UNIVERSIDAD AUTONOMA DE BAJA CALIFORNIA

Escuela de Ingeniería

Sistema Autónomo de Validación y Prueba de Algoritmos de Procesamiento Digital de Señales

Tesis para
obtener el título de

Ingeniero en Electrónica

Presenta

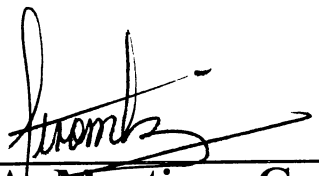
Rogelio Reyes Serrano

Ensenada B.C. Octubre 1995

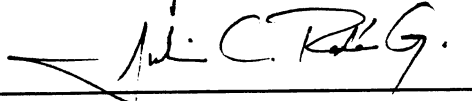
Sistema Autónomo de Validación y Prueba de Algoritmos de Procesamiento Digital de Señales

que fue desarrollado por
Rogelio Reyes Serrano

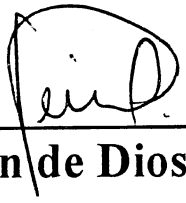
y aprobado por los sinodales



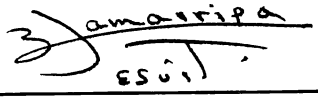
Dr. Ciro A. Martinez Garcia M.



M.C. Julio César Rolón Garrido



Ing. Juan de Dios Sánchez López



M.C. José de Jesús Zamarripa Topete

Dedicatoria.

A mis padres por haberme dado el Don de la vida.

A mis hermanos por soportarme en los momentos difíciles.

A los compañeros por el sacrificio hecho en nuestra carrera.

A nuestros maestros por su enseñanza y dedicación.

Al Dr. Ciro A. Martinez por apoyo y dirección de este trabajo.

Y a todas aquellas personas que pusieron un granito de arena para la realización de este trabajo.

Sistema Autónomo de Validación y Prueba de Algoritmos de Procesamiento Digital de Señales.

INDICE.

Lista de Figuras.

Lista de Tablas.

Capítulo I.

Introducción.

Con el avance de la tecnología de fabricación de circuitos integrados han surgido nuevos dispositivos que han permitido crear herramientas poderosas para procesar señales en forma digital. Uno de estos dispositivos es el procesador digital de señales.

Con el surgimiento de los procesadores digitales de señales y su gran potencial de procesamiento, aunados a la gran variedad de aplicaciones que tienen, surgió en el grupo de Procesamiento Digital de Señales del departamento de Electrónica y Telecomunicaciones del CICESE la necesidad de desarrollar herramientas para el procesamiento digital de señales.

Entre estas herramientas se encuentra una tarjeta de adquisición, procesamiento y síntesis de señales denominada TAPS, la cual permite desarrollar varios algoritmos para el procesamiento digital de señales, por ejemplo para la manipulación de voz y filtros digitales, entre otros.

Esta tarjeta fue diseñada para funcionar en el ducto ISA de 8 bits de una computadora personal compatible con IBM, dificultándose con ésto su traslado de un lugar a otro para su utilización, ya que se tiene que transportar con todo y computadora o instalarse junto con sus programas en otra computadora, haciendo engorroso el proceso, más aún cuando se tiene que transportar de una ciudad a otra para una demostración experimental.

Con éstas dificultades surgió la necesidad de diseñar y construir un prototipo capaz de operar la tarjeta TAPS sin la necesidad de tener que instalarla físicamente dentro de una computadora personal, debiendo tener las siguientes características:

- Ser autónomo.
- Tener capacidad de almacenamiento de algoritmos de procesamiento digital de señales en memoria de estado sólido.
- Tener capacidad de comunicación por puerto serie con una computadora personal para recibir por telecargado los algoritmos.

1.1. Objetivos.

El objetivo de este trabajo es realizar el diseño de un sistema autónomo que sea capaz de operar una tarjeta de procesamiento digital de señales sin la necesidad de utilizar en permanencia una computadora personal. Para ello, se pretende contar con un sistema capaz de almacenar en estado sólido varios algoritmos. Por otra parte, deberá ser capaz de comunicarse por puerto serie con una computadora personal y recibir algoritmos por telecargado.

1.2. Organización del trabajo.

El presente trabajo se ha estructurado de la siguiente forma:

En el capítulo II, se hace una breve descripción de la finalidad de la tarjeta TAPS, incluyendo algunas de las características principales de su procesador.

En el capítulo III, se describe el módulo de control, aspectos de operación, capacidades y aplicaciones.

En el capítulo IV, se describen los procesos de comunicación, cómo se realizan y aspectos de programación.

En el capítulo V, se detallan aspectos de la programación en alto nivel y en ensamblador, modo de operación de los mismos y sus características.

En el capítulo VI, se plantean algunos aspectos del programa ensamblador, tales como su estructura, cómo incluir algoritmos a éste, modos de operación del puerto serie, y la generación y modificación de las tasas de transferencia.

Finalmente en el capítulo VII se presentan las conclusiones.

Con el fin de evitar confusiones durante la lectura de este trabajo se hacen notar los siguientes puntos:

- Le llamamos **módulo de control** al prototipo diseñado.
- Cuando mencionamos **módulo de PDS** o **PDS** nos referimos a la tarjeta TAPS.
- En el caso en que se menciona como **DSP** o **DSP32** nos referimos al circuito integrado Procesador digital de señales como tal.

Capítulo II

Módulo de PDS.

El módulo de PDS es una tarjeta de adquisición, procesamiento y síntesis de señales (TAPS) la cual tiene como núcleo el procesador digital de señales DSP32 de AT&T. El diseño de la TAPS forma parte de la infraestructura del laboratorio de procesamiento digital de señales (PDS) del CICESE.

La TAPS es un prototipo que tiene como objetivo llevar a cabo la interacción entre el procesador digital de señales DSP32 y la computadora personal, además de permitir la adquisición y síntesis de señales. Es una tarjeta controlada por la computadora personal, cuya comunicación se realiza a través de una ranura de 62 líneas del ducto ISA.

El módulo de control [ver Capítulo III] es un prototipo diseñado para controlar la tarjeta TAPS emulando el ducto ISA de 8 bits de la computadora personal.

La forma en la cual el módulo de control se comunica con la TAPS, es enviándole a ésta los programas que ha de ejecutar; el módulo de control controla la ejecución de los programas (algoritmos) en el DSP32 de la TAPS actuando directamente sobre los registros de control de éste.

2.1 Procesador digital de señales DSP32.

El procesador digital de señales DSP32 es el componente medular de la tarjeta TAPS, y tiene las siguientes características básicas [AT&T, 1986]:

- Opera a 25 MHz
- 4096 octetos de memoria de escritura / lectura.
- 2048 octetos de memoria de lectura exclusiva pregrabada con rutinas de uso general [AT&T,1986].
- Unidad aritmética de 32 bits punto flotante (12.5 MFLOPS).
- Unidad de control de 16 bits de punto fijo (6.25 MIPS).
- 21 registros de 16 bits.
- 4 acumuladores de 40 bits punto flotante.
- Capacidad de manejo de memoria externa.

- Puerto paralelo de entrada/salida con opción de acceso directo a memoria (DMA).
- Puerto serie de entrada /salida con opción para DMA.

De las características anteriores destaca el puerto paralelo de entrada/salida denominado PIO por el cual se realiza la transferencia de información entre el DSP32 de la TAPS y el microcontrolador del módulo de control, además en él se tiene el control del PDS. Sus características son las siguientes:

El PIO tiene dos modos de transferencia: por DMA o controlado por programa. En el modo de DMA, el microcontrolador empieza la transferencia escribiendo primero la dirección de transferencia dentro de un registro interno del PIO, con la dirección inicial donde se localizan los datos, éstos pueden ser almacenados o recuperados desde la memoria del PDS. Después, el microcontrolador lee/escribe los datos desde/a el puerto paralelo. Este proceso de lectura/escritura continua hasta que el microcontrolador detiene la operación de DMA.

En el modo de programa, el microcontrolador y el PDS pueden, en general, leer/escribir desde/a el PIO al mismo tiempo que son cargados los datos en registros internos separados del PIO. La operación de lectura o escritura de el PDS en el modo de programa es terminada por la ejecución de una instrucción en el programa de aplicación del PDS. La operación de lectura o escritura del microcontrolador es a través del puerto paralelo de 8 bits. Cuando el PDS escribe al PIO, una señal puede ser usada para interrumpir al microcontrolador la cual proviene de una de las terminales de control del PIO. Existe otro tipo de modo de programa disponible, cuando tanto el PDS como el microcontrolador escriben al mismo registro interno del PIO. Este modo es destinado para la aplicación de transferencia cuando el conflicto originado por escritura simultánea de el PDS y el microcontrolador no existe.

2.1.1 Arquitectura del PIO.

El PIO consiste en un bloque de control, seis registros internos, lógica de control de error, y circuitería refrescante de memoria RAM interna del DSP.

Tiene 17 terminales para interfaz con la circuitería exterior, de las cuales ocho son bidireccionales para el ducto paralelo de datos (PDB) y nueve son de control.

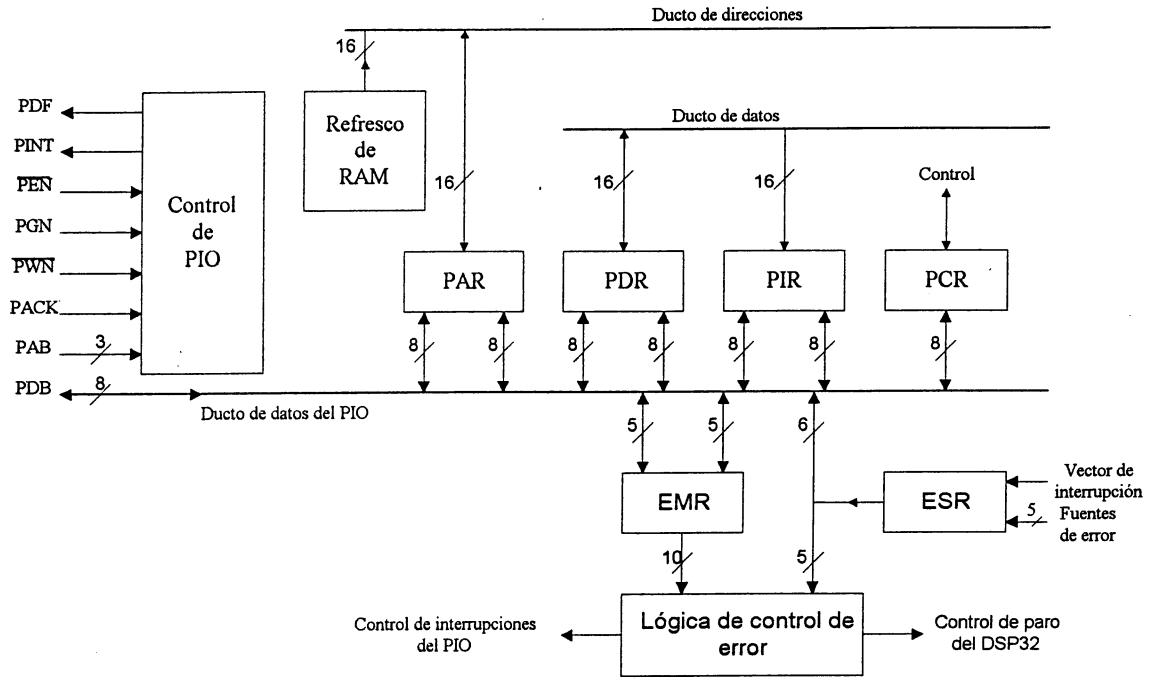


Figura 2.1 Arquitectura del PIO.

Símbolo	Nombre	Descripción
PDB0-PDB7	Ducto paralelo de datos	Ducto de ocho bits de datos bidireccionales para la interfaz con el microcontrolador.
PAB0-PAB2	Ducto paralelo de direcciones	Tres bits de direcciones para seleccionar la parte baja o alta de los registros PAR, PDR, PCR, EMR Y ESR.
PEN	Habilita la interfaz del procesador (Activa baja)	Habilita la lectura o escritura del ducto de datos (PDB)
PGN	Habilita la lectura del procesador (Activa baja)	Habilita como salida a PDB y permite que microcontrolador lea datos de un registro del PIO.
PWN	Habilita la escritura del procesador (Activa baja)	Habilita los registros del PIO para que puedan ser escritos por el microcontrolador.
PINT	Interrupciones del procesador (Activa alta)	Esta línea es usada para interrumpir al microcontrolador.
PDF	Ducto de datos lleno. (Activa alto)	Esta línea es activa cuando el PDR es escrito por el DSP o por microcontrolador y es inactiva cuando es leído.

Tabla 2.1 Descripción de las líneas del PIO.

La línea de control PACK se utiliza unicamente para leer el registro PIR (1), para leer los otros registros debe estar desactivada (0).

La tabla 2.2 muestra los registros internos del PIO

NOMBRE		CONTENIDO DE BIT
Registro de control	(PCR)	8 bits
Registro de dirección	(PAR)	16 bits
Registro de datos	(PDR)	16 bits
Registro de interrupción	(PIR)	16 bits
Registro de fuente de error	(ESR)	6 bits
Registro de máscara de error	(EMR)	10 bits

Tabla 2.2 Registros internos del PIO.

La tabla 2.3 muestra las direcciones físicas de los registros vistas por el PDS y por el módulo de control, dichas direcciones pueden ser cambiadas en la tarjeta TAPS, aunque en su diseñose tomó como la dirección base 0x300. En el módulo de control la dirección base es la 8000H ya que el espacio de direccionamiento de entrada/salida visto por el PDS se encuentra ocupado por la memoria RAM del módulo de control.

REGISTROS	Direcciones vistas por el PDS	Direcciones vistas por módulo de control
PAR BAJO	300H	8300H
PAR ALTO	301H	8301H
PDR BAJO	302H	8302H
PDR ALTO	303H	8303H
EMR BAJO DE 5 BITS	304H	8304H
EMR ALTO DE 5 BITS	305H	8305H
ESR	306H	8306H
PCR	307H	8307H
PIR MODO DE 8 BITS	308H	8308H
PIR BAJO EN 16 BITS	308H	8308H
PIR ALTO EN 16 BITS	309H	8309H

Tabla 2.3 Direcciones de los registros del PDS.

