UNIVERSIDAD AUTONOMA DE GUADALAJARA

INCORPORADA A LA UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA DE INGENIERIA EN COMPUTACION

CONTROL DE TEMPERATURA CON EL MICROPROCESADOR 6809.

TESIS PROFESIONAL

QUE PARA OBTENER EL TITULO DE

INGENIERO EN COMPUTACION

P R E S E N T A

MANUEL LOMELI PULIDO

GUADALAJARA, JAL. DICIEMBRE DE 1990

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

INDICE

INDICE DEL APENDICE

INTRODUCCION

Para la elaboración de esta tesis consideré dos situaciones como problemas a resolver, en primer lugar la falta de información sobre control digital con el microprocesador 6809; para aquellas personas que sólo conocen las características y funcionamiento de este microprocesador, o bien las personas que ya han logrado la elaboración de una microcomputadora en un sistema mínimo y desean utilizarla para una aplicación de control ésta es una tesis que soluciona ese problema, ya que consiste en el diseño de un control de temperatura con este microprocesador; en donde se utilizan todos los dispositivos del sistema mínimo como PIAS, RAM, ROM, Decodificador, Teclado, y Display.

Otro problema que consideré es el que se presenta en el consumo innecesario de corriente eléctrica, esto sucede cuando tenemos algún equipo encendido en momentos en los que no se ocupa por ejemplo fuera de los tiempos en los que un determinado proceso debe estar ejecutándose, o cuando el equipo está encendido siendo que sólo es necesario al detectar cierta variable física, que en nuestro caso es la temperatura.

Este segundo problema resultó del hecho de que en cierto lugar donde se presentara esta situación podría aplicarse este control con opciones de modificarlo según las necesidades que se requieran; ya que ésa es una de las ventajas del control, el volver a grabar la memoria con el programa adecuado, sin la necesidad de diseñar otro circuito.

La elaboración de esta tesis consiste en el diseño de un circuito controlador que determinará en que momento es necesario mandar la señal de control a un determinado sistema, el cual estará sujeto a la variable temperatura. Esta señal podría activar algún sistema de ventilación o de enfriamiento dentro de un determinado proceso.

Este controlador únicamente generará la señal, no llevará circuitos de potencia; en general consiste en lo siguiente; el usuario podrá introducir la temperatura máxima, la cual será el límite para mandar la señal de control, es decir, si la temperatura actual llega o pasa de ese valor se dará la señal de control y se mantendrá mientras no baje de esa temperatura. Durante el proceso el usuario podrá ver en Display la temperatura actual en todo momento.

La temperatura se registrará mediante un sensor de temperatura LM35D, el cual varía 10mV por cada grado centígrado en forma lineal; este voltaje se introduce a un convertidor analógico-digital, para lo cual se utilizará un arreglo de amplificadores con motivo de alimentar al convertidor con valores precisos según sus especificaciones, con el fin de que entregue una conversión correcta, y así poder introducir esta información a un puerto PIA, para que enseguida sea procesada por el microprocesador, y se ejecute la acción correcta.

Escogí la aplicación del microprocesador 6809 para el control porque es uno de los más utilizados industrialmente en la actualidad con este fin

además por la ventaja que mensione anteriormente que es la de poder modificar el control bajo otras circunstancias o características diferentes de - procesamiento con el simple hecho de modificar el programa existente y volverlo a grabar en la EPROM para que funcione adecuadamente; lo cual es algo mucho más sencillo que diseñar un nuevo circuito por cada variación en la necesidad de control.

Hay que tener en cuenta que ésta es sólo una aplicación, y no un curso del microprocesador, es decir, que aquí no se enseñará al lector desde qué es el microprocesador, o cómo se programa en lenguaje ensamblador, o el conocimiento de las instrucciones, por lo que sólo se dará una breve descripción de cada dispositivo como referencia y una detallada explicación de todo el diseño.

ANTECEDENTES.

Antes de entrar al análisis del controlador se tienen que tomar en cuenta las bases teóricas sobre el tipo específico de control que se maneja; así como las diferencias entre este tipo de control elaborado con un microprocesador y otros controladores basados en diferentes dispositivos.

Este controlador se clasifica dentro de los llamados sistemas de bucla cerrada por su habilidad para comparar el valor real de la variable controlada con el valor de referencia y automáticamente ejecutar una acción en base a esta comparación. La variable a controlar en nuestro caso es la temperatura y el valor de referencia es el máximo valor de temperatura que se introduce como punto de actuación del control.

Dentro de un sistema de bucla cerrada existe un dispositivo conocido como comparador, el cual compara el valor medido de la variable y el valor de referencia; este comparador que en nuestro caso se ubica dentro del microprocesador, genera una señal de error que se considera igual al valor medido menos el valor deseado, de modo que si el valor medido es más grande que el de referencia, la señal de error será positiva; si el valor medido es más pequeño, la señal de error será negativa.

El controlador se basa en este principio; la señal de control se generará cuando el error sea igual a cero o cuando sea positivo, es decir, que la temperatura actual es igual o mayor al valor de referencia; la misma señal -

de control se desactivará cuando el error sea negativo.

Existen 5 tipos de sistemas de bucla cerrada, este controlador se encuentra dentro de los controles todo o nada, en donde el dispositivo corrector final, o la señal de control tiene solamente dos posiciones o estados de operación que dependen de la señal de error; estas dos posiciones de la señal de control son activada o desactivada .

Todos los controladores todo o nada tienen una pequeña zona de actuación, que está definida como el más pequeño rango de valores medidos que debe atravesar para hacer que la señal de control vaya de una posición a la otra, es decir, que el valor medido debe pasar por encima del valor de referencia cierta cantidad para que la señal de control se active; de manera similar que ese valor medido pase por debajo del valor de referencia cierta cantidad para desactivar la señal de control.

La zona de actuación conocida también como histéresis, es en nuestro controlador de 1 grado, ya que para el lado positivo no es necesario que el valor medido sea mayor que el de referencia, es decir, que la señal de control se activará en el momento en que el valor medido sea igual o mayor al de referencia, pero se desactivará cuando el error sea negativo — por lo menos en uno. En este control se trabaja con números enteros únicamente.

Una vez comprendidos los conceptos sobre el control todo o nada, ana

licemos las distintas formas de como se han desarrollado este tipo de controles que dependen de la variable temperatura en función a dispositivos antecesores al microprocesador.

Podemos encontrar controles de temperatura basados en los siguientes métodos de diseño:

A) Bimetálicos

B) Transistores

C) Amplificadores operacionales

D) Sistemas digitales

E) Microprocesadores

A medida que avanzamos en esta lista podemos lograr un diseño de un controlador más eficiente, más fácil de construir por la eliminación de tantas etapas, más versatil por la facilidad de modificarlo y corregirlo, más entendible al estudio de sus conexiones; entre muchas otras, éstas son algunas de las ventajas que presenta cualquiera de estos métodos con respecto al anterior.

Los controles de temperatura con bimetal basan su funcionamiento en una espiral metálica que es posicionada en el medio donde se requiere controlar temperatura; esta espiral se construye de dos metales, con el metal de mayor coeficiente de dilatación en su interior. La espiral se desenrolla a medida que la temperatura sube logrando abrir el interruptor que conecta

algun dispositivo, de igual manera, cuando la temperatura baja, la espiral se enrolla y cierra ese interruptor. La espiral se encuentra colocada en un eje para poder rotarse y establecer la posición inicial, es decir, la temperatura deseada. En forma general es como este controlador funciona; es de importancia notar la desventaja que resulta de este controlador donde el establecimiento de todos los parámetros se realiza manualmente y no en forma digital como en el caso del microprocesador u otros métodos. Otra desventaja es que con el paso del tiempo estos metales pierden propiedades logrando una disminución en la presición y estabilidad del controlador hasta el punto de volverse ineficiente.

Los controladores basados en transistores presentan varias ventajas sobre los bimetálicos; podemos obtener una respuesta más óptima y estable ya que poseen una vida más prolongada por su composición en semiconductores, idealmente infinita, esta respuesta también resulta más rapida y versatil debido a que se pueden hacer arreglos con circuitos transistorisados como arreglos lógicos para transferir esa señal a otros dispositivos o etapas de control; también se pueden diseñar cadenas de amplificación para aumentar los niveles de señal y poder monitorearla con aparatos como el osciloscopio obteniendo su tipo de respuesta; otra ventaja es que tienen una compensación más alta a la temperatura. Con una secuencia estructurada se puede lograr la elaboración de un controlador sencillo, ya que para un controlador complicado donde se utilice intercambio de información para el manejo de muchos datos resulta casi imposible diseñarlo por la gran cantidad de arreglos que se deberían construir, haciéndolo en cierto modo ineficiente.

Un controlador diseñado con amplificadores operacionales presenta varias ventajas sobre los arreglos con transistores; se pueden eliminar gran cantidad de etapas lógicas y de amplificación; se pueden utilizar para diseñar arreglos matemáticos como los logarítmicos cuya principal característica es linealizar las curvas de respuesta para hacerla más rápida; se obtiene también un mejor establecimiento en niveles de voltaje, mayor ganancia del sistema, mayor facilidad para estabilizar y compensar este tipo de circuitos. En general se puede construir un mejor controlador que presente más ventajas para el manejo de información.

La utilización de sistemas digitales es de gran eficiencia para la elaboración de controladores ya que es superior a los anteriores métodos debido a la eliminación de tantas etapas de amplificación; son sistemas más rápidos, más estables, presentan una gran ventaja que es la adquisición de datos por medio de registros, poseen mayor facilidad para el manejo de información, se pueden manejar más fácilmente operaciones lógicas y matemáticas; por éstas y otras ventajas se pueden diseñar controladores más complicados y precisos.

Hasta ahora hemos analizado como se han elaborado controladores en diferentes métodos de diseño, observando la superioridad de cada uno con respecto al anterior, con todo ello el microprocesador es todavía más eficiente a todos los anteriores por la gran cantidad de información que puede manejar y por la forma en que puede ser modificada cualquier aplicación en la necesidad de control.

