_D=11V$, (Note 4)	$T_A=25°C$			20	nA
					1	500	nA
V_{INH}	Digital "1" Input Voltage			2.0			V
V_{INL}	Digital "0" Input Voltage					0.7	V
I_{INL}	Digital "0" Input Current	$V_{IN}=0.7V$	$T_A=25°C$		1.5	30	μA
						40	μA
$I_{INL(EN)}$	Digital "0" Enable Current	$V_{EN}=0.7V$	$T_A=25°C$		1.2	30	μA
						40	μA
t_{TRAN}	Switching Time of Multiplexer	*(Figure 1)*, (Note 5)	$T_A=25°C$		1.8		μs
t_{OPEN}	Break-Before-Make	*(Figure 3)*	$T_A=25°C$		1.6		μs
$t_{ON(EN)}$	Enable Delay "ON"	*(Figure 2)*	$T_A=25°C$		1.6		μs
$t_{OFF(EN)}$	Enable Delay "OFF"	*(Figure 2)*	$T_A=25°C$		0.2		μs
$I_{SO(OFF)}$	"OFF" Isolation	(Note 6)	$T_A=25°C$		−66		dB
CT	Crosstalk	LF13509 Series, (Note 6)	$T_A=25°C$		−66		dB
$C_{S(OFF)}$	Source Capacitance ("OFF")	Switch "OFF", $V_{OUT}=0V$, $V_S=0V$	$T_A=25°C$		2.2		pF
$C_{D(OFF)}$	Drain Capacitance ("OFF")	Switch "OFF", $V_{OUT}=0V$, $V_S=0V$	$T_A=25°C$		11.4		pF
I_{CC}	Positive Supply Current	All Digital Inputs Grounded	$T_A=25°C$		7.4	12	mA
					7.9	15	mA
I_{EE}	Negative Supply Current	All Digital Inputs Grounded	$T_A=25°C$		2.7	5	mA
					2.8	6	mA

Electrical Characteristics (Continued)

Note 1: *If the analog input voltage exceeds this limit, the input current should be limited to less than 10 mA.*

Note 2: The maximum power dissipation for these devices must be derated at elevated temperatures and is dictated by T_{JMAX}, θ_{JA}, and the ambient temperature, T_A. The maximum available power dissipation at any temperature is $P_D = (T_{JMAX} - T_A)/\theta_{JA}$ or the 25°C P_{DMAX}, whichever is less.

Note 3: These specifications apply for $V_S = \pm 15V$ and over the absolute maximum operating temperature range ($T_L \leq T_A \leq T_H$) unless otherwise noted.

Note 4: Conditions applied to leakage tests insure worse case leakages. Exceeding 11V on the analog input may cause an "OFF" channel to turn "ON".

Note 5: Lots are sample tested to this parameter. The measurement conditions of *Figure 1* insure worse case transition time.

Note 6: "OFF" isolation is measured with all switches "OFF" and driving a source. Crosstalk is measured with a pair of switches "ON", driving channel A and measuring channel B. $R_L = 200$, $C_L = 7$ pF, $V_S = 3$ Vrms, $f = 500$ kHz.

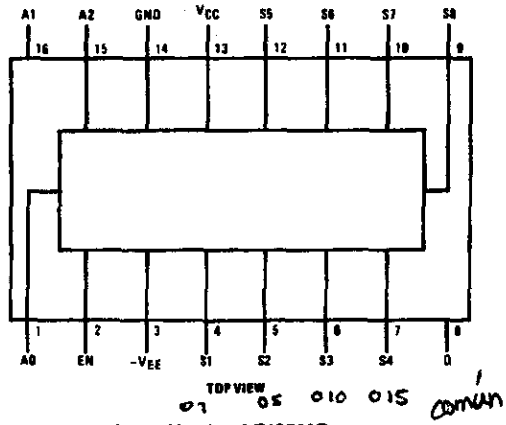
Note 7: Thermal Resistance θ_{JA} (Junction to Ambient)

Molded DIP (N)	150°C/W
Cavity DIP (D)	100°C/W

Connection Diagrams

LF13508

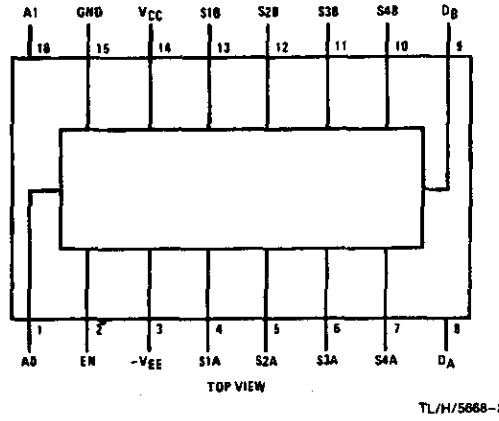
Dual-In-Line (N or D) or Small Outline (SO) Packages



Order Number LF13508D
See NS Package Number D16C
Order Number LF13508M
See NS Package Number M16A
Order Number LF13508N
See NS Package Number N16A

LF13509

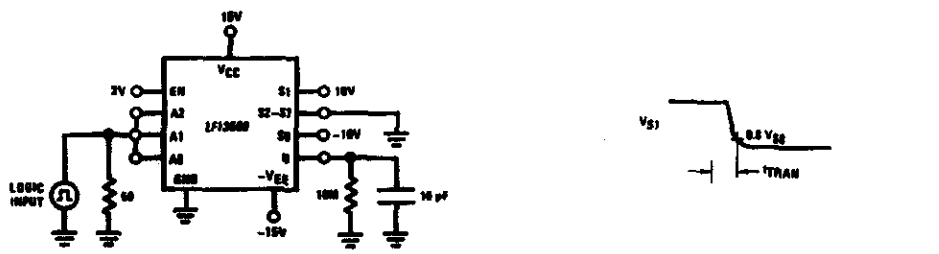
Dual-In-Line (N or D) or Small Outline (SO) Packages



TL/H/5668–2

Order Number LF13509D
See NS Package Number D16C
Order Number LF13509M
See NS Package Number M16A
Order Number LF13509N
See NS Package Number N16A

AC Test Circuits and Switching Time Waveforms



TL/H/5668–3

FIGURE 1. Transition Time