El PIO es la parte más importante, desde el punto de vista del módulo de control, en la tarjeta TAPS, debido a que por medio de éste se realiza la transferencia de información del microcontrolador al DSP32 o viceversa, además se ejecutan o detienen los programas que se encuentran en la memoria RAM del DSP32 controlando sus registros internos.

Para desarrollar el módulo de control y el programa ensamblador del mismo, fue necesario entender básicamente de la tarjeta TAPS, la arquitectura y el funcionamiento del PIO. Como cuáles son sus registros internos, la función de los mismos, direcciones en las que se encuentran y cómo se realiza el intercambio de información entre el microcontrolador y DSP a través de éstos.

Capítulo III

Módulo de control.

El módulo de control es un subsistema diseñado para controlar la tarjeta TAPS, a además cuenta con varios algoritmos de PDS almacenados en memoria de estado sólido varios algoritmos de procesamiento digital de señales, y permitir el telecargado de los algoritmos desde una computadora personal a través del puerto de comunicación serie.

La comunicación entre el módulo de control y la tarjeta TAPS se realiza por medio de una ranura de 62 líneas del ducto ISA, por lo que el módulo de control emula parcialmente el ducto ISA.

La forma en que el módulo de control interacciona con la tarjeta TAPS es enviándole los algoritmos que ha de ejecutar, y controlando su ejecución en el DSP32 actuando directamente sobre los registros de control de éste.

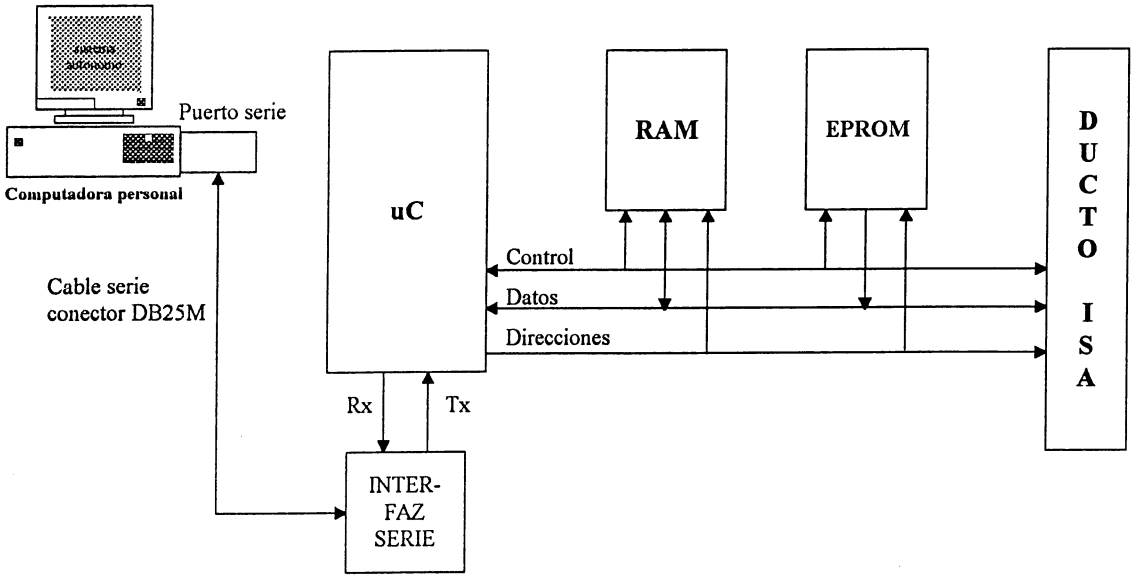


Figura 3.1 Diagrama a bloques del módulo de control.

La figura 3.1 muestra el diagrama a bloques del módulo de control en el cual se pueden identificar cuatro bloques funcionales: módulo de procesamiento, de memoria, ducto ISA e interfase serie. Éstos serán descritos a continuación.

3.1 Sub-módulo de procesamiento.

El sub-módulo de procesamiento está basado en el microcontrolador 8051 (Intel).

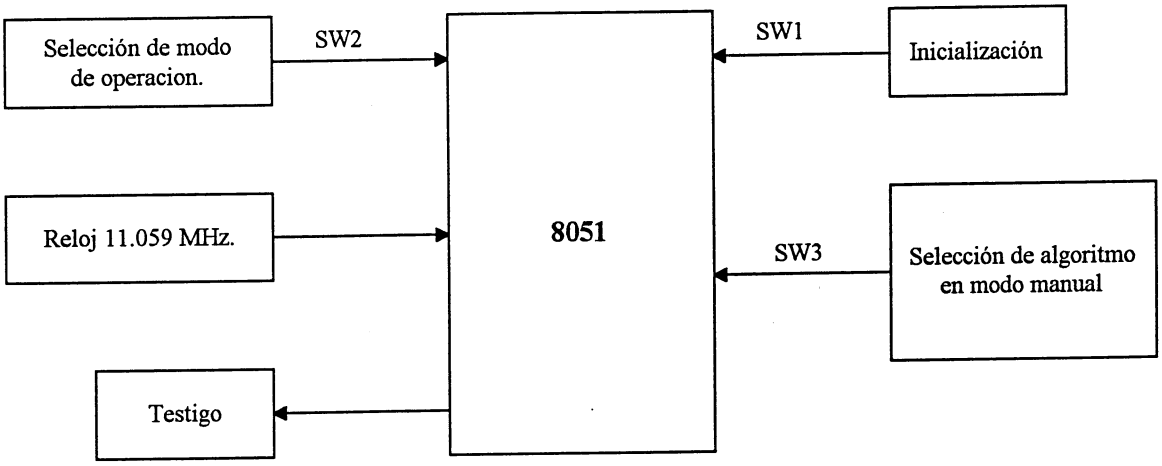


Figura 3.2 Diagrama a bloques del módulo de procesamiento.

La figura 3.2 muestra el diagrama a bloques del módulo de control en el que se pueden identificar los interruptores SW1, SW2 y SW3 así como el testigo.

El interruptor SW1 está normalmente abierto, al cerrarlo reinicializa físicamente al microcontrolador.

El interruptor SW2 permite seleccionar el modo de operación del módulo de control, en forma manual o interacción con la computadora.

Estado de Interruptor SW2	Modo de operación
0	Interacción con PC
1	Manual

Tabla 3.1 Selección del modo de operación.

El interruptor SW3 es de cuatro posiciones el cual permite seleccionar en forma binaria hasta dieciséis algoritmos que se encuentren grabados en la memoria EPROM del módulo de control, ésta selección sólo se podrá realizar cuando el modo de operación sea manual. (ver apartado 5.2.2).

El testigo es un diodo emisor de luz el cual parpadea cinco veces cada vez que el módulo de control se inicializa, esto permite saber si el programa del módulo de control está funcionando. Una vez inicializado el testigo permanece normalmente encendido.

3.2 Memoria.

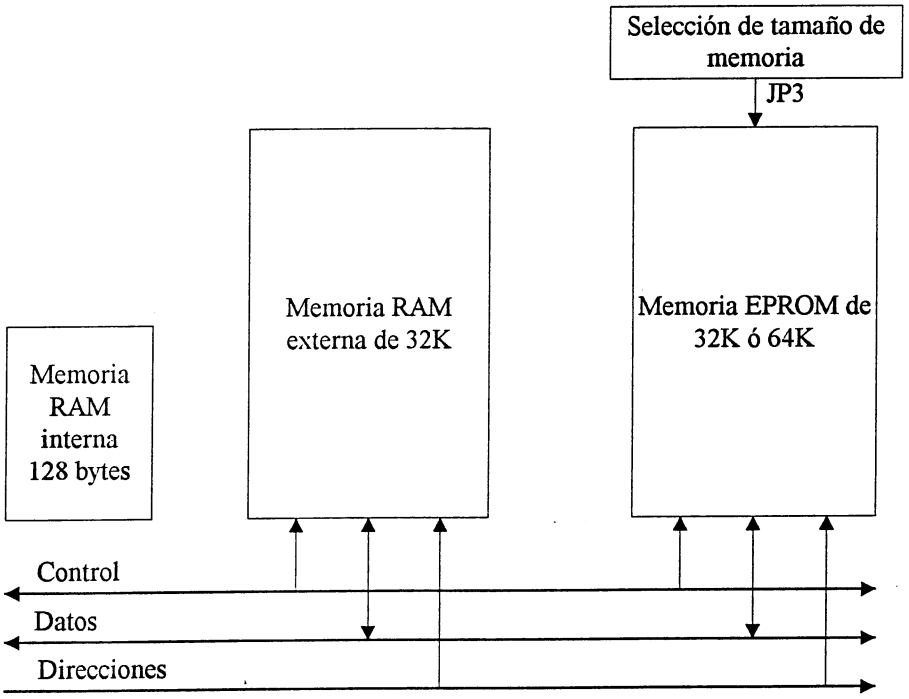


Figura 3.3 Diagrama a bloques de la memoria.

El módulo de control contiene memoria RAM y memoria EPROM, las cuales tienen las siguientes características:

- Memoria RAM de 32K x 8 bits.
- Memoria EPROM de 32K x 8 bits ó 64K x 8 bits.

El tamaño de la memoria EPROM puede ser de 32K x 8 ó 64K x 8, y se selecciona por medio del puente de conexión JP3 el cual tiene la siguiente configuración:

Puente JP3	Tamaño de memoria EPROM
1 2 3 ●—● ●	32K x 8 bits (27256)
1 2 3 ● ●—●	64 K x 8 bits (27512)

Tabla 3.2 Selección del tamaño de memoria EPROM.

3.3 Interfaz serie.

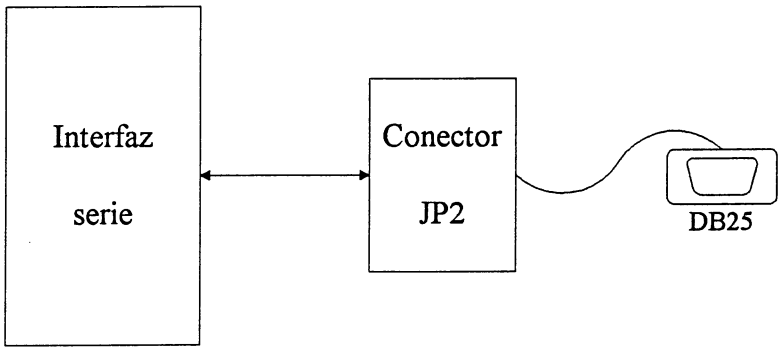


Figura 3.4 Interfaz de comunicación serie.

La interfaz de comunicación serie contiene el circuito de interfaz MAX232 con el cual se tiene transmisión y recepción en un sólo circuito integrado. Con esta interfaz se logra una tasa de transferencia de 1,200 hasta 19,200 baud siendo las tasas mínima y máxima respectivamente a las que puede comunicarse el microcontrolador con el exterior utilizando un cristal de 11.059 MHz.

El conector JP2 es del tipo molex y en el se conecta el cable serie con la configuración que se muestra en la Tabla 3.3.

Nombre	Conector JP2
Tx	1
Rx	2
GND	3

Tabla 3.3 Configuración del conector JP2 para la comunicación serie.

En el otro extremo del cable serie se tiene un conector tipo DB25M el cual se conecta al puerto serie de una computadora. El conector DB25M tiene la configuración mínima que se muestra en la Tabla 3.4.

Nombre	Conector DB25M
Tx	2
Rx	3
GND	7

Tabla 3.4 Configuración para el conector DB25M

3.4 Ducto ISA.

En la ranura del ducto ISA de 8 bits que se muestra en la figura 3.5 se inserta la tarjeta TAPS, de ésta toma las alimentaciones de voltaje para su funcionamiento, líneas de datos, direcciones y de control.

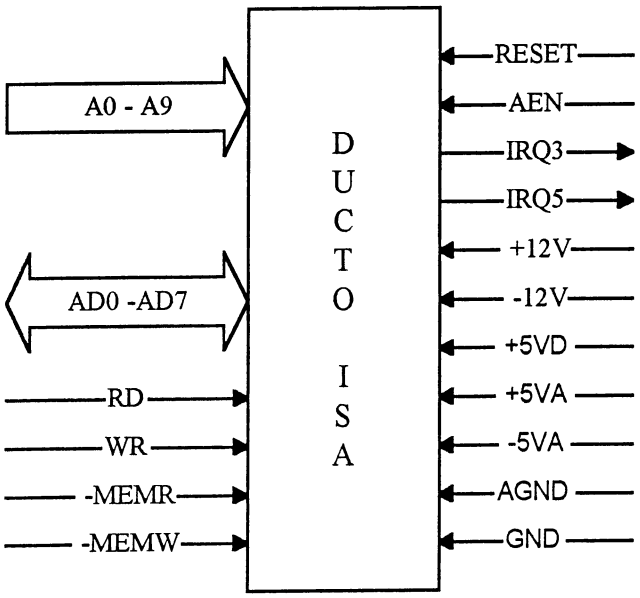


Figura 3.5 Ranura del ducto ISA de 8 bits.

Como se muestra en la figura 3.5 el ducto ISA, sólo contiene nueve bits de direcciones donde se tiene que las direcciones A0-A8 son directas y la de dirección A9* esta relacionada con la dirección A15 con la circuitería que se muestra en la figura 3.6.

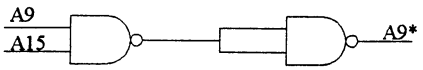


Figura 3.6 Circuito de direcionamiento con las líneas A9 y A15.

A15	A9	A9*
0	0	0
0	1	0
1	0	0
1	1	1

Tabla 3.5 Tabla de verdad del circuito de la fig. 3.6.

Con lo anterior se tiene el siguiente direccionamiento en la ranura del ducto ISA:

A9*	A8	A7	A6	A5	A4	A3	A2	A1	A0	HEX
0	0	0	0	0	0	0	0	0	0	0000
:	:	:	:	:	:	:	:	:	:	:
1	1	1	1	1	1	1	1	1	1	03FF

Tabla 3.6 Direcionamiento visto por módulo de PDS.

A15	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	HEX
1	0	0	0	0	0	0	0	0	0	0	8000
:	:	:	:	:	:	:	:	:	:	:	:
1	1	1	1	1	1	1	1	1	1	1	83FF

Tabla 3.7 Direccionamiento visto por módulo de control.