CAPITULO I

DISEÑO DE LOS CIRCUITOS CONTROLADOR Y SENSOR-CONVERTIDOR.

1.1. DISEÑO Y DESCRIPCION DEL CIRCUITO CONTROLADOR.

Haremos el análisis del circuito basándonos en el dibujo que se muestra en la figura 1.3. empecemos desde el microprocesador hacia los demás dispositivos. Las tres interrupciones, $\overline{NMI}$, interrupción incondicionada, $\overline{IRQ}$, interrupción condicionada, y $\overline{FIRQ}$, interrupción rápida condicionada, no se utilizan porque no se recibe ninguna señal externa que requiera su uso, por lo que fueron desactivadas conectándolas a 5V. El bus de direcciones del microprocesador A0 - A15 es conectado a todos los dispositivos línea por línea hasta la capacidad de direccionamiento de cada uno; en la memoria RAM 6116 - tenemos de A0 hasta A10, logrando direccionar 2K de memoria; en la 2732 tenemos de A0 hasta A11 para direccionar 4K de memoria ROM; en los dos PIAS 6821 se conecta solamente A0 y A1 en RS0 y RS1 que son los registros de selección del dispositivo, siendo así 4 las únicas localidades para direccionar, 2 corresponden a los registros de direcciones del puerto A y B, y las otras 2 a los registros de datos A y B. Observamos enseguida que en el codificador entran A13, A14, y A15, con esto estamos introduciendo el código de codificación para poder direccionar al dispositivo que sea el indicado; sabemos que el dispositivo de más capacidad es la EPROM 2732 con 4K de memoria, teniendo disponibles desde A12 hasta A15 en el microprocesador para la función de decodificación, por lo que A-13, A-14, A15 se conectan en A, B, C del decodificador.

A continuación se encuentra la tabla de decodificación:

DIRECCIONAMIENTO

A15	A14	A13	DECODIFICADOR	DISPOSITIVO
C	B	A		
0	0	0	Y0	RAM 6116
0	0	1	Y1	---
0	1	0	Y2	---
0	1	1	Y3	---
1	0	0	Y4	PIA 1
1	0	1	Y5	PIA 2
1	1	0	Y6	---
1	1	1	Y7	EPROM 2732

Después de ver esta tabla podemos darnos cuenta que mandando el código correcto al decodificador podremos direccionar cualquier dispositivo y su localidad deseada, en nuestro caso sólo 2 PIAS y las memorias, pero por ejemplo Y1, Y2, Y3, Y6 pueden ir conectadas a otros dispositivos como el ACIA, el TIMER, etc. las salidas de decodificación YX deben conectarse a las entradas habilitadoras de cada dispositivo como lo podemos apreciar en nuestro sistema; Y0 va a $\overline{E}$ que es la terminal que habilita la RAM 6116, Y4 se conecta al PIA 1 para habilitarlo en su terminal $\overline{CS2}$; de igual forma Y5 habilita al PIA 2. Se puede direccionar cualquier dispositivo con cada una de las salidas YX del decodificador, sólo que existe una restricción y se encuentra en la EPROM 2732, que necesariamente debe habilitarse con Y7, esto es así porque cuando el microprocesador recibe una señal de reestablecimiento en su terminal RESET, inmediatamente direcciona la localidad FFFE habilitando unos desde A15 hasta A1, entrando al decodificador en A15, A14, y A13 el número

binario 111, con lo que decodificamos Y7, y con ello la habilitación de la EPROM. Si no conectamos Y7 a la entrada habilitadora de la EPROM nunca se llegará al programa de reestablecimiento contenido en ella. Para comprender más lo anterior, cuando carguemos o mandemos la dirección 0XXX o 1XXX podremos direccionar toda la RAM; con 8XXX o 9XXX el PIA 1; con AXXX o BXXX el PIA 2; con EXXX o FXXX la EPROM. El porqué de poder direccionar el dispositivo con dos diferentes direcciones radica en que A12 no se conecta a ningún lado por lo que no importa si existe un 1 o un 0 en su línea, es decir, se puede accesar o direccionar, por ejemplo a la misma localidad de la RAM con 0XXX o con 1XXX ya que no importando A12 en A13, A14, y A15 entra el binario 000 decodificando Y0 para habilitar la memoria RAM 6116. Recordemos que sólo A15, A14 y A13 se utilizarón para este fin, pero tenemos desde A12 hasta A0 para direccionar cualquier localidad de cada dispositivo ; por ejemplo si se desea direccionar la localidad AFF de la EPROM debemos cargar el número hexadecimal FAFF para llegar a esa localidad, el número hexadecimal F de la izquierda corresponde al código de decodificación y el número AFF indica la localidad dentro de la memoria.

Después de haber comprendido el direccionamiento seguimos con el bus de datos D0-D7 el cual se manda línea por línea a todo dispositivo. Tenemos enseguida BA (5), línea disponible y BS (6), bus status, o estado de línea, que se utilizan para conocer el estado del microprocesador; no se conectan a ningún lado ya que no es de utilidad para los fines de este proceso. R/$\overline{W}$ (32), señal que indica la dirección de transferencia en las líneas de datos, se conecta a los dispositivos que lo requieran como los

PIAS y la RAM; la EPROM no tiene terminal R/$\overline{W}$ ya que sólo se puede leer. $\overline{DMA}$ (33), sirve para suspender la ejecución y tener disponibles las líneas del - microprocesador para diferentes usos como el de direccionamiento directo de memoria, o para refrescar memorias dinámicas, no se utilizó y se desactivó conectándola a 5V.

E (34), es una señal de sincronización a la frecuencia de operación, debe conectarse a todos los dispositivos utilizados por el microprocesador, - los PIAS tienen su entrada E donde es conectada la señal; pero en el caso de las memorias, que no tienen entrada E, fué necesario mandar la señal de sincronización al decodificador en G1, una de sus entradas habilitadoras provocando con esto la señal E de sincronización del micro en las entradas habilitadoras de las memorias, para de esta manera poder sincronizar correctamente los tiempos de accesos a los dispositivos y no perder información. Para comprender mejor cómo se produce el efecto anterior, analicemos las figuras 1.1 y 1.2. en la figura 1.1. observamos las señales E y Q además del bus de direcciones y de datos, aquí vemos que cuando Q sube garantizamos una dirección estable antes de que el dato sea leído por el dispositivo, en este mismo instante el dato es colocado en la línea de salida o bus de datos, siendo leído hasta la siguiente bajada de E; esto no tiene mayor problema en los PIAS ya que poseen la misma señal de sincronización E, pero cómo lograrlo en las memorias, ese es el problema, por eso se conectó E a la entrada habilitadora G1 del decodificador, logrando que antes de que E suba y habilite al codificador y con ello a cualquier memoria, exista una dirección estable, garantizándonos la correcta dirección a accesar, de esta manera cuando E suba

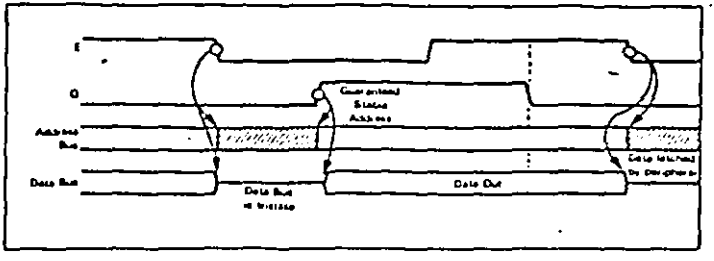


FIGURA I.1. E Y Q EN UN CICLO DE ESCRITURA.

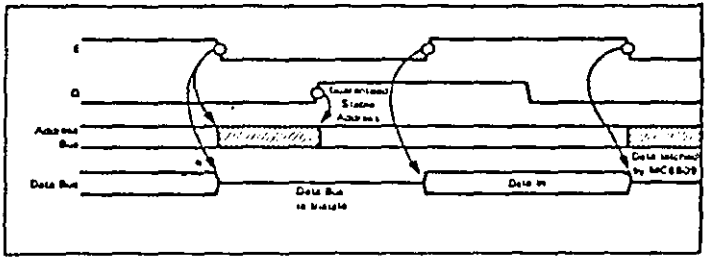


FIGURA I.2. E Y Q EN UN CICLO DE LECTURA.

y se active el dispositivo se direccione la localidad correcta.

Analizando la figura 1.2. correspondiente a un ciclo de lectura, observamos el mismo funcionamiento anterior, la única diferencia es que aquí los datos se presentan hasta la primer subida de E después de que Q sube, y son leídos por el microprocesador hasta la siguiente bajada de E. En las memorias es lo mismo, cuando Q va a alto se garantiza una dirección estable provocando que cuando E suba se trabaje con la correcta dirección. Al subir E, se habilita el decodificador, después del pequeño tiempo de respuesta del decodificador se habilita la memoria, y se accesa la localidad indicada, el dato es mandado a la línea de datos para que cuando E vaya a bajo sea leído por el microprocesador.

La salida Q (35), es una señal de cuadratura de E. El inicio de este pulso indica que la dirección presente en las líneas de direccionamiento es correcta y estable. Esta terminal no se conectó a ningún lado, aunque el microprocesador esté trabajando con ella como lo analizamos anteriormente.

MRDY (36), terminal de control que permite ampliar el tiempo del pulso E de sincronización para memorias o dispositivos lentos se desactivó conectándola a 5V, ya que las memorias utilizadas tienen tiempos de accesos menores al pulso E. Medio ciclo del pulso E de sincronización tarda 800nS. Ya que tenemos un cristal oscilador de 2.45 Mh y el tiempo de acceso de la memoria por ejemplo la EPROM 2732 es de 120nS. Con esto queremos decir que tenemos tiempo de sobra para realizar un acceso a memoria.

$\overline{\text{RESET}}$ (37). Esta entrada está acoplada internamente a un circuito disparador SCHMITT en donde una señal de bajo nivel que dure más de un ciclo de reloj reestablecerá al microprocesador. Debido a esa entrada disparadora SCHMITT de un voltaje de umbral mayor que el de los periféricos estándar, se puede usar un simple circuito R-C para reestablecer el sistema completo. Este voltaje de umbral alto, nos asegura que los periféricos dejarán el estado de reestablecimiento antes que el microprocesador.