Además se tienen ocho bits de datos, líneas de control como lectura (RD), escritura (WR), interrupciones (IRQ3, IRQ5), lectura y escritura a memoria (MEMR Y MEMW), dirección activa (AEN), reinicialización y líneas de alimentación.

Las líneas de alimentación están divididas en alimentaciones digitales y analógicas con sus respectivas tierras.

Pin en ducto ISA	Nombre	Descripción
A31-A22	A0-A9*	Líneas de dirección
A9-A2	AD0-AD7	Líneas de datos
B14	RD (IOR)	Leer en periférico
B13	WR (IOW)	Escribir en periférico
B12	-MEMR	Leer a memoria
B11	-MEMW	Escribir a memoria
B25	IRQ3	Petición de interrupción
B23	IRQ5	Petición de interrupción
A11	AEN	Dirección activada
B2	RESET	Reinicialización
B9,B7	+12V -12V	Líneas de alimentación
B29	+5VD	Líneas de alimentación digital
B3,B5	+5VA -5VA	Líneas de alimentación analógica
B1	AGND	Tierra analógica
B31	GND	Tierra digital

Tabla 3.8 Descripción de las líneas emuladas del ducto ISA de 8 bits.

El diseño del módulo de control es de fácil construcción [ver apéndice A], contiene los componentes básicos de un sistema mínimo que son: microcontrolador, cerrojo, memoria RAM, EPROM y comunicación serie, además tiene la ranura del ducto ISA de 8 bits.

Capítulo IV.
Comunicación.

La comunicación es la parte fundamental de este proyecto, ya que el módulo de control se comunica en forma serie con el exterior, que en este caso es la computadora personal, y en forma paralelo entre los módulos que lo componen.

Esta se divide en dos partes:

1.- Comunicación entre el módulo de control y el módulo PDS.

2.- Comunicación entre el módulo de control y la computadora personal.

4.1 Comunicación entre el módulo de control y el módulo PDS.

La comunicación entre el módulo de control y el módulo PDS se realiza en paralelo sobre 8 bits, ya que ésta es la longitud de la palabra tecnológica del microcontrolador integrado en el módulo de control.

El diagrama a bloques de la figura 4.1 muestra las señales de control, direcciones y datos con los que se comunican el microcontrolador y procesador digital de señales (señales de lectura y escritura, líneas de direcciones y 8 líneas de datos).

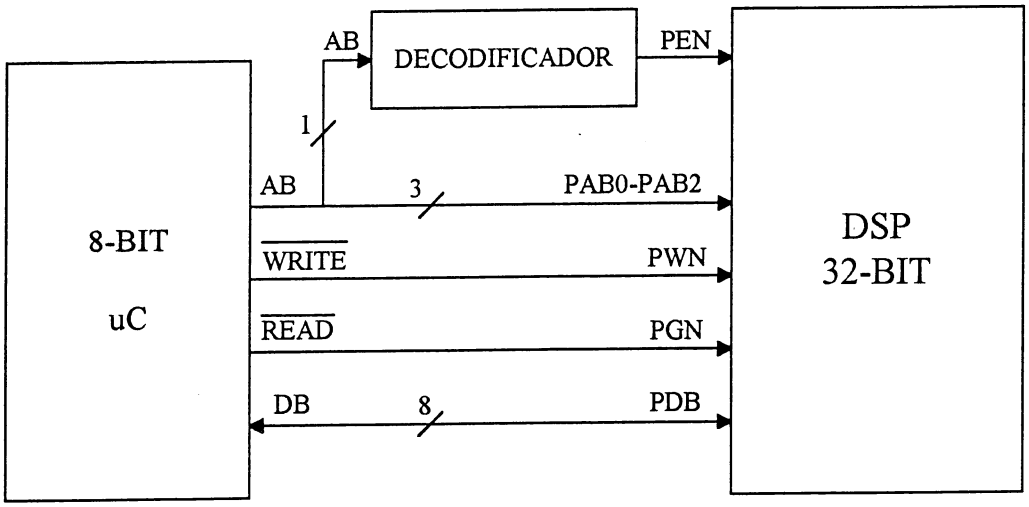


Figura. 4.1 Señales requeridas por el puerto paralelo de entrada -salida (PIO) del módulo DSP.

Cada instrucción del procesador digital de señales consta de 16 bits de direcciones y 32 bits de datos, clasificados de la siguiente manera:

DIRECCIONES		DATOS			
ALTAS	BAJAS	ALTAS		BAJAS	
		A-A	A-B	B-A	B-B
XX	XX	XX	XX	XX	XX

Tabla 4.1 Estructura de las instrucciones del PDS, X puede ser cualquier valor hexadecimal.

La forma de cargar una instrucción del módulo de control al módulo PDS es la siguiente:

Se configura el registro PCR (PIO Control Register) del PDS con la palabra de control 18H, con la cual se activa el acceso directo a memoria (DMA), así como el modo de auto incremento del registro PAR en DMA. Una vez configurado el registro PCR, el PDS esta listo para recibir información. Se manda la dirección baja al registro PAR (bajo) y luego la dirección alta al registro PAR (alto). Los datos se envían de manera similar, correspondiendo los bajo-bajo al registro PDR (bajo), los datos bajo-alto al PDR (alto), y los alto-bajo a PDR (bajo) y los alto-alto a PDR (alto).

REGISTRO PCR

BIT 7	6	5	4	3	2	1	0
REF	PIF	PDF	AUTO	DMA	ENI	INTM	RESET
0	0	0	1	1	0	0	0

Tabla 4.2 Configuración del registro PCR.

El proceso anterior se repite hasta que se encuentra la dirección 0000H, la cual indica el fin del algoritmo.

El registro PCR sólo se configura al inicio del cargado del algoritmo y no en cada instrucción.

El proceso de lectura de la memoria RAM del PDS es similar al proceso de escritura, con la diferencia de que en lugar de escribir sobre los registros PDR (bajo) y PDR (alto), éstos se leen, además de que es necesario que sean definidas las direcciones inicial y final de la lectura.

4.2 Comunicación entre el módulo de control y la computadora personal.

La comunicación entre el módulo de control y la computadora personal se realiza en serie a 9600 bauds con la ayuda de un programa de comunicación.

Es necesario que el módulo de control esté en modo de interacción con la computadora personal (SW2 = 0), para que exista la comunicación. Al entablarse ésta, el módulo de control manda el mensaje de identificación de la tabla 4.3.

```
**********************************************************
* SISTEMA AUTÓNOMO DE VALIDACIÓN Y PRUEBA DE ALGORITMOS *
*          DE PROCESAMIENTO DIGITAL DE SEÑALES           *
*         AUTOR:  ROGELIO REYES SERRANO                  *
*           VER:  1.5                        DIC,93      *
**********************************************************
```

Tabla 4.3 Mensaje de identificación.

Una vez enviado el mensaje de identificación, se transmite el menú principal con el cual se permite una interacción directa con el módulo de control. El menú es el siguiente:

```
**  ESCOGE TU OPCIÓN :     **
Correr algoritmo......(C)
Detener algoritmo.....(P)
Telecargado...........(T)
Algoritmos en EPROM...(E)
Leer RAM..............(R)
Leer RAM de PDS.......(D)
Llenar RAM..(BF)......(B)
  TU OPCIÓN ES ------>
```

Tabla 4.4 Menú principal.

Al presentarse el menú principal el usuario puede interactuar con el módulo presionando la tecla de la opción deseada, obteniendo una respuesta de acuerdo a su selección como se presenta a continuación:

Si presiona la opción :

(C) **Correr algoritmo.** Recibirá el mensaje ' Estoy corriendo ' y el PDS se encuentra corriendo el algoritmo que tiene almacenado en su memoria.

(P) **Detener algoritmo.** Recibirá el mensaje 'Estoy sin hacer nada ' y el PDS deja de correr el algoritmo que está en su memoria.

(T) **Telecargado.** Al seleccionar esta opción el módulo transmite el mensaje de la tabla 4.5 y está listo para recibir el algoritmo el cual se transmite con la ayuda del programa de comunicación, una vez recibido el algoritmo se presenta de nuevo el menú principal.

```
***  INICIO DE TELECARGADO   ***
     LISTO PARA RECIBIR ALGORITMO ...
```

Tabla 4.5 Mensaje de telecargado.

(E) **Algoritmos en EPROM.** Al ser seleccionada esta opción se despliega el submenú de la tabla 4.6 el cual lista los algoritmos que se encuentran grabados en la memoria EPROM.

```
*** ESCOGE EL ALGORITMO DESEADO ***
          GRABA.............(0)
          ECO...............(1)
          ECO2..............(2)
          ECO3..............(3)
          METAL.............(4)
          NORMAL............(5)
          RARO..............(6)
          REVERB............(7)
          SALÓN.............(8)
          ROBOT.............(9)
          BAJOS.............(A)
          DISPONIBLE........(B)
          DISPONIBLE........(C)
          DISPONIBLE........(D)
          DISPONIBLE........(E)
          DISPONIBLE........(F)
REGRESAR AL MENÚ PRINCIPAL  (R)
          TU OPCIÓN ES ---->
```

Tabla 4.6 Lista de algoritmos en memoria EPROM.

Al seleccionar una opción de éste submenú, se despliega la clave de la opción seleccionada y el siguiente texto:

'ES CORRECTA (S/N)'

si es la opción correcta, el algoritmo correspondiente es cargado a la memoria del módulo del PDS y se ejecuta, regresando enseguida al menú principal. Sí la opción no es la correcta, vuelve a presentar el menú de algoritmos.

(R) **Llenar RAM** ,(D) **Llenar RAM del PDS**, (B) **Llenar RAM** (BF), al seleccionar estas opciones el programa pregunta las direcciones inicial y final de la memoria correspondiente, ocupando con un campo de 4 dígitos hexadecimales cada una.

En la pantalla se despliega lo siguiente:

' DAME DIRECCIÓN INICIAL (XXXX) : '

al desplegarse el texto anterior el usuario puede dar la dirección inicial, al teclear el último dígito se despliega el siguiente texto:

' DAME DIRECCIÓN FINAL (XXXX) : '

al terminar de teclear el último dígito se despliega el contenido de datos que se encuentran en la RAM para las opciones (R) y (D), en el caso de la opción (B) se despliega el siguiente texto:

' CON QUE LA LLENO : '

el usuario tiene dos campos para dígitos hexadecimales, el intervalo seleccionado será llenado con los dígitos tecleados.

Capítulo V.

PROGRAMACIÓN

La programación se divide en tres secciones, dos para la computadora personal y una para el microcontrolador.

Se utiliza lenguaje de alto nivel Turbo Pascal para el desarrollo del programa MODIFICA, lenguaje ensamblador de la familia del microcontrolador para el programa de control del microcontrolador MCPDS, y un programa de comunicación.

5.1 Programa para modificar algoritmos, MODIFICA.

El programa MODIFICA tiene la finalidad de modificar los archivos de los algoritmos que se cargan al módulo PDS (dichos archivos están escritos en lenguaje para el DSP32), estos archivos tienen la extensión COD . El programa genera dos archivos a partir de un archivo con extensión COD, los cuales tienen como extensión A51 y TEL.

Los archivos con extensión A51 son incluidos en el programa ensamblador para ser grabados en la memoria EPROM del módulo de control, mientras que los archivos con extensión TEL son cargados al módulo de control a través de una computadora.

La forma de modificar los archivos con extensión COD es la siguiente:

Corra el ejecutable MODIFICA y teclee el nombre del archivo a modificar sin extensión, por ejemplo, si deseamos modificar el archivo ***normal.COD****, teclee sólo* ***normal*** *y el programa genera* ***normal.A51*** *y* ***normal.TEL****.*

Archivos con extensión COD. Estos archivos contienen los algoritmos generados para que se ejecuten en el procesador digital señales, dichos archivos están en código ASCII con el formato que se muestra en la tabla 5.1.

0000	00000000	instrucción inicial.
0004	0100000E	
0008	0000001F	
0020	1111FFFF	
0024	00231B00	
..	..	
..	..	
..	..	
..	..	
0120	11111111	instrucción final.

Tabla 5.1 Formato de los archivos con extensión COD.

donde cada instrucción contiene :

Direcciones	Tabulador	Datos	Retorno de carro y cambio de linea
0000	Tab	12345678	CR, LF

Tabla 5.2 Formato de una instrucción para el PDS.

Una instrucción está compuesta por cuatro octetos de direcciones, un tabulador, ocho octetos de datos, un retorno de carro (CR) y un cambio de línea (LF).

Archivos con extensión TEL. Estos archivos son una modificación de los archivos con extensión COD para que puedan ser telecargados al módulo de control a través del puerto serie, y éste a su vez los cargue al módulo PDS para su ejecución.

La modificación que se realiza en los archivos, es agregar dos líneas al final del archivo original: una de cuatro octetos y la otra de un octeto al final de éstos. Estas líneas se utilizan para detectar el fin del archivo o algoritmo en diferentes etapas.

La línea de cuatro octetos es una secuencia de cuatro ceros en la posición de las direcciones. Estos ceros se utilizan para detectar el fin del algoritmo cuando se está cargando desde la memoria RAM del módulo de control a la memoria RAM del módulo PDS. El programa de control al detectar la dirección [0000], salta la primera instrucción la cual está por lo general en la dirección [0000] , deja de cargar el algoritmo y lo manda a ejecutar.

La última línea de un octeto es la @ la cual se utiliza para detectar el fin del archivo cuando es telecargado de la computadora al módulo de control. Al detectar el programa de control el código hexadecimal de la @ deja de recibir y procede a cargar el algoritmo a la memoria RAM del módulo PDS.

El formato que tiene los archivos con extensión TEL es el que se muestra en la tabla 5.3.

0000	00000000	instrucción inicial.
0004	0100000E	
0008	0000001F	
0020	1111FFFF	
0024	00231B00	
..	..	
..	..	
..	..	
..	..	
0120	11111111	instrucción final.
0000		fin del algoritmo, módulo de control al PDS
@		fin de algoritmo, computadora al módulo de control

Tabla 5.3 Formato de los archivos con extensión TEL.

Archivos con extensión A51. Estos archivos están modificados para ser incluidos como macro instrucciones en el programa ensamblador del microcontrolador y puedan ser cargados desde la memoria EPROM del módulo de control a la memoria RAM del módulo PDS.