Mediante división de voltaje observamos que al recibir una señal de reestablecimiento obtenemos un valor cercano a cero en la terminal $\overline{\text{RESET}}$:

$$Vc(t) = \frac{(10)\ (5)}{470 + 10} = .104\ V$$

Esta señal entra al microprocesador y lo reestablece. De la misma forma que lo hace en los dos PIAS en su correspondiente entrada. Las memorias no reciben esta señal ya que no necesitan limpiar ningún registro interno. Inmediatamente después de ser desactivada la señal de reestablecimiento, el capacitor vuelve a cargarse a 5V, para que nuevamente se reciban en $\overline{\text{RESET}}$ del micro y PIAS, manteniendo un estado estable.

Enseguida tenemos las terminales 38 y 39, EXTAL y XTAL donde se conectó un cristal de 2.45 Mhz; este valor de frecuencia nos genera un tiempo lo suficientemente amplio en un ciclo de la señal de sincronización para garantizar un acceso sin pérdida de información en un ciclo de lectura o escritura. $\overline{\text{HALT}}$ (40); un nivel bajo en este contacto causará que el microprocesador pare de correr al final de la instrucción que se esté ejecutando; fué conectado a 5V

para desactivarlo por no utilizarse.

Con esto concluímos el análisis de las conexiones del microprocesador hacia los demás dispositivos, de los cuales se analizarán las terminales restantes acontinuación.

En la memoria RAM 6116 quedó únicamente de analizar $\overline{G}$, que es la terminal de OUTPUT ENABLE; un nivel bajo aplicado en esta terminal habilitará la salida de datos para poder ser leídos. Se conectó a tierra para tener disponible los datos en todo momento que se realice la función de lectura; ya que en la terminal $\overline{W}$ entra un valor lógico de 1, desactivando la función de escritura. Cuando entre en $\overline{W}$ un nivel lógico 0 se habilitará la función de escritura no importando lo que se tenga en la terminal $\overline{G}$.

En la EPROM 2732 tenemos que Y7 del decodificador se conecta a las dos entradas habilitadoras $\overline{OE}$ y $\overline{CE}$, con esto lo único que conseguimos es lograr el acceso más rápido de la memoria que es 120ns. En $\overline{OE}$ según fabricante; el tiempo de acceso en $\overline{CE}$ es de 200ns.

En cuanto al decodificador ya fué analizado anteriormente, aquí únicamente se conectó las terminales 4 y 5 que son entradas habilitadoras a nivel bajo y G1 a E del microprocesador.

En el PIA 1 se conectó al puerto -A- el teclado telefónico, las líneas de común están en la posición más significativa y las líneas de interrupción en -

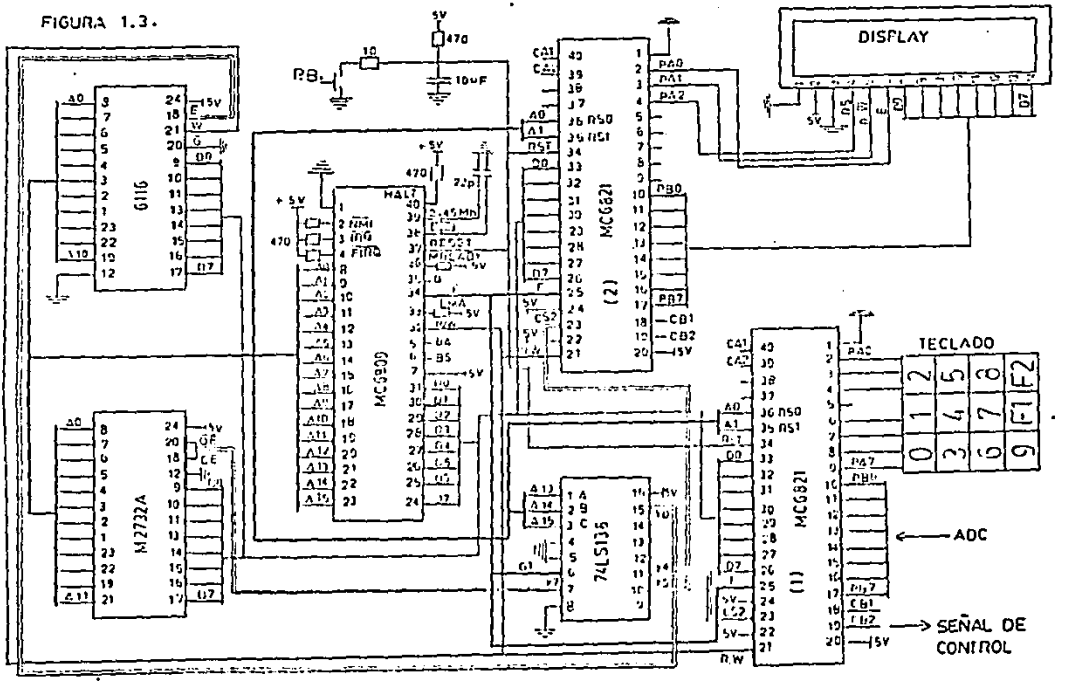
la posición menos significativa; en el segundo capítulo se analizará porqué - se conectó el teclado de esta manera ya que su función depende del programa que lo controla. La salida digital del convertidor se conectó al puerto -B- de este PIA desde PB0 hasta PB1. La señal de control la podemos observar en CB2, que es la línea de control del puerto B. Esta señal de control es la - que se generará cuando la temperatura actual sea igual o sobrepase a la que el usuario establecerá al inicio de el proceso.

El PIA 2 se utilizó para conectar el Display. En el puerto -A- se conectó la terminal E, que es la entrada habilitadora del Display; R/$\overline{W}$, señal que indica al Display lectura o escritura y RS, selección de registro, que se utiliza para seleccionar el registro interno del Display en PA0, PA1, PA2, - respectivamente. Con un nivel bajo en la terminal RS del Display el dato enviado será el código de una instrucción a ejecutar, mientras que con un nivel alto el dato enviado será mostrado en pantalla. En el puerto -B- de este PIA se conectó la línea de datos del Display D0-D7 desde PB0 hasta PB7.

Este orden de conectar los dispositivos periféricos a los PIAS es debido a que dentro del programa se establecerá en forma correcta la manera de - como trabajarlos, es decir, las direcciones que tendrán los dispositivos con relación a los puertos de los PIAS y la forma en que obtendremos y mandaremos información a cada uno de ellos.

Con esto concluímos la explicación de este circuito que se realizó en forma concreta; no se analizó con profundidad lo referente a todas las seña

FIGURA 1.3.


DISPLAY
TECLADO
0 1 2
3 4 5
6 7 8
9 F1 F2
ADC
SEÑAL DE CONTROL
MC6821 (2)
MC6821 (1)
MC6800
6116
M2732A
74LS138
PB
10uF
470
22p
+5V
R/W

les ya que esta tesis se fundamenta sólo en la aplicación del microprocesador no en su funcionamiento interno. Cualquier referencia a los dispositivos utilizados puede encontrarse en el apéndice de esta tesis.

1.2. DISEÑO Y DESCRIPCION DEL CIRCUITO SENSOR-CONVERTIDOR.

El circuito presentado en la figura 1.5. está diseñado para entregar en forma digital de 8 bits el voltaje que entrega el sensor de temperatura LM - 35D, después de pasar por un arreglo de amplificadores operacionales que se requieren para entregar la cantidad exacta de señal al convertidor analógico-digital 0803 con el fin de obtener una escala de conversión adecuada.

En este diseño se buscó algún dispositivo que pudiera sensar temperatura y de alguna forma poder trabajar con ella, encontré el sensor de temperatura LM 35D el cual varia 10mV por cada grado centígrado en forma lineal, esto fué muy importante porque se ahorró el trabajo de linealizar la señal entregada. Además de esto el sensor no necesita de calibración externa, ni tampoco el - restarle un voltaje constante a la salida para proporcionar la señal exacta - de voltaje a la temperatura en que se encuentre; por ejemplo, si polarizamos el sensor en donde la temperatura es de 25 grados centígrados, se podrá leer directamente .25V en su terminal Vout. Sin ningún circuito adicional; si la temperatura se eleva 2 grados, se leerá el valor .27V, de igual forma si la temperatura se disminuye se leerá el correspondiente valor.

El sensor se conectó como se ve en el circuito, tomando de Vout la señal

de voltaje; esta señal es la que se requiere convertir a forma digital para que el microprocesador pueda trabajar con ella; pero antes se tiene que ver que requisitos son los necesarios en el convertidor, es decir que cantidad de voltaje se debe introducir para generar cada conversión.

Este convertidor A/D 0803 realiza una conversión cada 20mV, los cuales se introducen en la terminal Vin (6). Esto puede verse en la siguiente tabla.

VOLTAJE (V)	BINARIO
0	0 0 0 0 0 0 0 0
.020	0 0 0 0 0 0 0 1
.040	0 0 0 0 0 0 1 0
.060	0 0 0 0 0 0 1 1
.080	0 0 0 0 0 1 0 0
.100	0 0 0 0 0 1 0 1
.120	0 0 0 0 0 1 1 0
.140	0 0 0 0 0 1 1 1
.160	0 0 0 0 1 0 0 0

Como se acaba de observar cada 20 mV se realiza una conversión; recordando que el sensor varía 10mV por cada grado centígrado, fué necesario introducir esta señal a un amplificador diferencial con ganancia de 2 con el fin de obtener 20mV, para que se entregue de este modo una conversión por la variación de cada grado centígrado.

La configuración del amplificador diferencial se encuentra en la figura 1.4.

La ganancia que se encuentra implícita en la siguiente ecuación para el voltaje de salida del amplificador diferencial es R2/R1.

$$VO = \frac{R2}{R1} (V2 - V1)$$

Observando el circuito figura 1.5. tenemos valores de 2K en R2 y valores de 1K en R1, con lo que la ecuación asume como resultado una ganancia de 2 independiente de los valores de V2 y de V1.

$$VO = \frac{2K}{1K} (V2 - V1) \qquad VO = 2 (V2 - V1)$$

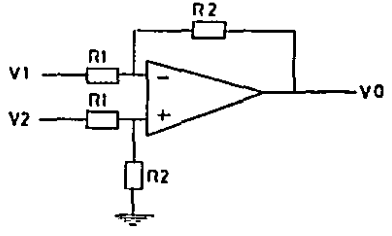


Figura 1.4. Amplificador Diferencial.

En V2 de este amplificador, que es la entrada no inversora se introduce la señal que entrega el sensor LM35D, después de haber pasado por el seguidor de voltaje, cuyo funcionamiento es únicamente para evitar un dren de corriente de la señal entregada por el sensor, sin modificar su valor. En V1 de el amplificador diferencial se introduce una señal de 0V ya que en V2 se reciben

valores exactos para convertir y no se vió la necesidad de fijar o establecer un voltaje de referencia.

Para observar todo el procedimiento anterior referente a la aplicación de la ecuación del amplificador diferencial, enseguida se presenta una tabla con los valores correspondientes a las señales entregadas para ciertos valores de temperatura.

TEMPERATURA	VI	V2 (V)	VO (V)	A/D	B-10
26	0	.26	.52	11010	26
27	0	.27	.54	11011	27
28	0	.28	.56	11100	28
24	0	.24	.48	11000	24
23	0	.23	.46	10111	23

En cuanto al control del convertidor se dejó de forma de operación libre conectando $\overline{INTR}$ a $\overline{WR}$ con $\overline{CS=0}$, en $\overline{WR}$ se introdujo la señal de un reloj LM 555 con el fin de que éste fuera el activador para ejecutar la conversión, la cual se efectúa cada 2s aprox.: se dejó 2 segundos ya que después de muchas pruebas se observó que la temperatura no varía más rápido de ese tiempo en condiciones ambientales sometidas a calentamiento, pero en pruebas donde el aumento de temperatura es muy rápido, puede variarse el reloj aumentando la frecuencia de sus pulsos, obteniendo menor tiempo entre cada señal activadora de conversión.

Dependiendo de las condiciones en las que se someterá el control, y de las señales que se requieran, se realizarán todos los ajustes necesarios.

Las restantes conexiones que se observan en el convertidor se tomaron del diagrama de prueba del convertidor 0803 del linear Data Book, National Semiconductor, página 3-34.

Cualquier referencia a los dispositivos utilizados en este circuito se podrá encontrar en el apéndice de esta tesis.

Los cálculos referentes al reloj son los siguientes:

* F = 1.44 / (RA + 2RB) * C

  F = 1.44 / (470 + 2(1.5K)) * 1000 u

  F = .45714 Hz.

* T = 1 / F

  T = 2.1875 seg.

* T alto = .7 (RA + RB) * C

  T alto = 1.155 seg.

* T bajo = .7 RB C

  T bajo = 1.05 seg.

CIRCUITO SENSOR-CONVERTIDOR

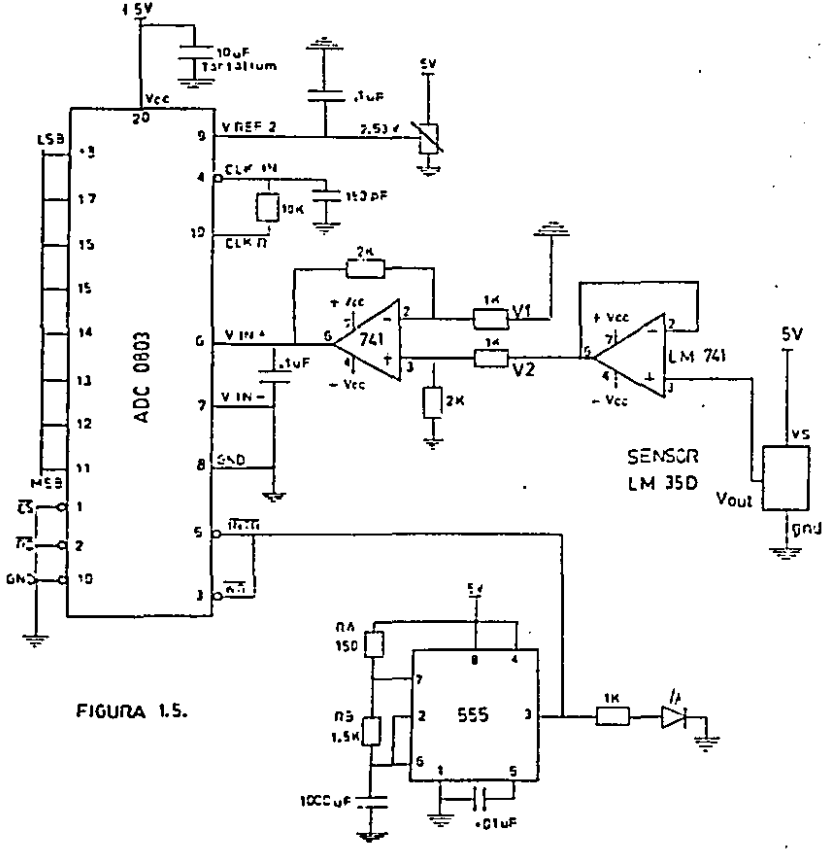


FIGURA 1.5.

CAPITULO II

DISEÑO DEL PROGRAMA

Este capítulo es el más importante ya que aquí se explica el diseño del programa, el cual es el encargado de hacer funcionar correctamente el control según las características establecidas, es decir, va a coordinar la relación entre los dos circuitos controlador y sensor-convertidor para obtener la información precisa y generar la adecuada señal de control.

A continuación se dará una breve explicación de todas las subrutinas utilizadas, para posteriormente dedicarnos a cada una en especial, donde se mostrará la subrutina tal y como quedó para usarse, especificando los puntos más importantes y los diagramas de flujo de las más complicadas.

II.1 DISEÑO GENERAL.

Iniciamos con la subrutina -RESET- que es la encargada de reestablecer el sistema, es decir, inicializa todos los registros, establece la dirección de los stacks o pilas de máquina y usuario establece también como funcionarán los puertos de los PIAS, manda los códigos de inicialización de Display, y espera hasta que se presione la tecla de inicio de proceso.

Pensando en que el diseño se estableció para que el usuario introdujera el valor de temperatura máxima, punto donde actuará la señal de control, se creó la subrutina -TECLA- que determina mediante rastreo que valor es el que se introduce por medio del teclado, para trabajar posteriormente con él. Como necesitamos introducir un valor entre 0 y 99, límite del máximo valor permitido, se encuentra la subrutina -DATO2- que va 2 veces a la subrutina —

-ECO-; su función se explica en seguida, dentro de la cual se llama a -TECLA- para obtener el número que se quiere introducir y almacenarlo en un registro para utilizarlo después.

Para poder ver todo lo que se teclea está la subrutina -ECO- que es la que se encarga de llamar a -TECLA- cada vez que ésta se requiere. -ECO- llama a -TECLA- para obtener el valor tecleado, después llama a -ASCII- que es una subrutina de conversión de hexadecimal a código ASCII dentro de la cual se llama a -DISPLA- que es una subrutina que maneja el Display, mostrando cualquier código ASCII que se le envíe. De esta forma es como podemos ver en el Display todo lo que se introduce por el teclado.

La subrutina -PROCES- es una de las más importantes pues es la encargada de llevar a cabo el proceso de control, recibiendo señales del convertidor analógico-digital y actuando de acuerdo a ellas.

Dentro de -PROCES- se llama a otras subrutinas como -DECHEX- que es una subrutina que convierte el valor que viene de -DATO2- a hexadecimal, ya que cuando introducimos el valor máximo de temperatura se realiza en decimal, por ejemplo 50 grados, pero es necesario convertirlo para que el microprocesador trabaje con él ya que si no es así el microprocesador tomará el valor tecleado directamente como hexadecimal, es decir, 50 en hexadecimal es 80 en decimal, lo cual no es lo que se quería introducir.

La subrutina -MOSTRA- es llamada también dentro de -PROCES- la cual se encarga de mostrar constantemente en Display el valor entregado por el con-

vertidor, para lo cual se auxilia de -HEXDEC- que funciona únicamente para convertir a decimal el valor que es leído del convertidor, el cual está en hexadecimal; enseguida se llama a -BDISP- subrutina que toma ese valor para separarlo y mandarlo a la subrutina -ASCII- para que sea convertido a código ASCII y mostrado en Display.

Existe la subrutina -DIV-, la cual divide dos números, ésta se utiliza dentro de -HEXDEC-. La subrutina -RETARD- únicamente se utiliza para hacer tiempo en el proceso. -ALFA- subrutina que se encarga de mostrar en Display frases ya establecidas en las subrutinas -RESET- y -PROCES-; por último tenemos la subrutina -COM- la cual se encarga de mandar a Display comandos para manejo de cursor o pantalla; es diferente a -DISPLA-, que sólo muestra caracteres ASCII.

En general son todas las subrutinas utilizadas, enseguida se analizarán cada una de ellas haciendo énfasis en los puntos más importantes.

II.2 SUBRUTINA RESET.

-RESET- es la subrutina encargada de inicializar todo el sistema como PIAS, Display, y stacks. Esta es la primer subrutina a donde va el microprocesador, o donde comienza el proceso a funcionar. Al recibirse en el microprocesador una señal de reestablecimiento se limpian todos los registros internos y se direccionan las localidades FFFE y FFFF, lugar de la EPROM donde se establecerá la dirección a donde el microprocesador deberá

dirigirse, es decir, donde se encuentra nuestra subrutina -RESET-, no importa que la subrutina -RESET- esté al principio, en medio o al final de nuestro programa, éste funcionará igual siempre y cuando sea perfectamente ensamblado.