La modificación que se realiza al programa original en cada línea de instrucción es la siguiente:

Se le agrega al inicio de cada línea la instrucción DB, se le añade un cero al inicio de cada par de octetos y una H al final de éstos. Cada par es separado por una coma, y se elimina el caracter del tabulador para que cada línea quede con el siguiente formato:

DB 000H, 001H, 002H, 003H, 004H, 0FFH

También se le agrega a éste formato una línea al final conteniendo ceros, los cuales servirán como indicadores de fin del algoritmo para el programa de control.

Quedando entonces el siguiente formato para un archivo con extensión A51:

DB 000H,000H, 000H,000H,000H,000H	instrucción inicial.
DB 000H,004H, 001H,000H,000H,00EH	
DB 000H,008H, 000H,000H,000H,01FH	
DB 000H,020H, 011H,011H,0FFH,0FFH	
DB 000H,024H, 000H,023H,01BH,000H	
.. ..	
.. ..	
.. ..	
.. ..	
DB 001H,020H, 011H,011H,011H,011H	instrucción final.
DB 00H,00H, 00H,00H,00H,00H	fin del algoritmo

Tabla 5.4 Formato de los archivos con extensión A51.

5.2 Programa de control, MCPDS.

La función principal del programa de control es la de cargar los algoritmos al módulo de PDS (ya sea que estén grabados en memoria EPROM o que sean telecargados desde una computadora personal), ejecutar o interrumpir un algoritmo que se encuentra en

la memoria del módulo del PDS, y comunicar al módulo de control en serie con una computadora personal.

El programa de control funciona de dos maneras :

1.- Interactuando con una computadora o terminal.

2.- En forma manual.

El modo de operación se selecciona por medio del interruptor SW2 (tabla 3.1), el estado del interruptor es leído por medio del puerto 3.5.

5.2.1 Modo de interacción con la computadora o terminal.

En el modo de interacción con la PC se tiene un menú principal cuyas opciones se muestran en la tabla 4.4.

Correr el algoritmo. - Al seleccionar esta opción se configura el registro PCR el cual se encuentra en la dirección 307H para el módulo del PDS, y en la dirección 8307H para el módulo de control, con la palabra de control 01H la cual activa al PDS para que ejecute el algoritmo que tiene grabado en su memoria. Al término de la configuración el programa regresa al menú principal.

REGISTRO PCR

BIT 7	6	5	4	3	2	1	0
REF	PIF	PDF	AUTO	DMA	ENI	INTM	RESET
0	0	0	0	0	0	0	1

Cuando: RESET = 0 PARAR, RESET = 1 CORRER.

Tabla 5.5 Configuración del Registro PCR para activar al PDS.

Detener algoritmo.- Configura el registro PCR con la palabra de control 00H con la cual se interrumpe al PDS por medio del bit 0 del registro PCR el cual corresponde al RESET y lo mantiene en estado de espera hasta que se ponga en estado alto el bit 0. Al término de la configuración el programa regresa al menú principal.

Telecargado.- Al seleccionar esta opción el módulo de control transmite un mensaje a la terminal o computadora el cual se despliega en pantalla, ver tabla 5.6, al término del mensaje el módulo de control está listo para recibir el algoritmo en ASCII, cuando se recibe caracter por caracter, se filtran los caracteres de cambio de línea (LF), retorno de carro (CR), tabulador, espacio y @ que indica el fin del archivo, los que corresponden a direcciones y datos se convierten a hexadecimal y se guardan en la memoria RAM.

*** INICIO DE TELECARGADO ***
LISTO PARA RECIBIR ALGORITMO . . . . .

Tabla 5.6 Mensaje en telecargado.

Cada instrucción del algoritmo consta de 12 caracteres ASCII y cada caracter es de 8 bits, que al convertirlos a hexadecimal, sólo nos interesan los cuatro bits menos significativos, debido a que con cuatro bits podemos representar cualquier número hexadecimal. Con ello eliminamos los cuatro bits más significativos y unimos dos caracteres para formar uno solo. Esto es:

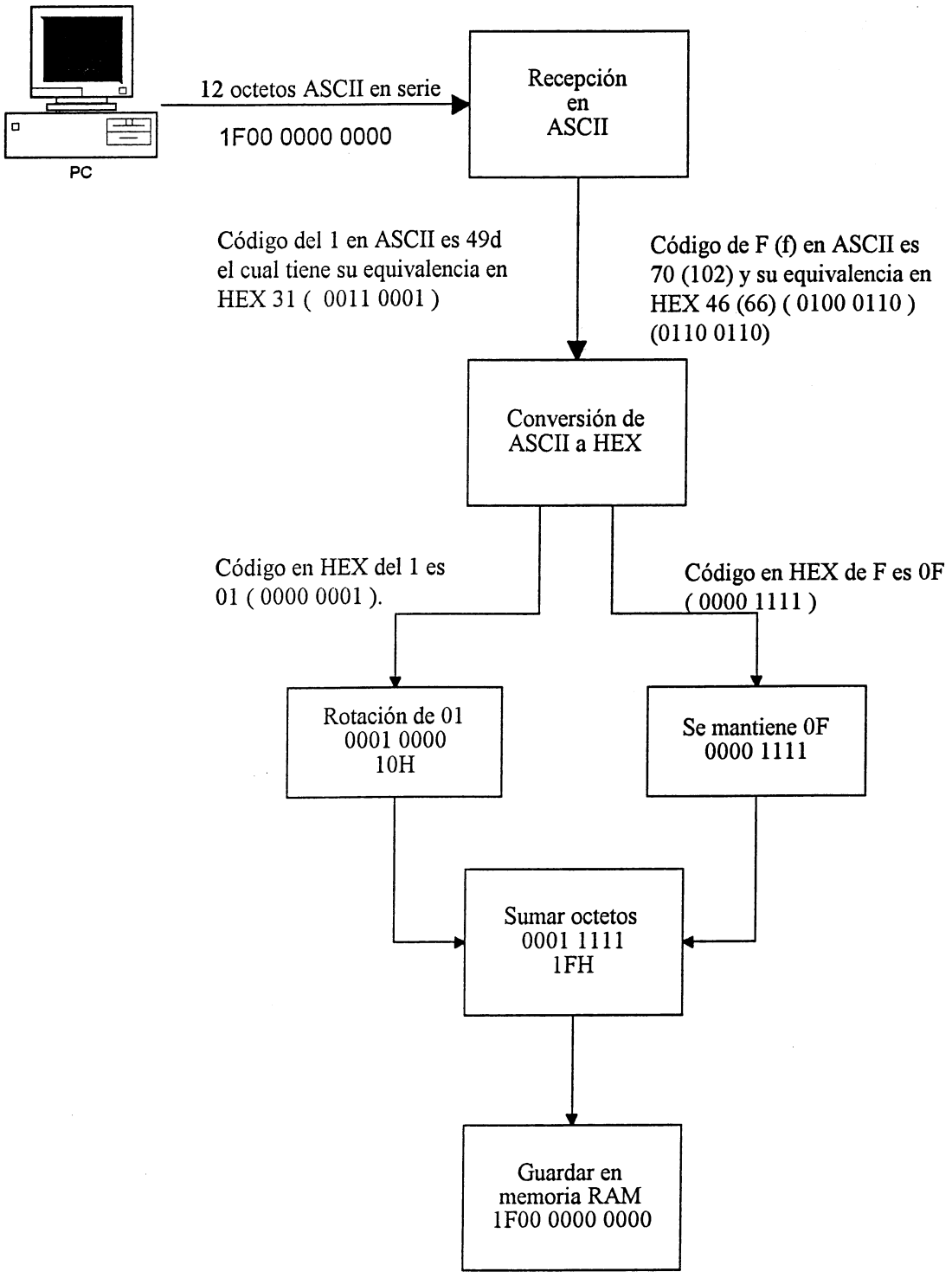


Figura 5.1 Conversión de datos de ASCII a Hexadecimal.

Al recibir el primer caracter en ASCII con el siguiente formato (XXXX XXXX), donde X puede ser 0 o 1 lógico , éste se convierte a hexadecimal obteniéndose el siguiente formato (0000 XXXX). Enseguida se hace una rotación de tal forma que se obtiene (XXXX 0000) el cual se guarda en un registro. Luego se recibe el segundo caracter realizándose la conversión ASCII a hexadecimal obteniéndose el mismo formato (0000 YYYY). Una vez que se tienen los dos octetos se suman obteniéndose uno sólo (XXXX YYYY) y se almacena en la memoria RAM. Este proceso se repite "n" veces hasta que se detecta el caracter "@ ", el cual indica el fin del algoritmo.

Con el proceso anterior se comprimen los algoritmos 60% debido a que cada instrucción en ASCII consta de 12 octetos y su equivalente en hexadecimal en 6, ésto es que un caracter en ASCII se representa en 8 bits y ese mismo caracter en hexadecimal en 4 bits, además se filtran los caracteres de cambio de línea (LF), retorno de carro (CR) y tabulador (TAB).

Al comprimirse los algoritmos un 60% se tiene que la longitud máxima de un algoritmo en ASCII es de 80 Kbytes para que pueda ser almacenado en la memoria RAM de 32 Kbytes por medio del proceso de Telecargado.

Una vez que se tiene almacenado el algoritmo en la memoria RAM del módulo de control se procede a almacenarlo a la memoria del PDS.

Algoritmos en EPROM.- Esta opción contiene un submenú en el cual se listan los nombres de los algoritmos que se tienen almacenados en la memoria EPROM. Estos pueden ser seleccionados con el número o la letra que le corresponda.

```
*** ESCOGE EL ALGORITMO DESEADO ***
        GRABA.............(0)
        ECO...............(1)
        ECO2..............(2)
        ECO3..............(3)
        METAL.............(4)
        NORMAL............(5)
        RARO..............(6)
        REVERB............(7)
        SALÓN.............(8)
        ROBOT.............(9)
        BAJOS.............(A)
        DISPONIBLE........(B)
        DISPONIBLE........(C)
        DISPONIBLE........(D)
        DISPONIBLE........(E)
        DISPONIBLE........(F)
REGRESAR AL MENÚ PRINCIPAL  (R)
        TU OPCIÓN ES ---->
```

Tabla 5.7 Menú de algoritmos en memoria EPROM.

Al seleccionar una opción aparece la clave de ésta y la pregunta si es correcta la opción que se seleccionó. Sí la respuesta es negativa se presenta de nuevo el submenú, y si es afirmativa se carga el algoritmo seleccionado en la memoria del PDS y lo ejecuta, regresándose el control al menú principal.

Leer la RAM.- Al seleccionar ésta opción, el programa pregunta la dirección inicial y final desplegando el contenido de la memoria RAM del módulo de control entre las direcciones dadas.

Leer RAM del PDS.- Su operación es igual que la opción anterior, pero ésta despliega el contenido de la memoria RAM del módulo PDS.

Llenar RAM (BF).- Al ser seleccionada ésta opción, el programa pregunta la dirección inicial y final así como los datos con que se quiere llenar el rango seleccionado de la memoria RAM del módulo de control. Los datos pueden ir desde 00 hasta FF.

5.2.2 Modo manual.

En el modo manual sólo se pueden cargar en la memoria del módulo PDS los algoritmos que se encuentran grabados en la memoria EPROM. De éstos algoritmos sólo se pueden seleccionar los primeros 16 debido a la manera en que se seleccionan, en forma binaria, por medio de un interruptor de 4 posiciones.

La forma de seleccionar un algoritmo es la siguiente, se tiene un menú de algoritmos ver tabla 4.6.

Si por ejemplo se selecciona el algoritmo GRABA, se utiliza el número 0 en binario.

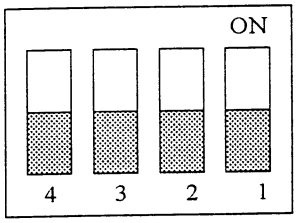


Figura 5.2 Interruptor SW3 de 4 posiciones para selección de algoritmos en modo manual.

Una vez puesto en el interruptor SW3 el número 0 en forma binaria se presiona el interruptor de reinicialización SW1, al inicializarse, el sistema lee la combinación binaria del interruptor SW3 (0) cargando el algoritmo que se encuentra en esa combinación, en éste caso el algoritmo GRABA.

Para seleccionar otro algoritmo se pone la combinación binaria que le corresponda, mientras no sea presionado el interruptor de reinicialización el algoritmo GRABA que se encuentra en la memoria del módulo de PDS se sigue ejecutando, al presionar reinicialización el sistema se reinicializa y carga a la memoria del módulo de PDS el algoritmo seleccionado y lo ejecuta.

Capítulo VI

Programa ensamblador.

Una de las bondades que ofrece el macro ensamblador de la familia del microcontrolador (MCS 51) es que permite tener una programación modular, con lo cual se facilita la programación para incluir o excluir algoritmos, así también facilita tener de 1 a "n" algoritmos siempre y cuando la suma de los "n" algoritmos más el programa principal no rebase los 64 Kbytes de memoria EPROM que se tienen en el módulo de control.

La figura 6.1 muestra esquemáticamente como se tiene la modularidad del programa ensamblador MCDSP.A51.

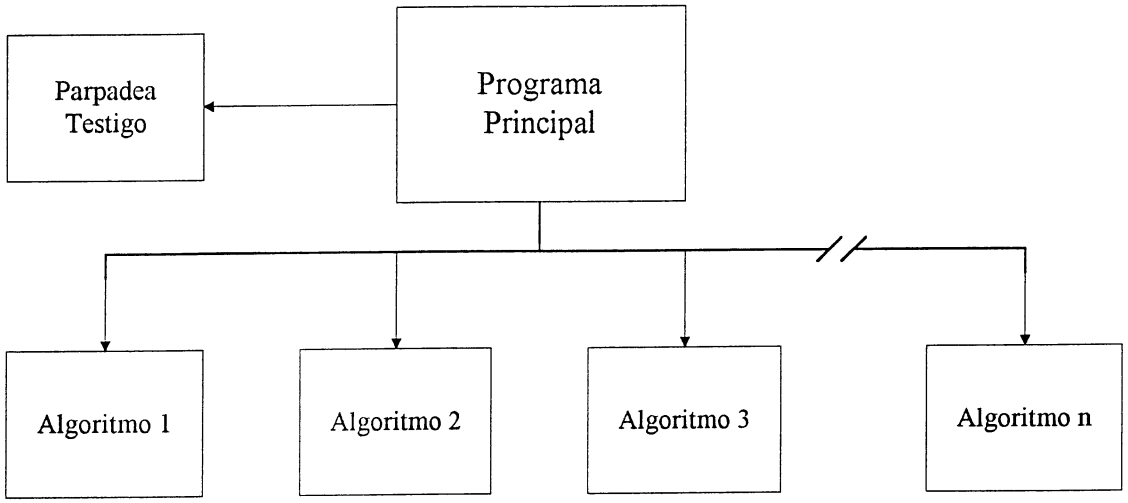


Figura 6.1 Diagrama a bloques del programa en ensamblador MCDSP.A51.