La subrutina -RESET- comienza estableciendo la longitud de los stacks:

LDS	#$7FF,	Hacemos que el registro S apunte a la dirección 7FF que es la posición más baja de la RAM 6116.
LEAS	-20,S	Apuntamos nuevamente a 20 espacios más arriba.
TFR	S,U	Transferimos el valor del registro S al registro U.
LEAS	-2,S	El registro S apunta a 2 localidades atrás.
CLR	-1,U	Se limpia la localidad anterior a donde apunta U.

Después de realizar esta sección, el stack de máquina apunta desde la posición 7FF menos 22 hacia atrás. El stack de usuario apunta desde 7FF menos 21 hasta 7FF, el cual se utilizará en la subrutina -DISPLA- para almacenar todos los caracteres que se estén mostrando cuando ocurra un retorno y presentarlos en la línea anterior, ya que el Display utilizado consta de 2 líneas.

Enseguida se inicializan los PIAS. Tomando en cuenta que se han inicializado los registros, es decir que hay ceros en los registros de control A y B, lo cual significa que podemos entrar directamente a programar los puertos o en todo caso a usarlos como entradas ya que existen ceros en el registro de dirección de datos y no hay necesidad de programarlos.

LDD #$F034 Cargamos al registro D el número F034, A=F0, B=34.

STD $8000 Mandamos el valor del registro A, A=F0, hacia la dirección - 8000, registro de dirección de datos A, estableciendo como - entradas los primeros 4 bits o menos significativos y como- salidas los 4 bits más significativos del puerto -A- que serán utilizados para el manejo del teclado. El valor del re - gistro B=34, 0011 0100, se manda a la dirección 8001, con lo cual ponemos el bit 2 del registro de control -A- en 1, para ya poder utilizar este puerto.

STB $8003 Con esto mandamos B=34 a 8003, registro de control -B- del- PIA 1 habilitando a 1 el bit 2 del registro de control, para ya utilizar el puerto B, que quedó todo como entrada, debido a la inicial señal de reestablecimiento; aquí en donde se conecta la salida del convertidor A/D. Además ponemos en 1 los bits 4 y 5 del mismo registro para poder utilizar el control CB2 como salida; ver tabla 2.4. del 6821 en el apéndice.

LDA #$FF Cargamos al registro -A- el número FF.

STD $A000 El puerto -A- del PIA 2 queda todo como salidas por haber - mandado A=FF a la localidad A000 que es el registro de dirección de datos. Y nuevamente B=34 se manda a A001, registro - de control -A-, para ya poder utilizar el puerto -A- del PIA 2.

STD $A002 El puerto B de este PIA queda de forma idéntica al puerto A. Se dejaron los dos puertos como salidas para el manejo de Display; con

el puerto -A- se controlan sus terminales de control y con el puerto -B- el acceso de datos.

Después de inicializar los PIAS se mandan los códigos de inicializa - ción de Display a -COM-, que es la subrutina encargada de las funciones del

```
Display: LDA    #$38
         LBSR   COM
         LDA    #$0F
         LBSR   COM
         LDA    #$01
         LBSR   COM
         LDA    #$06
         LBSR   COM
```

Con el 38 establecemos las siguientes funciones; interface o comunicación de 8 bits, y manejo de línea de 16 caracteres.

Con el 0F se establece el encendido del Display, cursor habilitado y parpadeo del mismo. Con el código 01 se limpia pantalla y el cursor se posiciona en línea 1, columna 1, o home position. Con el 06 se logra que el cursor se mueva hacia la derecha sin corrimiento de Display.

Para comprender los códigos anteriores revisar la tabla 6.1. del Dis - play de cristal líquido en el apéndice de esta tesis.

Después se manda a Display el mensaje ya establecido, por medio de — -ALFA-. Enseguida leemos teclado, si es la tecla indicada, se va inmediatamente a proceso, de lo contrario muestra un mensaje de error.

La subrutina -RESET- es la siguiente:

```
RESET   LDS   #$7FF
        LEAS  -$20,S
        TFR   S,U
        LEAS  -2,S
        CLR   -1,U
        LDD   #$F034
        STD   $8000
        STB   $8003
        LDA   #$FF
        STD   $A000
        STD   $A002
        LDA   #$38
        LBSR  COM
        LDA   #$0F
        LBSR  COM
        LDA   #$01
        LBSR  COM
        LDA   #$06
        LBSR  COM
TEC     LEAY  MSG,PCR
        LBSR  ALFA
        LBSR  TECLA
        CMPA  #$0A
        BEQ   PRO1
        LEAY  ERR,PCR
        LBSR  ALFA
        LBRA  TEC
PRO1    LBSR  PROCES
        BRA   TEC
MSG     FCB   $0D
        FCC   \MANUEL LOMELI P.\
        FCB   03
ERR     FCB   $0D
        FCC   \ERROR\
        FCB   03
        RTS
```

1. SUBRUTINA TECLA.

La subrutina tecla es la encargada de controlar el teclado, que es el

medio por donde se introduce información al sistema; con motivo de minimizar las entradas de un puerto y con la posibilidad de tener disponibles como mínimo 11 teclas, del 0 al 9 y una de función se utilizó un teclado de matriz de 3*4, que es el tradicional teclado telefónico.

El primer punto a solucionar es cómo está distribuida esa matriz en cuanto a las teclas y terminales de entrada y salida. Este teclado por su parte posterior tiene 8 terminales, de las cuales 4 pueden ser conectados a Vcc y 4 conectados a tierra o al revés, logrando así la matriz de funcionamiento cuando se oprime cada tecla, que es la que cierra el circuito.

Numerando de 1 a 8 las terminales no importando si se desea empezar de izquierda o derecha, siempre y cuando 4 vayan a Vcc y 4 a tierra, tenemos la siguiente tabla que establece que tecla es la que cierra el circuito entre dos de las 8 terminales.

TERMINALES	TECLA	TERMINALES	TECLA
1-5	2	2-7	7
2-5	1	3-7	8
4-5	3	4-7	9
1-6	5	2-8	*
2-6	4	3-8	0
4-6	6	4-8	#

Una vez que se tiene esta tabla se deduce la forma de cómo serán conec-

tadas las terminales al puerto -A-. Quedando de la siguiente forma:

A0	A1	A2	A3	A4	A5	A6	A7
2	1	4		5			
2	1	4			6		
2	3	4				7	
2	3	4					8

De PA0 a PA2 queda entonces la matriz de 3*4 que corresponde al teclado:

1 2 3
4 5 6
7 8 9
* 0 #

PA3 no se utilizó; de PA4 a PA7 quedaron los pines 5, 6, 7, 8 que mediante el programa se pondrán a tierra uno por uno con el fin de identificar a - que renglón corresponde la tecla presionada, para posteriormente identificarla columna y establecer el valor que nosotros ocupemos de esa tecla.

Los valores que finalmente tendrán las teclas, los cuales serán calculados por el programa al momento de detectarlas son los que a continuación se muestran:

0 1 2
3 4 5
6 7 8
9 A B

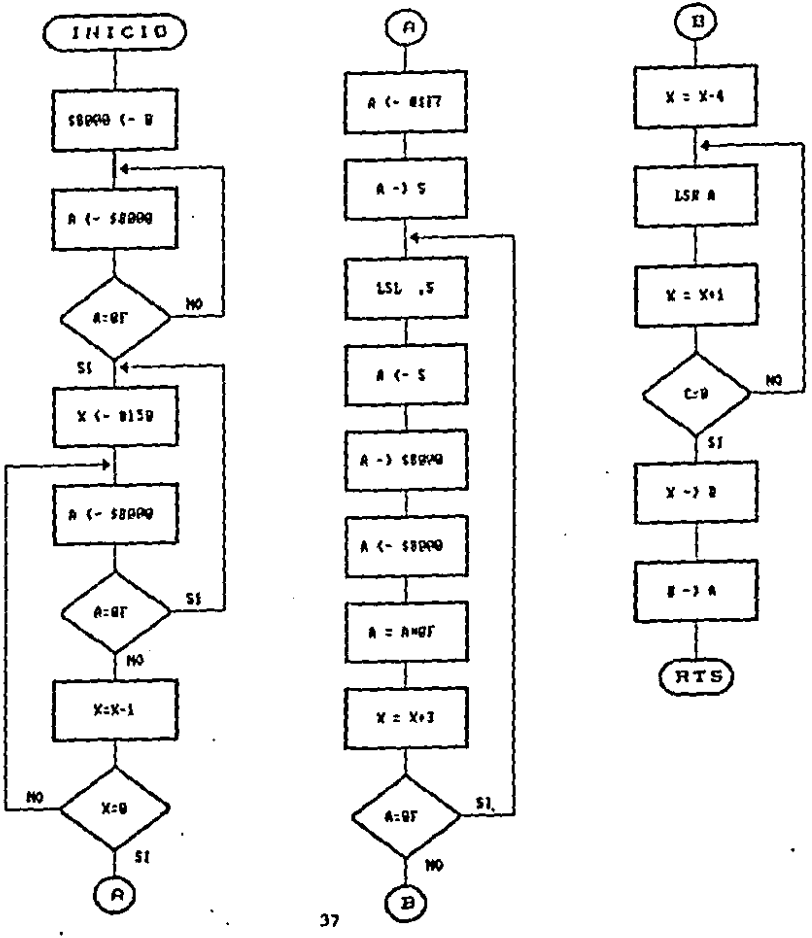
De 0 a 9 para formar cualquier número entre 0 y 99, que será la temperatura máxima que se introduce; "A" para identificar que se entra al proceso y "B" en este caso no tiene utilidad.

Un aspecto importante es el ruido que se produce por el rebote de cualquier contacto, en nuestro caso el teclado, este ruido causaría que al presionar una tecla, el valor de ella se presentara varias veces como si se hubiera tecleado; esto debido a la rapidez con que el microprocesador realiza la lectura al puerto, en el orden de microsegundos, por lo que antes de que se desactive finalmente el contacto por este efecto de rebote el microprocesador sigue leyendo ese mismo valor; este problema se soluciona estableciendo dentro del programa un ciclo que haga el tiempo necesario para asegurar la desactivación de la tecla antes de que se realice otra lectura al puerto. A continuación se hará referencia al diagrama de flujo de la subrutina --TECLA-.

Al inicio del programa se manda un cero a la localidad 8000, que es el puerto -A- del PIA 1, donde se encuentra conectado el teclado; con esto se mandan ceros a los últimos 4 bits o más significativos del puerto; los menos significativos no responden por que fueron programados como entradas; esto es así para que al momento de que una tecla sea presionada y cierre el circuito interno del teclado, se introduzca un cero a uno de los 3 bits menos significativos para que cuando el microprocesador lo identifique mediante la lectura de estos bits se pueda entrar enseguida a rastrear cual fué esa tecla.

Enseguida está un primer ciclo que nos detecta si existe información antes de la actual tecla activada, si no existe, se puede continuar con el programa, es decir, A=0F, lo cual nos dice que en los primeros bits donde se conectaron las terminales de interrupción no hay un cero. De lo contra--

SUBRUTINA TECLA .



rio se vuelve a leer el puerto hasta la desactivación de esa información anterior.

Encontramos después el ciclo donde leemos el puerto y hacemos un tiempo para asegurar no leer posteriormente ese mismo valor. Se carga al registro X el número decimal 150, leemos el puerto -A- donde están conectadas las tres líneas del teclado, si no se ha presionado alguna tecla el programa se queda en ese ciclo de lectura hasta detectar información. Si ya se ha presionado cualquier tecla, es decir, A <> 0F, lo cual indica que ya existe algún cero en alguno de los bits menos significativos y por lo tanto A <>0F empieza a funcionar el contador hasta disminuir el valor de X a 0 esto con el fin de hacer tiempo y asegurar que antes de que el microprocesador salga de esta subrutina y vuelva a leer otro valor se haya desactivado el efecto de rebote, que anteriormente se mencionó, por eso se cargó el valor de 150 al registro X, que es simplemente un valor lo suficiente grande para garantizar la desactivación de ese efecto.

Enseguida tenemos la sección de detección de renglón donde cargamos al registro -A- el número F7, 1111 0111, y lo guardamos en el stack de máquina para poder utilizar nuevamente el registro; se realiza una serie de corrimientos a la izquierda de este valor y durante cada uno de ellos el contenido de esa localidad previamente recorrido se manda a la dirección 8000, con el motivo de poner cada vez un cero en cada uno de los últimos 4 bits del puerto A, donde están conectadas las terminales de común del teclado, hasta registrar mediante lecturas enmascaradas de los bits menos significativos,

LDA $8000, A=A*0F, que renglón fué donde se activó la tecla, e incrementamos un contador de 3 en 3, que corresponde a las 3 teclas de cada renglón.

Enseguida tenemos un ciclo para detectar que columna fué tecleada y así tener el rastreo completo. Antes de entrar en este ciclo se realiza una resta al contador, X=X-4, dado que antes de identificar el renglón se realiza el incremento. La detección de la columna se realiza mediante corrimientos del registro A, que tiene la lectura del puerto y por consiguiente un cero en alguna posición, que fué cuando se detectó el renglón. Este corrimiento es hacia la derecha, LSR A, así mediante comparaciones del bit de acarreo del código de condición llegaremos hasta el bit que contiene el cero; durante cada corrimiento se incrementa nuestro contador en uno, X=X+1, que corresponde a cada columna y así tenemos al final en el contador X el valor de la tecla que se presionó.

Como necesitamos que el valor de la tecla quede en el registro A, por que es el registro que se asignó desde un principio para manejo de datos, hacemos transferencias hasta pasar el valor de X hacia A; X→D, B→A.

```
Subrutina  Tecla:

TECLA      PSHS   X,B
           LEAS   -1,S
           CLR    $8000
UNO        LDA    $8000
           CMPA   #$0F
           BNE    UNO
DOS        LDX    #150
TRES       LDA    $8000
           CMPA   #$0F
```

```
            BEQ   DOS
            LEAX  -1,X
            BNE   TRES
            LDA   #$F7
            STA   ,S
CUATRO      LSL   ,S
            LDA   ,S
            STA   $8000
            LDA   $8000
            ANDA  #$0F
            LEAX  3,X
            CMPA  #$0F
            BEQ   CUATRO
            LEAX  -4,X
CINCO       LSRA
            LEAX  1,X
            BCS   CINCO
            TFR   X,D
            TFR   B,A
            LEAS  1,S
            PULS  X,B
            RTS
```

II.4 SUBRUTINA -ASCII-.

La función de esta subrutina es la de convertir cualquier valor hexadecimal a su correspondiente ASCII, con el fin de que pueda ser mandado al Display mediante la subrutina -DISPLA- y poderlo observar en pantalla.

La subrutina -ASCII- es la siguiente:

```
ASCII       PSHS    X,A
            LEAX    TABLA,PCR
            ANDA    #$0F
            LDA     A,X
            LBSR    DISPLA
            PULS    X,A
            RTS
TABLA       FDB     $3031
            FDB     $3233
            FDB     $3435
```

```
        FDB     $3637
        FDB     $3839
        FDB     $4142
        FDB     $4344
        FDB     $4546
```

A esta subrutina llegan valores únicamente de 4 bits y mediante acceso indexado a la tabla obtenemos el correspondiente valor ASCII, para posteriormente mandarlo a la subrutina -DISPLA-.

II.5 SUBRUTINA ECO.

Esta subrutina fué elaborada con el fin de que sirviera de enlace entre teclado y el Display, es decir, es la encargada de llamar a -TECLA- -- cuando se introducen valores por teclado, después llama a -ASCII- para convertir cada valor y transferirlo a pantalla.

Subrutina ECO se muestra a continuación:

```
ECO         BSR     TECLA
            CMPA    #$A
            BHS     NOP1
            BSR     ASCII
            ANDCC   #$FE
            BRA     ECOFIN
NOP1        ORCC    #$1
ECOFIN      RTS
```

Su función es la siguiente, llama a -TECLA- para obtener el valor presionado y lo compara con A, si es mayor o igual termina subrutina con una indicación, ORCC #1, que pone el bit de acarreo en 1, la cual se utiliza pa

ra indicar a -DATO2- que el valor presionado no está entre 0 y 9, logrando así que el valor no aparezca en pantalla hasta que se teclee un valor permitido.

11.6. SUBRUTINA DATO2.

Esta subrutina es utilizada por -PROCES- para permitir la entrada correcta de valores al sistema, es decir, si se introduce un 35, lo que hacemos primero es teclear el tres y luego el cinco, números que aparecerán instantaneamente en Display si es que están entre 0 y 9, de lo contrario se volverá a leer el teclado hasta encontrar un valor permitido. para que este valor quede almacenado correctamente es necesario primero recorrer ese trea, que recién presionado está en la posición menos significativa hacia la más significativa, esto lo logramos con 4 veces la instrucción LSLA, para posteriormente almacenarlo en el stack, todo esto antes de ir por el segundo número que es el cinco; una vez introducido el cinco lo que se hace es simplemente sumarlo al valor almacenado para así tener el dato almacenado de forma correcta en el stack.

Subrutina DATO2:

```
DATO2   LEAS    -1,S
NOP2    BSR     ECO
        BCS     NOP2
        LSLA
        LSLA
        LSLA
        LSLA
        STA     ,S
```

```
NOP3    BSR    ECO
        BCS    NOP3
        ADDA   ,S
DATFIN  LEAS   1,S
        RTS
```

11.7. SUBRUTINA BDISP.

Esta subrutina es utilizada por -MOSTRA- para mostrar correctamente - el valor que está entregando el convertidor A/D que corresponde al valor de temperatura actual. Tiene el mismo principio que DATO2 sólo que aquí - llega el valor completo de 8 bits, y hay que separarlo de 4 en 4 para mandarlo a Display; primeramente se almacena en stack, después recorremos el número 4 posiciones a la derecha, con el fin de que quede en los 4 bits menos significativos para mandarlo a pantalla, enseguida cargamos nuevamente todo el número y lo enmascaramos, ANDA #$0F; los 4 bits más significativos desaparecen dejando así sólo los 4 bits menos significativos para mandarlo a Display.

Subrutina BDISP :

```
BDISP   PSHS   A
        LSRA
        LSRA
        LSRA
        LSRA
        BSR    ASCII
        LDA    ,S
        ANDA   #$0F
        BSR    ASCII
        PULS   A
        RTS
```

II.8. SUBRUTINA ALFA

Esta es una subrutina que nos sirve únicamente para mostrar mensajes en Display, los cuales deben estar ya incluídos dentro de otras subrutinas y previamente apuntados por el registro Y.

Es muy sencilla, sólo cargamos al registro -A- el carácter apuntado por el registro -Y-, incrementamos el apuntador y el carácter es mandado a Display. Esta secuencia se realiza hasta encontrar el carácter 03, que también previamente se introduce como parte del mensaje. Observar mensajes en subrutinas -RESET- y -PROCES-.

Subrutina ALFA :

```
ALFA    PSHS    A
FAL     LDA     ,Y+
        LBSR    DISPLA
        CMPA    #03
        BNE     FAL
        PULS    A
        RTS
```

II.9. SUBRUTINA DECHEX.

Esta subrutina sirve para convertir cualquier valor decimal a hexadecimal; es utilizada por la subrutina -PROCES-, que es la que se encarga del proceso de control. Como anteriormente se explicó, -DECHEX- fué creada para que el usuario pudiera teclear el dato en decimal y luego fuera convertido a hexadecimal, ya que si no se convierte, ese valor se tomará en hexadecimal, que es una cantidad mayor a la deseada.

El proceso de conversión consiste en multiplicar el número de las decenas o de los 4 bits más significativos por el número hexadecimal A, y al resultado sumarle el número de la posición menos significativa.

Este método resulta para cualquier valor numérico, por ejemplo, si queremos convertir 45 a hexadecimal, primero multiplicamos en hexadecimal 4*A - resultando 28, después sumamos a este resultado el número 5, también en hexadecimal, 28+5 = 2D, que es el resultado final.