La estructura del programa ensamblador es la siguiente :

a) Títulos y comentarios.
b) Equivalencias.
c) Programa principal.
d) Subrutinas.
e) Macro instrucciones (algoritmos).

Las macro instrucciones son programas externos que el ensamblador incluye al programa ensamblador MCDSP.A51. Dentro del programa principal se encuentra la

macro instrucción parpadea testigo, y los algoritmos se encuentran al final del programa como macro instrucciones las cuales tienen el formato de la tabla 5.4 del capítulo V.

6.1 Inclusión de algoritmos al programa ensamblador.

La forma de incluir un algoritmo al programa es la siguiente:

Si tiene la tabla 4.6 de algoritmos y desea incluir el algoritmo PRUEBA.A51 en la posición (B), modifique en el programa principal MCDSP.A51 la tabla escribiendo el nombre del algoritmo en la posición (B) en este caso PRUEBA.

Una vez modificada la tabla en la sección de las macro instrucciones localice las siguientes instrucciones:

```
ALGO_B:
                    LCALL CONFIGURA
                    LCALL CARACTERES
;INCLUDE(NOMBRE.A51)
;GENONLY
                    LJMP        CORRE
```

Para incluir el algoritmo es necesario quitar los puntos y comas de las instrucciones y escribir en la instrucción INCLUDE(NOMBRE.A51) el nombre del algoritmo a incluir que en este caso es PRUEBA.A51, quedando :

```
ALGO_B:
                    LCALL CONFIGURA
                    LCALL CARACTERES
INCLUDE(PRUEBA.A51)
GENONLY
                    LJMP        CORRE
```

Una vez hechos los cambios anteriores se compila el programa MCDSP.A51 con el programa COMPILA.BAT. Para hacer el compilado es necesario que se encuentren en la misma trayectoria del programa MCDSP.A51, el programa MACROST3.A51, el cual contiene las instrucciones para que parpadeé el testigo, así como todos los algoritmos que van a ser incluidos en el programa.

El programa COMPILA.BAT contiene las mandos:

```
ASM51 %1.a51
OH   %1.OBJ
```

y se invoca COMPILA MCDSP, al compilarse se generan los archivos : MCDSP.OBJ, MCDSP.HEX y MCDSP.LST

Dependiendo del programador de memorias EPROM se utilizan los archivos con extensión OBJ y HEX, el archivo con extensión LST nos sirve para analizar la programación ya que incluye los códigos de las instrucciones y datos.

6.2 Comunicación serie.

El puerto de comunicación serie es full duplex, es decir, puede transmitir y recibir simultáneamente. El puerto serie tiene registros de transmisión y recepción que son accesados por el registro de función especial SBUF. Cuando se escribe a SBUF se carga al registro de transmisión y cuando se lee a SBUF se accesa al registro de recepción.

El puerto serie puede operar de 4 modos:

Modo 0: Los datos serie entran y salen por RXD. Por TXD salen por medio de un reloj de corrimiento. 8 bits son transmitidos/recibidos : 8 bits de datos (primero el menos significativo). La tasa de transmisión es 1/12 de la frecuencia del oscilador del sistema.

Modo 1: 10 bits son transmitidos por TXD o recibidos por RXD: un bit de inicio (0), 8 bits de datos (primero el menos significativo) y un bit de paro (1). En la recepción, el bit de paro se pone en RB8 del registro de función especial SCON. La tasa de transferencia es variable (ver apartado 6.2.1).

Modo 2: 11 bits son transmitidos por TXD o recibidos por RXD: un bit de inicio (0), 8 bits de datos (primero el menos significativo), un noveno bit de datos programable y un bit de paro (1). En transmisión el noveno bit (TB8 en SCON) se le puede asignar el valor de "0" o "1". O, por ejemplo, el bit de paridad P del PSW (Program Status Word) puede ser movido a TB8. En recepción el noveno bit se pone en RB8 del registro SCON, cuando el bit de paro es ignorado. La tasa de transferencia puede ser 1/32 o 1/64 de la frecuencia del oscilador.

Modo 3: El modo 3 es igual al modo 2 excepto que la tasa de transferencia es variable.

En los cuatro modos, la transmisión es iniciada por una instrucción que use el registro SBUF como destino. Y la recepción es iniciada en el modo 0 por la condición RI = 0 y REN =1 y en los otros modos por la entrada de un bit de inicio cuando REN = 1.

La selección del modo de operación del puerto serie se realiza configurando el registro de control del puerto serie SCON. Este registro puede ser direccionado por bits.

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Tabla 6.1 Registro de control del puerto serie SCON.

SM0	SCON.7	Selección del modo de operación del puerto serie.
SM1	SCON.6	Selección del modo de operación del puerto serie.
SM2	SCON.5	Habilita la comunicación
REN	SCON.4	Habilita (1) o deshabita (0) la recepción.
TB8	SCON.3	Es el noveno bit que será transmitido en los modos 2 y 3, se activa o desactiva por programación.
RB8	SCON.2	Es el noveno bit que se recibe en los modos 2 y 3. En el modo 1, si SM2 = 0, RB8 es el bit de paro es que se recibe, en el modo 0 no es usado.
TI	SCON.1	Bandera de interrupción en transmisión, se limpia por programa.
RI	SCON.0	Bandera de interrupción en recepción, se limpia por programa.

SM0	SM1	Modo	Descripción	Tasa de transferencia
0	1	0	Registro de Corrimiento	fosc / 12
0	0	1	8 bit UART	Variable
1	1	2	9 bit UART	fosc / 32 o fosc / 64
1	0	3	9 bit UART	Variable

Tabla 6.2 Selección del modo de operación del puerto serie.

Las palabras de control para configurar el puerto serie se muestran en la tabla 6.4.

Modo	SCON	SM2
0	10H	SM2 = 0
1	50H	
2	90H	
3	D0H	

Tabla 6.3 Palabras de control para configurar puerto serie.

6.2.1 Generación de las tasas de transferencia.

Las tasas de transferencia de la comunicación serie son de 1200 hasta 19200 bauds las cuales están regidas por la ecuación :

$$\text{Tasa Transferencia} = \frac{K \times fosc}{32 \times 12 \times [256 - TH1]}$$

Despejando TH1

$$TH1 = 256 - \frac{K \times fosc}{384 \times \text{Tasa Transferencia}}$$

donde :

TH1 = Reloj / Contador 1

fosc = 11.0592 MHz. frecuencia del cristal.

K=1 Si SMOD = 0 (SMOD se encuentra en el registro PCON)

K=2 Si SMOD = 1

Cuando es usado el reloj 1 para generar las tasas de transferencia y SMOD = 1, la tasa de transferencia es el doble cuando el puerto serie es usado en los modos de reloj 1,2 o 3.

Tasa de Transferencia	fosc MHz	SMOD	Reloj 1		
			C / T	Modo	TH1
1200	11.0592	0	0	2	E8H
2400	11.0592	0	0	2	F4H
4800	11.0592	0	0	2	FAH
9600	11.0592	0	0	2	FDH
19200	11.0592	1	0	2	FDH

Tabla 6.4 Configuración de tasas de transferencia del puerto serie.

donde:

C / T es el selector de reloj o contador (Counter or Timer) T=0 , C=1

Modo es el modo de operación del reloj y se selecciona en el registro TMOD.

RELOJ 1				RELOJ 0			
GATE	C / T	M1	M0	GATE	C / T	M1	M0

Tabla 6.5 Registro de control de modo TMOD.

M1 y M0 bits selectores de modo de reloj.

M1	M0	Modo de operación	Función del Reloj 1	TMOD Palabra de control
0	0	0	13 bit	00H
0	1	1	16 bit	10H
1	0	2	8 bits Auto recargable	20H
1	1	3	no corre	30H

Tabla 6.6 Selección del modo de operación del reloj.

La comunicación serie esta programada para trabajar en el modo 1 del puerto serie y el modo 2 del reloj a una tasa de 9600 bauds.

Si se desea cambiar estos parámetros solo se tiene que cambiar en la tabla de equivalencias del programa ensamblador [ver anexo A] los siguientes parámetros:

Para cambiar el modo de operación del puerto serie, seleccione la palabra de control de la tabla 6.4 y escríbala en la etiqueta:

```
SERIE EQU 50H    ; Modo 1 del puerto serie.
```

Para modificar la tasa de transferencia, seleccione la palabra de control de la columna TH1 de la tabla 6.5 y escríbala en etiqueta:

```
FRE   EQU 0FDH ;Frecuencia de operación de 9600 Bauds.
```

Para modificar el modo de operación del reloj, seleccione la palabra de control de la tabla 6.7 y escríbala en la etiqueta:

```
REL   EQU 20H   ;Modo 2 del reloj.
```

Conclusiones.

Este proyecto se ha enfocado al diseño y construcción de una tarjeta basada en un microcontrolador al que se le denomina módulo de control, su finalidad es controlar y manipular una tarjeta de adquisición, procesamiento y síntesis de señales, además de contar con una interfaz serie para comunicarse con una computadora personal y por medio de ésta telecargar algoritmos de procesamiento digital de señales. Así también permite contar con algoritmos almacenados en memoria de estado sólido, formando de ésta manera, un sistema autónomo de procesamiento digital de señales capaz de operar sin la necesidad de utilizar en permanencia una computadora personal.

Al término del proyecto se obtuvo un sistema autónomo el cual no solo puede recibir algoritmos por telecargado a través del puerto serie, sino también, permite leer la memoria RAM tanto del módulo de control como del módulo de procesamiento digital de señales.

Por otra parte, el módulo de control también se diseñó para operar una tarjeta comercial de procesamiento digital de señales en su parte de circuitería más no así en su programación.

Así también, el módulo de control permite ser utilizado como " tarjeta madre " de propósito general para la operación de otras tarjetas de aplicación específica.

Glosario.

ASCII	American Standard Code for Information Interchange.
BF	Block Fill (Llenar Bloque).
CPU	Central Processor Unit (Unidad central de procesamiento).
DMA	Direct Memory Access (Acceso Directo a Memoria).
DSP32	Digital Signal Processor 32.
EMR	Error Mask Register. (Registro de Máscara de Error).
EPROM.	Electrically Programmable Read Only Memory. (Memoria de Sólo Lectura Electricamente Programable)
ESR	Error Source Register. (Registro de Fuentes de Error).
IBM	International Business Machines.
ISA	Industry Standard Arquitecture (Arquitectura estandar de la industria).
MFLOPS	Millones de instrucciones de punto flotante por segundo.
MIPS	Millones de Instrucciones Por Segundo.
PAR	PIO Address Register. (Registro de direcciones del PIO).
PC	Personal Computer (Computadora Personal).
PCR	PIO Control Register. (Registro de control del PIO).
PDR	PIO Data Register. (Registro de datos del PIO).
PIR	PIO Interrupt Register. (Registro de interrupciones del PIO).
PDS	Procesamiento Digital de Señales.
PIO	Parallel Input Output. (Puerto Paralelo de entrada salida).
RAM	Random Access Memory (Memoria de acceso aleatorio).
TAPS	Tarjeta de Adquisición, Procesamiento y Síntesis de señales.
UART	Universal Asynchrounous Receiver Transmitter (Receptor y transmisor asincrono universal).

Bibliografía.

INTEL, Embedded Microcontrollers and Processors, 1992.

PHILIPS, 8049/8051 Family Microcontroller User's Guide, 1988.

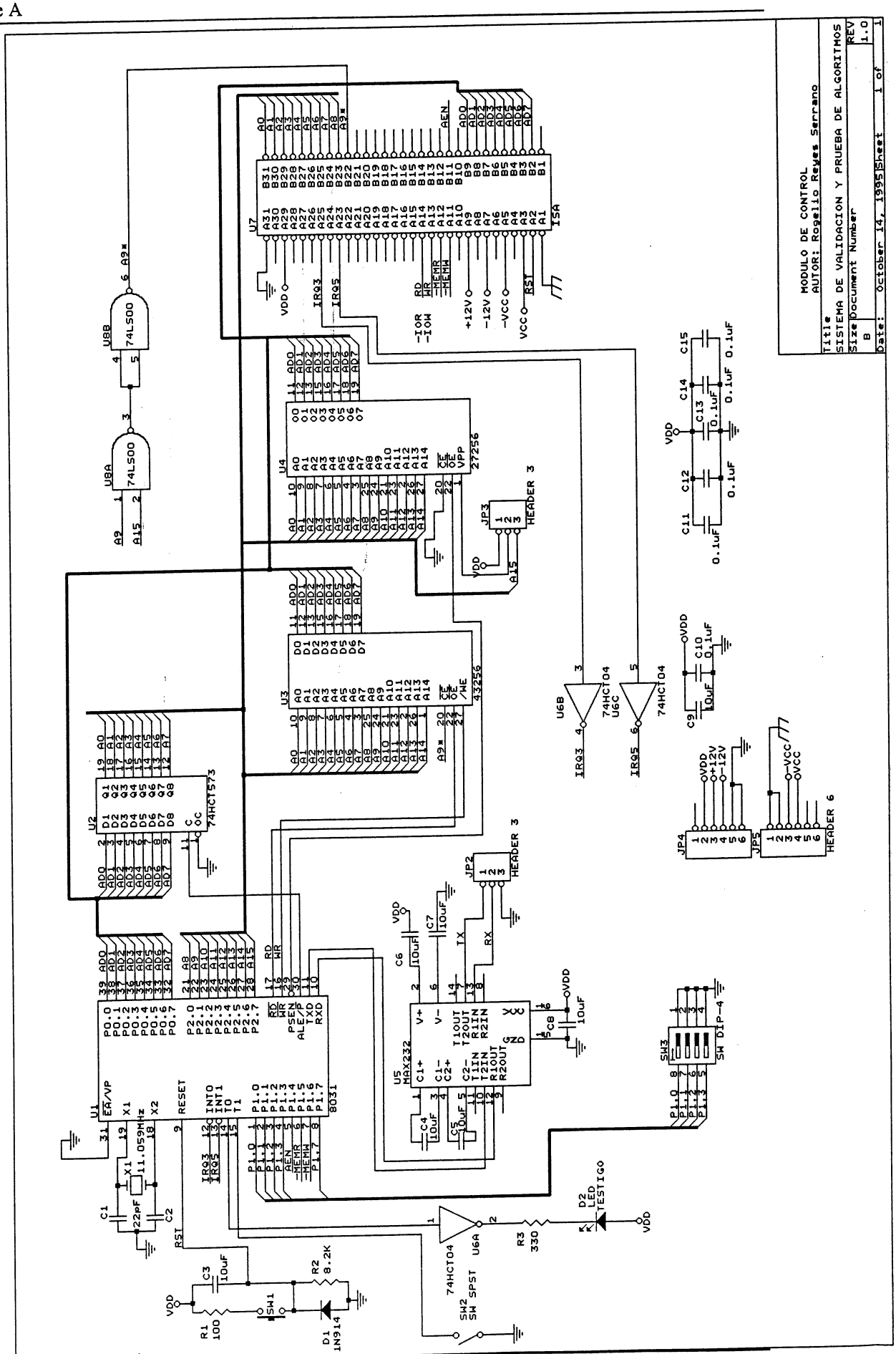
DSP32 Digital signal processor, AT&T Co.,1986.