Subrutina DECHEX :

```
DECHEX    PSHS    B
          LEAS    -1,S
          STA     ,S
          LDB     #$0A
          LSRA
          LSRA
          LSRA
          LSRA
          MUL
          LDA     ,S
          ANDA    #$0F
          STA     ,S
          ADDB    ,S
          TFR     B,A
          LEAS    1,S
          PULS    B
          RTS
```

Como necesitamos multiplicar el número que se encuentra en los 4 bits más significativos, primero lo almacenamos en el stack de máquina para no perderlo, cargamos al registro B el número hexadecimal A, recorremos el dato 4 veces a la derecha para dejarlo en la posición menos significativa y lo -

multiplicamos con el registro B, enseguida cargamos el número almacenado en el stack al registro -A- y eliminamos los bits más significativos, ANDA - #$0F, para luego sumarlo al registro B, que posee el resultado de la multiplicación, quedando así en el registro -A- el número ya convertido.

II.10. SUBRUTINA HEXDEC.

Esta subrutina convierte un valor de hexadecimal a decimal. Es usada- dentro de -MOSTRA-, con el fin de que los valores leídos del convertidor - A/D que están en hexadecimal sean convertidos a decimal para mandarlos a Display.

El método utilizado de conversión consiste en dividir el valor hexadecimal por el número hexadecimal A, inmediatamente después el resultado de - la división se posiciona en los 4 bits más significativos; el método concluye sumándole a ese valor el residuo de la división que se guardó cuando esta se realizó.

Como ejemplo de este método tenemos el número 2D, que lo convertiremos a decimal, primero lo dividimos por A, 2D/A, dando como resultado 4 y sobrando 5, recorriendo el 4 a la posición más significativa y sumándole el residuo 5 obtenemos el 45 decimal.

La subrutina HEXDEC es la siguiente:

```
HEXDEC   PSHS    B
         LEAS    -1,S
         LDB     #$0A
         BSR     DIV
         STB     ,S
         LSLA
         LSLA
         LSLA
         LSLA
         ADDA    ,S
         LEAS    1,S
         PULS    B
         RTS
```

Analizando la subrutina podemos observar que asignamos al registro B el valor HEX A, llamamos a DIV que divide dos números A/B, es decir el registro A, entre el registro B, así fue diseñada por eso se cargó en el registro B - el valor A; el valor que se va a convertir se encuentra en el registro A. - Después se recorre el resultado que es regresado en el registro -A- y le sumamos el residuo que se regresa en el registro B.

II.11. SUBRUTINA DIV.

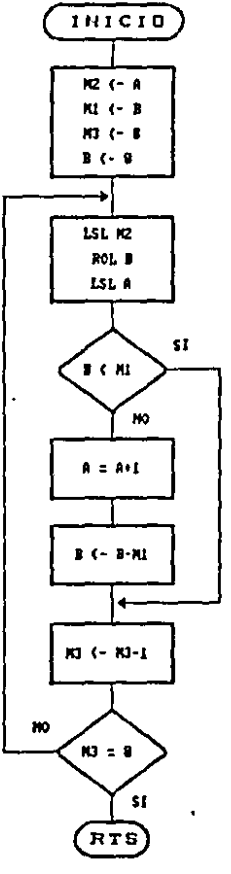
Dentro del lenguaje ensamblador 6809 encontramos entre otras la desventaja de carecer de división, a tal efecto de desarrollo la subrutina -DIV-, que nos divide dos números, el número almacenado en el registro -A- entre el número del registro B.

Antes de llamar a la subrutina -DIV- es necesario cargar al registro - -A- el número que se desea dividir, y al registro -B- el número por el cual va a dividirse o divisor. La operación de división se realizará como se realiza cualquier división en binario entregándose el resultado en el registro

-A- y el residuo en el registro B.

Analizando el diagrama de flujo que se encuentra enseguida podemos observar que inicialmente almacenamos los dos valores que llegan en A y en B, para poder utilizar estos registros dentro de la subrutina; son almacenados en el stack, pero los presentamos con una M que significa memoria. Almacenamos un 8 en otra memoria que nos servirá de contador, representando los 8 bits del número que va a dividirse. El método comienza recorriendo hacia la izquierda M2 que contiene al dividendo, LSL M2, para posicionar el bit más significativo en el bit de acarreo, realizamos después un ROL B, para introducir por la izquierda ese bit de acarreo al registro B, que será nuestro dividendo parcial, es decir que en este registro se irán introduciendo los bits de M2 con el fin de ir comparando este registro con M1 que posee al divisor y detectar hasta que momento el registro B es mayor a M1, osea que ya es divisible, después de estar introduciendo uno por uno los bits de M2 al registro B. Antes de entrar a comparar, si B>M1, se realiza un corrimiento del registro A, LSLA, introduciendo un cero en la izquierda, anticipándonos a que si la comparación B<M1 es cierta, es decir, B todavía no es divisible por M1, exista un cero en ese registro que será la respuesta final; el programa saltará después hasta el punto donde se decrementa M3; cada decremento indica que ya se han comparado N bits de los 8 que posee el dividendo; si la comparación B<M1 resulta falsa, lo cual dice que B es igual o más grande que M1 y por lo tando divisible, se incrementa el registro -A-, con lo cual se pone un uno en la posición menos significativa indicando que ya es divisible y toca a uno; se realiza

SUBRUTINA DIV.



después la resta B=B-M1, donde queda el residuo de esta división parcial, - para enseguida decrementar M3 y proseguir a introducir todos los bits de M2 en B y seguir comparando hasta el último, cuando M3=0. De esta forma el resultado queda en el registro -A- y el residuo en el registro B.

Subrutina DIV :

```
DIV     LEAS    -3,S
        STB     ,S
        STA     1,S
        LDA     #8
        STA     2,S
        CLRB
CICLO   LSL     1,S
        ROLB
        LSLA
        CMPB    ,S
        BLO     NOTOCA
        INCA
        SUBB    ,S
NOTOCA  DEC     2,S
        BNE     CICLO
        LEAS    3,S
        RTS
```

II.12 SUBRUTINA COM.

Para el manejo de las funciones del Display se creó -COM-. Todas las - funciones disponibles para el Display se encuentran en la tabla 6.1. del - Display de cristal líquido en el apéndice de esta tesis. Por lo que es necesario cargar al registro -A- el código de la instrucción deseada antes de - llamar a -COM- .

Analizando la subrutina encontramos que el código de la instrucción se

manda al puerto -B- del PIA 2, STA $A002, donde está conectada la línea de datos del Display. El dato enviado es necesario que se interprete como instrucción por lo que se debe poner a nivel bajo la terminal RS, selección de registro del Display que se encuentra conectada a la entrada PA2 del puerto A; de igual manera es necesario que mandemos a la terminal enable, conectada en PA0, un pulso para que el Display lea la instrucción y la ejecute; también se requiere poner a nivel bajo la terminal R/$\overline{W}$ del Display, conectada a PA1, para así escribir en él, estas funciones se logran mandando un 1 al puerto A, es decir en PA0 estará el 1, en PA1 y PA2 tendremos un 0 que es lo que queremos, en seguida decrementamos la dirección A000 -- con lo cual provocamos el pulso en la terminal enable del Display. Desde el momento del pulso y la ejecución de la instrucción el Display tarda poco más de un ciclo de máquina; para lo cual se estableció un pequeño ciclo de retardo que provoca que el microprocesador se dilate un poco para que el Display termine su función antes de mandarle otro dato. Por último mandamos un cuatro a la dirección A000 estableciendo un uno en PA2 que es RS del Display para dejarlo nuevamente como entrada de información a mostrar.

La subrutina COM es la siguiente:

```
COM     PSHS    B,A
        STA     $A002
        LDB     #1
        STB     $A000
        DEC     $A000
WAIT    INCB
        BNE     WAIT
        LDB     #4
        STB     $A000
        PULS    A,B
        RTS
```

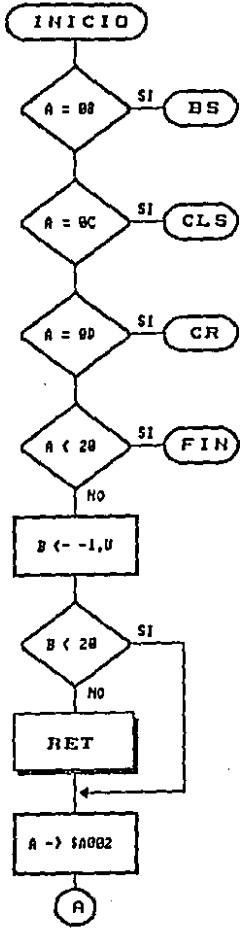
II.13. SUBRUTINA DISPLA.

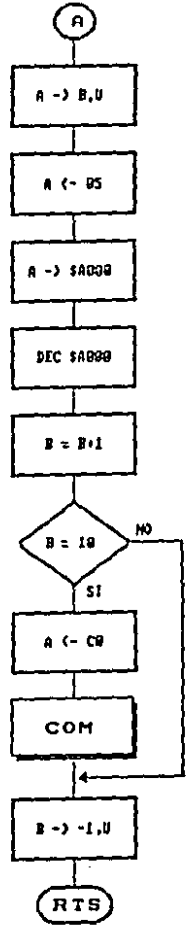
Esta subrutina es la más extensa de todas, ya que requiere de un gran cuidado para el manejo de la correcta visualización de los datos. Para la comprensión de esta subrutina, haremos su análisis basándonos en los diagramas de flujo que se situan a continuación.

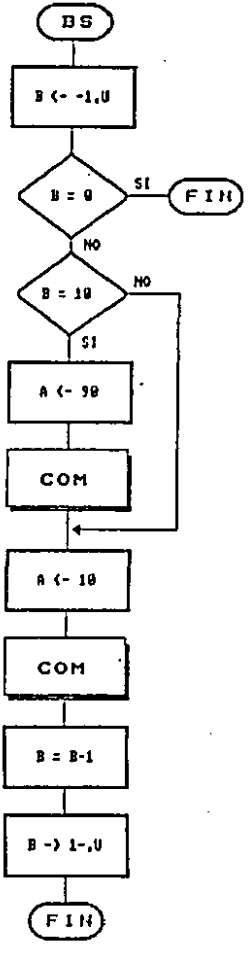
Antes de iniciar con el análisis tendremos en cuenta que el stack de usuario lo usaremos para almacenar todos los datos que se muestran, con motivo de poderlos utilizar nuevamente en otra línea si ocurre un retorno. - También tendremos reservado un lugar para tener la posición del Display donde actualmente se encuentra el cursor, -1,U que desde la subrutina -RESET- fué asignado.