Hitachi, IC Memory Data Book, 1991.

MAXIM, New Releases Data Book, 1993.

TOSHIBA, Logic TC74HC Series, 1988.

O'Brien Stephen, Turbo Pascal 6 McGrawHill, 1992.


MODULO DE CONTROL
AUTOR: Rogelio Reyes Serrano
Title
SISTEMA DE VALIDACION Y PRUEBA DE ALGORITMOS
Size
B
Document Number
REV
1.0
Date: October 14, 1995
Sheet 1 of 1
U1
8031
U2
74HCT573
U3
43256
U4
27256
U5
MAX232
U6
74HCT04
U7
ISA
U8
74LS00
SW1
SW2
SW SPST
SW3
SW DIP-4
JP1
JP2
JP3
JP4
JP5
HEADER 3
HEADER 6
X1
11.059MHz
D1
1N914
D2
LED
TESTIGO
R1
100
R2
8.2K
R3
330
C1
22pF
C2
C3
10uF
C4
10uF
C5
10uF
C6
10uF
C7
10uF
C8
10uF
C9
10uF
C10
0.1uF
C11
0.1uF
C12
0.1uF
C13
0.1uF
C14
0.1uF
C15
0.1uF
VDD
VCC
+12V
-12V
RST
TX
RX
IRQ3
IRQ5
-IOR
-IOW
MEMR
MEMW
AEN
ALE/P
PSEN
RD
WR
TXD
RXD

```
;*****************************************************************
; VER: 1.57                                                      *
; ULTIMA MODIF:28-FEBRERO-95                                     *
;                                                                *
; FECHA: 17-MAYO-1993                                            *
; AUTOR : ROGELIO REYES SERRANO                                  *
;                                                                *
; PROGRAMA DE CONTROL.                                           *
; TITULO DE TESIS DE LICENCIATURA:                               *
;   SISTEMA AUTONOMO DE VALIDACION Y PRUEBA DE ALGORITMOS DE     *
;              PROCESAMIENTO DIGITAL DE SENALES                  *
;*****************************************************************
;
; ************ EQUIVALENCIAS *****************
;
INICIO          EQU     0000H
MONITOR         EQU     0100H
PILA_INICIAL    EQU     6FH     ;Dirección inicial del apuntador datos
ALMACENA_DATOS  EQU     0000H   ;Dirección de la memoria RAM externa
CR              EQU     0DH     ; retorno de carro
TABULADOR       EQU     09H     ; tabulador
LF              EQU     0AH     ; line feed
ESC             EQU     1BH     ; ESCAPE
ESPACIO         EQU     20H
REL             EQU     20H     ;Modo 2 del reloj
FRE             EQU     0FDH    ;frecuencia de operación de 9600 Baud.
SERIE           EQU     52H     ;Modo 1 del puerto serie.
R0_BNK0         EQU     00H     ;Dirección del registro 0 en el banco 0
R1_BNK0         EQU     01H     ;Dirección del registro 1 en el banco 0
TESTIGO         BIT     P3.4    ;BIT asignado para el testigo
AEN             BIT     P1.4    ;
PC_o_MANUAL     BIT     P3.5    ;BIT asignado para determinar el modo de
                                ;operación manual o por PC
COMPA           EQU     0FH
GUARDA          EQU     50H
BELL            EQU     07H
SI              EQU     53H
NO              EQU     4EH
CORRIO          EQU     43H
PARAR           EQU     50H
TELE_CARGADO    EQU     54H
EPROM           EQU     45H
LEER_RAM        EQU     52H
LEER_DSP        EQU     44H
LLENA_RAM       EQU     42H
REGRESO         EQU     52H
CERO            EQU     00H
UNO             EQU     01H
DOS             EQU     02H
TRES            EQU     03H
CUATRO          EQU     04H
SEIS            EQU     06H
MUELLE          EQU     30H
SEGMENTO        EQU     31H
```

```
MUELLE_5            EQU     35H
FIN_TELE            EQU     40H         ;CODIGO DE @ EN HEX
MAYUSCULAS          EQU     31H         ;Cantidad a restar a las mayusculas.
MINUSCULAS          EQU     51H         ; "     "   " " MINUSCULAS
NUMEROS             EQU     30H         ; "     "   " " NUMEROS
NUMERO              EQU     40H         ;
MAYUSCULA           EQU     47H         ;
PCR                 EQU     8307H       ;Dirección del registro PCR en el DSP
CONFIG_PCR          EQU     18H         ;Palabra de configuracion para PCR
PAR_L               EQU     8300H       ;Dirección del registro PAR_L en DSP
PAR_H               EQU     8301H           ;Dirección del registro PAR_H en DSP
PDR_L               EQU     8302H           ;Dirección del registro PDR_L en DSP
PDR_H               EQU     8303H           ;Dirección del registro PDR_H en DSP
CORRE_LIBRE         EQU     01H             ;Palabra de control para correr DSP
PARA_DSP            EQU     00H             ;Palabra de control para para al DSP
;
;*** POSICIONES DEL SELECTOR DE ALGORITMOS ****
;
POSICION            EQU     0FH
POSICION_0          EQU     00H
POSICION_1          EQU     01H
POSICION_2          EQU     02H
POSICION_3          EQU     03H
POSICION_4          EQU     04H
POSICION_5          EQU     05H
POSICION_6          EQU     06H
POSICION_7          EQU     07H
POSICION_8          EQU     08H
POSICION_9          EQU     09H
POSICION_A          EQU     0AH
POSICION_B          EQU     0BH
POSICION_C          EQU     0CH
POSICION_D          EQU     0DH
POSICION_E          EQU     0EH
POSICION_F          EQU     0FH
;
;******** REGISTRO PCR DEL DSP ***********
;       7   6   5   4   3   2   1   0  BIT'S
;       |   |   |   |   |   |   |   |  RESET 0 = HALT, 1 = RUN
;       |   |   |   |   |   |   |
;       |   |   |   |   |   |   |-----   INTMODE 0= 8 BIT, 1= 16 BIT
;       |   |   |   |   |   |
;       |   |   |   |   |   |----------  ENI 0= Disable, 1 enable PIR
;       |   |   |   |   |
;       |   |   |   |   |---------------  DMA 0= disable, 1 enable ON PIO
;       |   |   |   |
;       |   |   |   |--------------------  AUTO 1= auto incremented DMA
;       |   |   |                          0= NO auto incremented on DMA
;       |   |   |-------------------------  PDF (PDR status) Set 1= PDR written
;       |   |                              DSP or uP 0= PDR READ DSP or uP
;       |   |-----------------------------  PIF (PIR status) SET 1=PIR WR by
;       |                                  DSP, 0= PIR RD by uP
;       |-----------------------------------  REF 0= enable auto RAM refresh
;                                              1= disable auto RAM refresh
```

```
;
;*****************************************
; MACRO INSTRUCCION
;
;*****************************************
$INCLUDE(MACROST3.A51)                    ;incluye la macro intrucción que se
                                          ;encuentra en MACROST3.A51 que
                                          ;corresponde al testigo.

;$GENONLY
;$INCLUDE(MACROST2.A51)
;$GENONLY
;
;*****************************************
;
;              PROGRAMA PRINCIPAL
;
;*****************************************
                ORG     INICIO            ;Dirección en 0000H
;*****************************************
ARRANQUE_FRIO:
                LJMP  ARRANQUE_TIBIO      ;Salto largo a arranque_tibio
                                          ;para brincar las interrupciones
                                          ;que se encuentran en la parte baja
                                          ;de la memoria.

;*****************************************
;
                ORG     MONITOR           ;Dirección en 0100H
ARRANQUE_TIBIO:
                MOV     SP,#PILA_INICIAL  ;coloca el stack pointer en la
                                          ;direccion indicada.
                CLR     AEN               ;Pone en bajo el puerto 1.4 que
                                          ;corresponde al AEN
                %PARPADEA_TESTIGO         ;hace un llamado a la macro instrucción
                                          ;parpadea testigo.
                LCALL   CONFIGURA_COM_SERIE
                CLR     C
                MOV     C,P3.5            ;Lee el bit del puerto si es uno
                MOV     GUARDA,C          ;Hace el procedimiento manual si no
                JC      MANUAL            ;Se toma el control por medio de la
                CLR     C                 ;terminal. El resultado obtenido se
                LCALL   IDENTIFICACION    ;Almacena en la localidad 50H
                LCALL   CONTROL

MANUAL:         CLR     C
                MOV     A,P1              ;Lee el puerto 1
                ANL     A,#POSICION       ;AND CON 0FH
NO_ES:          CJNE    A,#POSICION_0,NO_ES_0
                LCALL   ALGO_0
NO_ES_0:        CJNE    A,#POSICION_1,NO_ES_1
                LCALL   ALGO_1
NO_ES_1:        CJNE    A,#POSICION_2,NO_ES_2
                LCALL   ALGO_2
NO_ES_2:        CJNE    A,#POSICION_3,NO_ES_3
                LCALL   ALGO_3
```

```
NO_ES_3:    CJNE    A,#POSICION_4,NO_ES_4
            LCALL   ALGO_4
NO_ES_4:    CJNE    A,#POSICION_5,NO_ES_5
            LCALL   ALGO_5
NO_ES_5:    CJNE    A,#POSICION_6,NO_ES_6
            LCALL   ALGO_6
NO_ES_6:    CJNE    A,#POSICION_7,NO_ES_7
            LCALL   ALGO_7
NO_ES_7:    CJNE    A,#POSICION_8,NO_ES_8
            LCALL   ALGO_8
NO_ES_8:    CJNE    A,#POSICION_9,NO_ES_9
            LCALL   ALGO_9
NO_ES_9:    CJNE    A,#POSICION_A,NO_ES_A
            LCALL   ALGO_A
NO_ES_A:    CJNE    A,#POSICION_B,NO_ES_B
            LCALL   ALGO_B
NO_ES_B:    CJNE    A,#POSICION_C,NO_ES_C
            LCALL   ALGO_C
NO_ES_C:    CJNE    A,#POSICION_D,NO_ES_D
            LCALL   ALGO_D
NO_ES_D:    CJNE    A,#POSICION_E,NO_ES_E
            LCALL   ALGO_E
NO_ES_E:    LCALL   ALGO_F
  CORRE:    LCALL   A_CORRER        ;Llama subrutina para correr DSP
            LJMP    FINAL
;
;*****************************************
;
;                   SUBRUTINAS
;
;*****************************************
;
;*****************************************
;   CONFIGURACION DE LA COMUNICACION SERIE
;*****************************************
; Inicialización de la comunicación serie a 8 bits a 9600 Bauds con un
; cristal de 11.0592 MHz. En la comunicacion serie se envian 10 bits 1 de
; inicio (0), 8 de datos y 1 de paro (1)
;
    CONFIGURA_COM_SERIE:
            MOV     SCON,#SERIE     ;Modo 1, palabra de 8 bits
            SETB    TR1             ;Habilita temporizador 1
            MOV     TMOD,#REL       ;temporizador de corrida libre modo 2
            MOV     TH1,#FRE        ;frecuencia de opereración 9600 Baud.
            RET
;
;*****************************************
; SUBRUTINA DE IDENTIFICACION
;*****************************************
;   Subrutina para identificar al sistema y al autor.
;
    IDENTIFICACION:
            LCALL   XCADENA
            DB      CR,LF
```

```
                DB '*****************************************************************'
                DB      CR,LF
                DB '*  SISTEMA AUTONOMO DE VALIDACION Y PRUEBA DE ALGORITMOS *'
                DB      CR,LF
                DB '*     DE PROCESAMIENTO DIGITAL DE SENALES                 *'
                DB      CR,LF
                DB '*   AUTOR:  ROGELIO REYES SERRANO                         *'
                DB      CR,LF
                DB '*     VER 1.0                   DIC,93                    *'
                DB      CR,LF
                DB '*****************************************************************'
                DB      CR,LF,LF,ESC
                RET
;
;*****************************************
; SUBRUTINA TELE_CARGO
;*****************************************
TELE_CARGO:
                LCALL   TELECARGADO
                LCALL   CARGA_AL_DSP
                LCALL   A_CORRER
                LJMP    FINAL
;
;*****************************************;
SUBRUTINA DE TELECARGADO
;*****************************************;
    TELECARGADO:
                LCALL   XCADENA             ;Llama subrutina para enviar mensaje
                                            ;de telecargado
                DB      CR,LF
                                            ;mensaje a enviar
                DB      ' ***  INICIO DE TELECARGADO  *** '
                DB      CR,LF
                DB      '   LISTO PARA RECIBIR ALGORITMO ..... '
                DB      CR,LF
                DB      ESC                 ;Fin de mensaje de telecargado,
                                            ;transmisión y recepción.
;
                MOV     DPTR,#ALMACENA_DATOS
        MAIN:   LCALL   RECV                ;LLama subrutina para recibir
                                            ;caracter ascii
                CJNE    A,#FIN_TELE,SIGUE   ;Detecta fin del algoritmo.
                SJMP    FIN
        SIGUE:  CJNE    A,#TABULADOR,SIGUE1 ;Elimina caracter del tabulador
                SJMP    MAIN
       SIGUE1:  CJNE    A,#CR,SIGUE2        ;Elimina el caracter de retorno de carro
                SJMP    MAIN
       SIGUE2:  CJNE    A,#LF,SIGUE3        ;Elimina el caracter de cambio de línea
                SJMP    MAIN
       SIGUE3:  CJNE    A,#ESPACIO,SIGUE4
                SJMP    MAIN
       SIGUE4:  LCALL   CONVERSION          ;Llamado a subrutina de conversión de
                                            ;ascii a hex.
                SWAP    A                   ;Rotación del byte
```

```
                    MOV     R1,A
                    LCALL   RECV
                    LCALL   CONVERSION
                    ORL     A,R1            ;hace una OR entre el acumulador y
                                            ;registro #1
                    LCALL   ALMACEN
                    SJMP    MAIN            ;Salto corto a MAIN, regresa a recepción
            FIN:
                    RET
;
;*****************************************
;  SUBRUTINA DE CONTROL
;*****************************************
;
          CONTROL:
                    LCALL   XCADENA
                    DB      CR,LF
                    DB      '********  ESCOGE TU OPCION :     ******** '
                    DB      CR,LF
                    DB      '       CORRER ALGORITMO..... (C)'
                    DB      CR,LF
                    DB      '       DETENER ALGORITMO.....(P)'
                    DB      CR,LF
                    DB      '       TELECARGADO.....(T)'
                    DB      CR,LF
                    DB      '       ALGORITMOS EN EPROM .(E)'
                    DB      CR,LF
                    DB      '       LEER RAM .....(R)'
                    DB      CR,LF
                    DB      '       LEER DSP .....(D)'
                    DB      CR,LF
                    DB      '       LLENAR RAM (BF).....(B)'
                    DB      CR,LF,LF
                    DB      '       TU OPCION ES -----> '
                    DB      ESC
;
                    LCALL   RECV
                    LCALL   SEND
                    CJNE    A,#CORRIO,PARO
                    LCALL   A_CORRER
                    LJMP    CONTROL
          PARO:     CJNE    A,#PARAR,TELE
                    LCALL   STOP
                    LJMP    CONTROL
          TELE:     CJNE    A,#TELE_CARGADO,EPR
                    LJMP    TELE_CARGO
           EPR:     CJNE    A,#EPROM,LEER_LA_RAM
                    LCALL   DAME_No_DE_ALGORITMO
    LEER_LA_RAM:
                    CJNE    A,#LEER_RAM,LEER_RAM_DSP
                    LCALL   LEER_RAM_EXTERNA
                    LJMP    CONTROL
    LEER_RAM_DSP:
                    CJNE    A,#LEER_DSP,BLOCK
```

```
                LCALL   LEER_RAM_DEL_DSP
BLOCK:          CJNE    A,#LLENA_RAM,ERROR
                LCALL   BLOCK_FILL
                LJMP    CONTROL
ERROR:
                MOV     A,#BELL
                LCALL   SEND
                LCALL   XCADENA
                DB      CR,LF
                DB      ' OPCION ERRONEA '
                DB      CR,LF,ESC
                LJMP    CONTROL
                RET
;
;*********************************************
;  SUBRUTINA  DAME ALGORITMO
;*********************************************
;
;
        DAME_No_DE_ALGORITMO:
        ADRIANA:
                LCALL   XCADENA
                DB      CR,LF
                DB      '  **** ESCOGE EL ALGORITMO DESEADO ****  '
                DB      CR,LF
                DB      '                          GRABA .....(0)'
                DB      CR,LF
                DB      '                          ECO .....(1)'
                DB      CR,LF
                DB      '                          ECO2 .....(2)'
                DB      CR,LF
                DB      '                          ECO3 .....(3)'
                DB      CR,LF
                DB      '                          METAL .....(4)'
                DB      CR,LF
                DB      '                          NORMAL .....(5)'
                DB      CR,LF
                DB      '                          RARO .....(6)'
                DB      CR,LF
                DB      '                          REVERB .....(7)'
                DB      CR,LF
                DB      '                          SALON .....(8)'
                DB      CR,LF
                DB      '                          ROBOT .....(9)'
                DB      CR,LF
                DB      '                          BAJOS .....(A)'
                DB      CR,LF
                DB      '                          DISPONIBLE.....(B)'
                DB      CR,LF
                DB      '                          DISPONIBLE.....(C)'
                DB      CR,LF
                DB      '                          DISPONIBLE.....(D)'
                DB      CR,LF
                DB      '                          DISPONIBLE.....(E)'
                DB      CR,LF
```

```
                DB      '                                DISPONIBLE.....(F)'
                DB      CR,LF
                DB      ' REGRESAR AL MENU PRINCIPAL .....(R)'
                DB      CR,LF
                DB      '          TU OPCION ES ---> : '
                DB      ESC
;
   ME_EQUIVOQUE:
                LCALL RECV
                MOV     R3,A
                LCALL   SEND
                LCALL   XCADENA
                DB      CR,LF
                DB      '  ES CORRECTA (S/N) '
                DB      CR,LF,ESC
                LCALL   RECV
                CJNE    A,#SI,NEL
                MOV     A,R3
                CJNE    A,#REGRESO,ZARINA
                SJMP    HASTA_AQUI
        ZARINA: LCALL   CONVERSION
                CJNE    A,#COMPA,ESTA_BIEN
                JC      ESTA_BIEN
                SJMP    CAMPANA
     ESTA_BIEN:
                CJNE    A,#REGRESO,PROSIGO
                SJMP    HASTA_AQUI
       PROSIGO:
                LJMP    NO_ES
           NEL:
                CJNE    A,#NO,CAMPANA
                LJMP    ADRIANA
       CAMPANA:
                MOV     A,#BELL
                LCALL   SEND
                LJMP    ME_EQUIVOQUE
   HASTA_AQUI:
                RET
;
;*******************************************
;   Subrutina para recepción.
;*******************************************
;
                                        ;Inicia rutina de recepción. RECV:
                JNB     RI,RECV         ;Espera un caracter.
                CLR     RI              ;Limpia bandera de recepción
                MOV     A,SBUF          ;Mueve caracter que se encuentra en
                                        ;el buffer del puerto serie
                                        ;al acumulador.
                RET                     ;fin de rutina
;
;*******************************************
;   Subrutina de transmisión.
;*******************************************
;
```

```
SEND:   JNB     TI,SEND     ;Espera hasta que la transmisión
                            ;este lista.
        CLR     TI          ;Limpia bandera de transmisión.
        MOV     SBUF,A      ;Mueve lo que hay en el acumulador
                            ;al buffer del puerto serie
        RET                 ;Fin de la rutina
;
;*********************************************
;  SUBRUTINA LLENA RAM
;*********************************************
;
;  Subrutina para llenar la RAM con datos deseados dadas las direcciones
;  inicial y final.
;
    BLOCK_FILL:
;
        LCALL   DE_DONDE_A_DONDE
        LCALL   XCADENA
        DB      CR,LF
        DB      ' CON QUE LO LLENO (XX): '
        DB      ESC
        LCALL   RECV
        LCALL   SEND
        LCALL   CONVERSION
        SWAP    A
        MOV     R3,A
        LCALL   RECV
        LCALL   SEND
        LCALL   CONVERSION
        ORL     A,R3
        MOV     DPL,86H
        MOV     DPH,85H
        INC     DPTR
        MOV     R3,A
CHAPIS: MOV     A,R3
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,R7
        CJNE    A,DPH,CHAPIS
        MOV     A,R6
        CJNE    A,DPL,CHAPIS
        RET
;
;*********************************************
;  SUBRUTINA DE CONVERSION DE ASCII A HEXADECIMAL
;*********************************************
;  Esta subrutina hace la conversión de ascii a hexadecimal del 0..9 y de
;  A,B,C,D,E,F y a,b,c,d,e,f.
;
;
CONVERSION: MOV     R0,A
        CJNE    R0,#30H,B1
        MOV     A,#00H
B1:     CJNE    R0,#31H,B2
```

```
                MOV     A,#01H
B2:             CJNE    R0,#32H,B3
                MOV     A,#02H
B3:             CJNE    R0,#33H,B4
                MOV     A,#03H
B4:             CJNE    R0,#34H,B5
                MOV     A,#04H
B5:             CJNE    R0,#35H,B6
                MOV     A,#05H
B6:             CJNE    R0,#36H,B7
                MOV     A,#06H
B7:             CJNE    R0,#37H,B8
                MOV     A,#07H
B8:             CJNE    R0,#38H,B9
                MOV     A,#08H
B9:             CJNE    R0,#39H,BA
                MOV     A,#09H
BA:             CJNE    R0,#41H,BB
                MOV     A,#0AH
BB:             CJNE    R0,#42H,BC
                MOV     A,#0BH
BC:             CJNE    R0,#43H,BD
                MOV     A,#0CH
BD:             CJNE    R0,#44H,BE
                MOV     A,#0DH
BE:             CJNE    R0,#45H,BF
                MOV     A,#0EH
BF:             CJNE    R0,#46H,BAa
                MOV     A,#0FH
BAa:            CJNE    R0,#61H,BBb
                MOV     A,#0AH
BBb:            CJNE    R0,#62H,BCc
                MOV     A,#0BH
BCc:            CJNE    R0,#63H,BDd
                MOV     A,#0CH
BDd:            CJNE    R0,#64H,BEe
                MOV     A,#0DH
BEe:            CJNE    R0,#65H,BFf
                MOV     A,#0EH
BFf:            CJNE    R0,#66H,B0
                MOV     A,#0FH
B0:             RET
;
;*******************************************
;     SUBRUTINA DE ALMACENAMIENTO DE INFORMACION
;*******************************************
; Subrutina para almacenar la información en la memoria RAM externa
; desde las direcciones  0000H hasta la 8000H  que son 32K
;
ALMACEN:
                MOVX    @DPTR,A         ;Mueve el acumulador a una dirección
                                        ;externa  ( dirección de la RAM ) INC
                        DPTR            ;Incrementa el DPTR
                CLR     A
```

```
                    RET
;
;*********************************************
; SUBRUTINA PARA CARGAR LOS ALGORITMOS AL DSP
;*********************************************
;     Subrutina para cargar los algoritmos al DSP por medio de comunicación
;     en paralelo. Primero manda direcciones ( 2 octetos ) y luego datos
;     ( 4 octetos ).
;
          CARGA_AL_DSP:
                    LCALL     CONFIGURA
                    MOV       DPTR,#ALMACENA_DATOS
                    MOV       R2,#CERO
     REPITE:        MOV       R0,#MUELLE          ;Cargo R0 con una dirección de RAM (30H)
                    MOV       R1,#SEIS            ;Cargo R1 con dato.
     OTRO_DATO:     MOVX      A,@DPTR             ;Cargo el acumulador con dato que apunta
                                                   ;el DPTR en RAM externa.
                    MOV       @R0,A               ;Lo contenido en acumulador lo guardo en
                                                   ;RAM interna que es apuntada por R0. INC DPTR
                    INC       R0
                    DJNZ      R1,OTRO_DATO        ;Decremento Registro #1, si no es
                                                   ;cero salto.
                    MOV       R4,DPL
                    MOV       R5,DPH
                    CJNE      R2,#CERO,FIN_ARCHIVO
          DE_NUEZ:
                    LCALL     DSP_EPROM
                    MOV       R2,#UNO
                    MOV       DPH,R5
                    MOV       DPL,R4
                    SJMP      REPITE
     FIN_ARCHIVO:
                    MOV       R0,#MUELLE
                    CJNE      @R0,#CERO,DE_NUEZ
                    INC       R0
                    CJNE      @R0,#CERO,DE_NUEZ
                    RET
;
;*********************************************
;     SUBRUTINA  DE_DONDE_A_DONDE
;*********************************************
;
;     Subrutina para capturar las direcciones inicial y final del
;     intervalo que se desea leer de la RAM.
;
     DE_DONDE_A_DONDE:
                    MOV       R5,#UNO
                    LCALL     XCADENA
                    DB        CR,LF
                    DB  '     DAME DIRECCION INICIAL (XXXX) :'
                    DB  ESC
     RECIBE:        LCALL     RECV
                    LCALL     SEND
                    LCALL     CONVERSION
```

```
                SWAP    A
                MOV     R6,A
                LCALL   RECV
                LCALL   SEND
                LCALL   CONVERSION
                ORL     A,R6
                CJNE    R5,#UNO,BAJA
                MOV     85H,A
                MOV     R5,#DOS
                SJMP    RECIBE
        BAJA:
                MOV     86H,A
                MOV     R5,#UNO
                LCALL   XCADENA
                DB      CR,LF
                DB '    DAME DIRECCION FINAL (XXXX) :'
                DB ESC
    RECIBES:    LCALL   RECV
                LCALL   SEND
                LCALL   CONVERSION
                SWAP    A
                MOV     R6,A
                LCALL   RECV
                LCALL   SEND
                LCALL   CONVERSION
                ORL     A,R6
                CJNE    R5,#UNO,BAJAS
                MOV     R7,A
                MOV     R5,#DOS
                SJMP    RECIBES
        BAJAS:
                MOV     R6,A
                LCALL   XCADENA
                DB      CR,LF,ESC
                RET
;
;************************************************
;   SUBRUTINA LEER_RAM_EXTERNA
;************************************************
;   Subrutina para leer ram externa dada una dirección inicial
;   y una final
;
            LEER_RAM_EXTERNA:
;
                LCALL   DE_DONDE_A_DONDE
                MOV     R3,#SEIS
                MOV     DPL,86H
                MOV     DPH,85H
                INC     DPTR
            KK3:MOV     R1,#DOS
                MOVX    A,@DPTR
                MOV     R4,A
                SWAP    A
            KK1:ANL     A,#0FH
```

```
                    LCALL   HEX_ASCII
                    LCALL   SEND
                    MOV     A,R4
                    DJNZ    R1,KK1
                    INC     DPTR
                    MOV     A,R7
                    CJNE    A,DPH,KK3
                    MOV     A,R6
                    CJNE    A,DPL,KK3
                    SJMP    AZUL
          KK2:      DJNZ    R3,KK2
                    MOV     R3,#SEIS
                    PUSH    DPL
                    PUSH    DPH
                    LCALL   XCADENA
                    DB      CR,LF,ESC
                    POP     DPH
                    POP     DPL
                    LJMP    KK2
          AZUL:
                    RET
;
;*********************************************
; SUBRUTINA  HEX_ASCII
;*********************************************
;
;     Subrutina para hacer la conversión de hexadecimal a ascii.
;
       HEX_ASCII:
                    CJNE    A,#0AH,NUM
            NUM:    JC      NUME
                    ADD     A,#31H
                    SJMP    Z
            NUME:
                    ADD     A,#30H
              Z:    DA      A
                    RET
;
;*********************************************
;     SUBRUTINA LEER RAM DEL DSP
;*********************************************
;
;
            LEER_RAM_DEL_DSP:
                    LCALL   CONFIGURA
                    LCALL   DE_DONDE_A_DONDE
;
                    MOV     DPTR,#PAR_L
                    MOV     A,86H
                    INC     A
                    MOVX    @DPTR,A
                    INC     DPTR
                    MOV     A,85H
                    MOVX    @DPTR,A
;
```

```
                MOV     R5,#CERO
                MOV     R1,#DOS
                MOV     R0,#MUELLE
AMARILLO:       MOV     DPTR,#PDR_L
                MOVX    A,@DPTR
                MOV     @R0,A
                INC     DPTR
                INC     R0
                MOVX    A,@DPTR
                MOV     @R0,A
                INC     R0
                DJNZ    R1,AMARILLO
                MOV     DPL,86H
                MOV     DPH,85H
                MOV     R3,#CUATRO
                MOV     R0,#33H
AA2:            MOV     R1,#DOS
                MOV     A,@R0
                MOV     R4,A
                SWAP    A
AA1:            ANL     A,#0FH
                LCALL   HEX_ASCII
                LCALL   SEND
                INC     DPTR
                MOV     A,R7
                CJNE    A,DPH,PAPA
                SJMP    DPLB
PAPA:           INC     A
                CJNE    A,DPH,PEPA
DPLB:           MOV     A,R6
                CJNE    A,DPL,PAPI
PAPI:           INC     A
                CJNE    A,DPL,PEPA
                SJMP    VERDE
PEPA:           MOV     A,R4
                DJNZ    R1,AA1
                DEC     R0
                DJNZ    R3,AA2
                MOV     86H,DPL
                MOV     85H,DPH
                LCALL   XCADENA
                DB      CR,LF,ESC
                LJMP    AMARILLITO
VERDE:
                MOV     DPTR,#PCR
                MOV     A,#PARA_DSP
                MOVX    @DPTR,A
                RET
;
;*****************************************
;
;  SUBRUTINA DE  RETARDO_LARGO
;*****************************************
;
;
```

```
RETARDO_LARGO:
                PUSH    PSW
                CLR     RS1             ;Limpia el bit RS1 de PSW
                CLR     RS0             ;Limpia el bit RS0 de psw
                                        ;Al limpiar estos dos bits se
                                        ;selecciona el banco de registros 0.
                PUSH    R0_BNK0         ;
                PUSH    R1_BNK0         ;
                MOV     R0,#07FH        ;Mueve el dato inmediato 7F a R0
        TEM2:   MOV     R1,#0FFH        ;"  "  "  "     FF a R1
        TEM1:   NOP
                DJNZ    R1,TEM2         ;Decrementa registro 1 y si no es
                                        ;cero salta a tem2
                DJNZ    R0,TEM1         ;decrementa R0 y si no salta a tem1
                POP     R1_BNK0
                POP     R0_BNK0
                POP     PSW
                RET
;
;*****************************************
;  SUBRUTINA CONFIGURA
;*****************************************
;   Subrutina para configurar el PDS, el registro que se configura es
;   el PCR con la palabra 18H o 00011000b
;
        CONFIGURA:
;
                MOV     DPTR,#PCR       ;
                MOV     A,#CONFIG_PCR   ;CONFIGURO EL REGISTRO
                                        ;PCR DEL DSP
                MOVX    @DPTR,A         ;
                RET
;
;*****************************************
; SUBRUTINA PARA ACTIVAR EL DSP
;*****************************************
;
; Subrutina para configurar el registro PCR del DSP32, de tal modo que  este ejecute el algoritmo
; cargado. La palabra para correrlo es 01H
;
        A_CORRER:
                MOV     R3,#TRES
                MOV     DPTR,#PCR       ;
                MOV     A,#CORRE_LIBRE  ;CONFIGURO EL REGISTRO PCR DEL
DSP
          RRS:  MOVX    @DPTR,A         ;PARA QUE ESTE EMPIEZE A
EJECUTAR
                                        ;EL ALGORITMO.
                DJNZ  R3,RRS
                LCALL   XCADENA
                DB      CR,LF
                DB      'ESTOY  CORRIENDO'
                DB      CR,LF,ESC
                RET
```

```
;
;*******************************************
; SUBRUTINA PARA PARAR AL DSP
;******************************************* ;
;       Subrutina para detener al DSP
            STOP:
                        MOV     DPTR,#PCR
                        MOV     A,#PARA_DSP
                        MOVX    @DPTR,A
                        LCALL   XCADENA
                        DB      CR,LF
                        DB      ' ESTOY SIN HACER NADA '
                        DB      CR,LF,ESC
                        RET
;
;*******************************************
; SUBRUTINA CARACTERES
;*******************************************
; Subrutina para leer un algoritmo de la memoria EPROM, la forma de  leerlo es por línea la cual consta
de 8 octetos con el siguiente  formato  DB  00H,01H,02H,03H,04H,05H
;
            CARACTERES:
;
                        POP     DPH
                        POP     DPL
                        MOV     R2,#CERO
                        MOV     R0,#MUELLE
                        MOV     R1,#SEIS
          BOLITA:       CLR     A
                        MOVC    A,@A+DPTR
                        MOV     @R0,A
                        INC     DPTR
                        INC     R0
                        DJNZ    R1,BOLITA
                        MOV     R4,DPL
                        MOV     R5,DPH
                        CJNE    R2,#CERO,FIN_ALGO
          DE_LIMON:
                        LCALL   DSP_EPROM
                        MOV     R2,#UNO
                        MOV     DPH,R5
                        MOV     DPL,R4
                        MOV     R2,#UNO
                        SJMP    DE_NUEVO
        FIN_ALGO:
                        MOV     R0,#MUELLE
                        CJNE    @R0,#CERO,DE_LIMON
                        INC     R0
                        CJNE    @R0,#CERO,DE_LIMON
                        MOV     DPH,R5
                        MOV     DPL,R4
                        MOV     A,#00
                        JMP     @A+DPTR
;
```

```
;*****************************************
;    SUBRUTINA PARA CARGAR ALGORITMOS AL DSP DE LA EPROM
;*****************************************
;
      DSP_EPROM:
                        MOV     R0,#MUELLE
                        MOV     DPTR,#PAR_L         ;Cargo DPTR con dirección del registro
                                                    ;de direcciones bajas del DSP.
                        INC     R0
                        MOV     A,@R0               ;El contenido que apunta R0 lo cargo al ACC
                        MOVX    @DPTR,A             ;Mando el contenido del acumulador a una
                                                    ;dirección externa que es apuntada por DPTR.
                        INC     DPTR
                        DEC     R0
                        MOV     A,@R0               ;R0=31
                        MOVX    @DPTR,A             ;
                        MOV     R0,#MUELLE_5        ;Cargo R0 con dirección de ram 35h
      OTRO_OK1:         MOV     DPTR,#PDR_L         ;cargo DPTR con direccion del registro
                                                    ;PDR_L de DSP
                        MOV     A,@R0
                        MOVX    @DPTR,A             ;Escribo en registro PDR_L
                        DEC     R0
                        INC     DPTR                ;incremento DPTR para apunte a PDR_H
                        MOV     A,@R0               ;cargo al acumulador con lo que hay en
                                                    ;dirección de la RAM interna.
                        MOVX    @DPTR,A             ;Escribo en registro PDR_h del PDS
                        DEC     R0
                        CJNE    R0,#SEGMENTO,OTRO_OK  ;Si R0 es igual a dos salto a otro_ok
                        RET
;
;*****************************************
; Subrutina de desplegado de mensajes
;*****************************************
;    Subrutina para desplegar mensajes en pantalla por medio de la comunicación serie.
;
        XCADENA:                                    ;inicia rutina para desplegar
                                                    ;mensajes en pantalla
                        POP     DPH                 ;Carga DPTR con dirección del primer
                        POP     DPL                 ;caracter del mensaje que se encuentra
                                                    ;abajo del llamado a esta subrutina. CLR    A
                                                    ;Limpia Acumulador
                        MOVC    A,@A+DPTR           ;Fetch primer caracter de la cadena
            ETI:        LCALL   SEND                ;llamado a rutina de transmisión
                        INC     DPTR                ;Incrementa DPTR
                        CLR     A                   ;limpia ACC
                        MOVC    A,@A+DPTR           ;Manda el siguiente caracter a la
                                                    ;salida para transmisión
                        CJNE    A,#ESC,ETI          ;Compara el accumulador con ESC. si no
                                                    ;es igual salta a ETI, si es igual ;continua con
                                                    la siguinte instrucción.
                        MOV     A,#1                ;
                        JMP     @A+DPTR             ;salto indirecto relativo de el DPTR.
                                                    ;con lo cual termina el llamado a esta
                                                    ;subrutina.(esta instrucción es un
                                                    ;equivalente a RET)
```

```
;
;*****************************************
;
;
;     MACRO INSTRUCCIONES
;
;*****************************************
;
```

; Estas macro instruciones se utilizan para integrar los algoritmos ; al programa al momento del ensamblado.

```
ALGO_0:         LCALL   CONFIGURA
                LCALL   CARACTERES
$INCLUDE(GRABA.A51)
$GENONLY
                LJMP    CORRE
;
ALGO_1:         LCALL   CONFIGURA
                LCALL   CARACTERES
$INCLUDE(ECO.A51)
$GENONLY
                LJMP    CORRE
;
ALGO_2:         LCALL   CONFIGURA
                LCALL   CARACTERES
$INCLUDE(ECO2.A51)
$GENONLY
                LJMP    CORRE
;
ALGO_3:         LCALL   CONFIGURA
                LCALL   CARACTERES
$INCLUDE(ECO3.A51)
$GENONLY
                LJMP    CORRE
;
ALGO_4:         LCALL   CONFIGURA
                LCALL   CARACTERES
$INCLUDE(METAL.A51)
$GENONLY
                LJMP    CORRE
;
ALGO_5:         LCALL   CONFIGURA
                LCALL   CARACTERES
$INCLUDE(NORMAL.A51)
$GENONLY
                LJMP    CORRE
;
ALGO_6:         LCALL   CONFIGURA
                LCALL   CARACTERES
$INCLUDE(RARO.A51)
$GENONLY
                LJMP    CORRE
;
ALGO_7:
                LCALL   CONFIGURA
                LCALL   CARACTERES
```

```
$INCLUDE(REVERB.A51)
$GENONLY
                        LJMP    CORRE
;
ALGO_8:
                        LCALL   CONFIGURA
                        LCALL   CARACTERES
$INCLUDE(SALON.A51)
$GENONLY
                        LJMP    CORRE
;
ALGO_9:
                        LCALL   CONFIGURA
                        LCALL   CARACTERES
$INCLUDE(ROBOT.A51)
$GENONLY
                        LJMP    CORRE
;
ALGO_A:
                        LCALL   CONFIGURA
                        LCALL   CARACTERES
$INCLUDE(BAJOS.A51)
$GENONLY
                        LJMP    CORRE
;
ALGO_B:
                        LCALL   CONFIGURA
                        LCALL   CARACTERES
;$INCLUDE(DISPONIBLE.A51)        ;Quitar los ; y poner el nombre del
;$GENONLY                        ;algoritmo dentro del paréntesis.
                        LJMP    CORRE
;
ALGO_C:
                        LCALL   CONFIGURA
                        LCALL   CARACTERES
;$INCLUDE(DISPONIBLE.A51)
;$GENONLY
                        LJMP    CORRE
;
ALGO_D:
                        LCALL   CONFIGURA
                        LCALL   CARACTERES
;$INCLUDE(DISPONIBLE.A51)
;$GENONLY
                        LJMP    CORRE
;
ALGO_E:
                        LCALL   CONFIGURA
                        LCALL   CARACTERES
;$INCLUDE(DISPONIBLE.A51)
;$GENONLY
                        LJMP    CORRE
;
ALGO_F:
```

```
                    LCALL   CONFIGURA
                    LCALL   CARACTERES
;$INCLUDE(DISPONIBLE.A51)
;$GENONLY
                    LJMP    CORRE
;
          FINAL: NOP
                    MOV     C,50H
                    JC      CC
                    LJMP    CONTROL
              CC:
                    SJMP    CC
                    END             ;Fin del programa.


;VER:   1.0
;ULTIMA MODIF: 6-JULIO-93
;
;FECHA: 17-MAYO-1993
;AUTOR : ROGELIO REYES SERRANO
;***********************************************
;
; DEFINICION DE MACRO INSTRUCCIONES PARA TELECARGADO
;***********************************************
; Esta macro instrucción es para parpadear el testigo y así poder saber que el programa esta
; corriendo, el testigo parpadea despues de prender o inicializar
; el modulo de control.
%*DEFINE(PARPADEA_TESTIGO)
(
PARPADEA_TESTIGO:
                    PUSH    ACC
                    MOV     ACC,#CINCO      ;mueve el dato cinco al acumulador
          CICLO:    CLR     TESTIGO         ;limpia testigo
                    LCALL   RETARDO_LARGO   ;Llama a la subrutina
                    SETB    TESTIGO         ;pone el bit testigo en uno (P0.0)
                    LCALL   RETARDO_LARGO
                    DJNZ    ACC,CICLO       ;decrementa el acumulador y si no
                                            ;es cero salta a ciclo.
                    POP     ACC
)
```