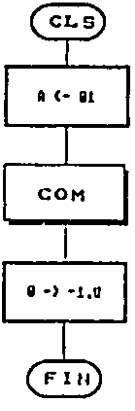
Inicialmente se compara el código que se recibe para determinar si corresponde a alguna función como CLS, limpiar pantalla, backspace, regresar un espacio, o un retorno; de lo contrario es un dato que va directamente a mostrarse y seguimos con la secuencia siempre y cuando sea un valor mayor - al 20 en hexadecimal o 33 en ASCII que corresponden a caracteres. Cargamos - enseguida al registro B la posición del cursor, si ya es igual o mayor a 20 en HEX. Que es la máxima localidad dentro de la pantalla; ya que el Display consta de dos líneas de 16 espacios, recurrimos a otra subrutina que es - -RET-, para ejecutar un retorno; -RET- pudo haber quedado dentro de -DIS - PLA-, pero para mayor comprensión es subrutina independiente; si el valor

SUBRUTINA DISPLAY.






BS
B <- -1,0
B = 0
SI
FIN
NO
B = 10
NO
SI
A <- 90
COM
A <- 10
COM
B = B-1
B -> 1-,0
FIN


CLS
A <- 01
COM
0 -> -1,0
FIN

de la posición del cursor es menor a 20 tenemos lugar todavía y se manda el dato a mostrar, STA $A002, enseguida lo almacenamos también en el stack y provocamos un pulso en la terminal enable para que el Display lea ese valor esto se logra mandando un 5 a la dirección A000 y decrementándola después;- con el número 5 se pone en 1 los bits PA0 y PA3, enable y RS; al decrementar la dirección queda un 4 en el puerto con lo que se manda el pulso al Display. Enseguida incrementamos B que es nuestro contador de posición del cursor, ya que se ha introducido un carácter más, y lo comparamos con el número HEX. 10, si es igual quiere decir que ya no tenemos lugar en la primer línea y tenemos que pasar el cursor al principio de la siguiente mandando el código C0 a -COM- el C0 es el código de la instrucción para posicionar el cursor en la primer posición de la segunda línea. Por último se almacena la posición del cursor en -1,U.

Si al inicio de la subrutina se detecta un 08 se generará un retorno de espacio, backspace. se inicia comparando la posición del cursor, si es cero no es posible ejecutarlo, si es igual al HEX. 10 es la última posición de la primer línea y mandamos a -COM- el código 90 que pone al cursor en este lugar, después se envía el código 10 a -COM- que es la instrucción para retorno de espacio; se decrementa en 1 la posición del cursor y se almacena.

Al recibirse un 0C se realiza un CLS, limpiar pantalla, simplemente mandamos el código 1 a -COM- que es el código de la instrucción CLS, enseguida se inicializa el contador que tiene la posición del cursor ya que después de un CLS el cursor se encuentra en la posición 0.

Por último, si se recibe un OD al inicio de esta subrutina se realiza un retorno llamando a -RET-, subrutina que se explica después.

Subrutina DISPLA :

```
DISPLA   PSHS    B,A
         CMPA    #$8
         BEQ     BS
         CMPA    #$0C
         BEQ     CLS
         CMPA    #$0D
         BEQ     CR
         CMPA    #$20
         BLO     DISFIN
         LDB     -1,U
         CMPB    #$20
         BLO     PANTA
         BSR     RET
PANTA    STA     $A002
         STA     B,U
         LDA     #$05
         STA     $A000
         DEC     $A000
         INCB
         CMPB    #$10
         BNE     FI
         LDA     #$C0
         BSR     COM
FI       STB     -1,U
         BRA     DISFIN
BS       LDB     -1,U
         BEQ     DISFIN
         CMPB    #$10
         BNE     ATRAS
         LDA     #$90
         BSR     COM
ATRAS    LDA     #$10
         BSR     COM
         DECB
         STB     -1,U
         BRA     DISFIN
CLS      LDA     #$1
         BSR     COM
         CLR     -1,U
         BRA     DISFIN
```

```
CR          BSR     RET
DISFIN      PULS    B,A
            RTS
```

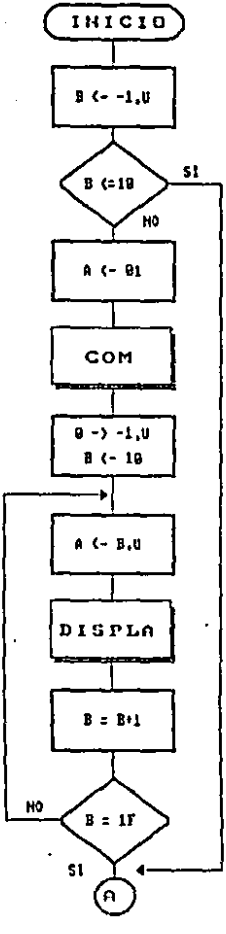
II.14 SUBRUTINA RET.

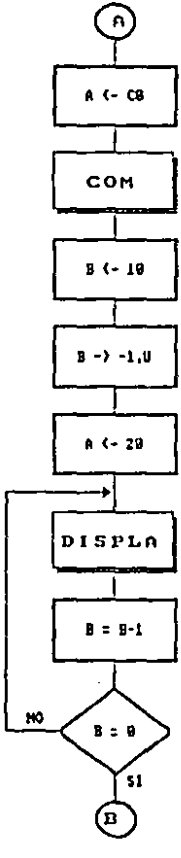
Como anteriormente se mencionó -RET- es parte de -DISPLA- por lo que pudo haber quedado integrada a ella, pero con motivo de seguir una secuencia de programación en módulos se elaboró independiente, como ha sido el caso de todas las subrutinas analizadas.

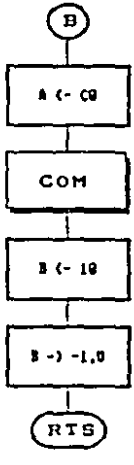
La función de esta subrutina es la de ejecutar un retorno, la estudiaremos atendiendo al diagrama de flujo. Lo primero que se realiza al llamar a -RET- es revisar donde se encuentra el cursor, B←-1,U, si es menor o igual al HEX. 10, es decir, la máxima posición de la primer línea, debemos posicionarnos al principio de la segunda enviando el comando CO a -COM-; enseguida almacenamos nuestra posición actual y borramos la segunda línea mandando a -DISPLA- el número HEX. 20, 32 en ASCII, que realiza un espacio a la derecha esto se efectúa hasta decrementar el registro B que ya se le había asignado el HEX.10. Después de borrar toda la segunda línea, el cursor se vuelve a posicionar al principio de ella para terminar.

Si la posición del cursor es mayor al HEX. 10, quiere decir que estamos en la segunda línea, con lo que ocupamos borrar la primera y pasar toda la segunda en su lugar, esto se logra borrando toda la pantalla con el código 1 a -COM-, enseguida mandamos a -DISPLA- todos los caracteres que tene -

SUBRUTINA RET.







mos almacenados desde la posición HEX. 10 del stack hasta el final, para lo cual se carga al registro B el número HEX. 10 que será nuestro punto de inicio para sacar los caracteres almacenados, A←B,U. Esta secuencia se realiza hasta que B=1F que indica la última posición del Display, o sea la última localidad dentro del stack donde se guardó información.

Subrutina RET :

```
RET       PSHS    A
          LDB     -1,U
          CMP     #$10
          BLS     BORRAR
          LDA     #1
          BSR     COM
          CLR     -1,U
          LDD     #$10
SCROLL    LDA     B,U
          BSR     DISPLA
          INCB
          CMPB    #$1F
          BNE     SCROLL
BORRAR    LDA     #$C0
          LBSR    COM
          LDB     #$10
          STB     -1,U
          LDA     #$20
R1        BSR     DISPLA
          DECB
          BNE     R1
          LDA     #$C0
          LBSR    COM
          LDB     #$10
          STB     -1,U
R2        PULS    A
          RTS
```

II.15, SUBRUTINA RETARD.

Es la más corta de todas las subrutinas, se utiliza únicamente para ha

cer tiempo y se basa en un ciclo que decrementa al número hexadecimal FFFF. A la derecha de la subrutina se encuentran los ciclos de máquina de cada instrucción.

```
RETARD   LDX    #$FFFF    3
CICLO    LEAX   -1,X      5
         BNE    CICLO     3
         RTS              5
```

Tomando en cuenta que se está trabajando a 1/4 de la frecuencia del cristal, es decir, 2.4576/4 = .6144 Mhz, y que la subrutina consta de 11 ciclos de máquina ya que no se toma en cuenta los ciclos de la instrucción RTS, por efectuarse sólo una vez, obtenemos un tiempo de subrutina (11)(1/.6144M) = 17.9us. Esta subrutina se ocupa para que tarde aprox. 1seg. Por lo que se decrementa el número FFFF que en decimal es 65635 , obteniendo así un tiempo total de (17.9us)(65635) = 1.17 seg.

II.16. SUBRUTINA MOSTRA.

-MOSTRA- es una subrutina que tiene la función de mostrar en Display la temperatura actual. El programa comienza mandando a la subrutina -DISPLA- dos veces el código 08 que corresponde a la función de regreso de espacio, backspace, con el fin de que siempre aparezca el valor en el mismo lugar; después se llama a -HEXDEC- para convertir a decimal el valor hexadecimal que entrega el convertidor A/D. Una vez convertido este valor se llama a -BDISP- para mandarlo a Display correctamente.

La subrutina es la siguiente:

```
MOSTRA   PSHS    A
         LDA     #8
         LBSR    DISPLA
         LBSR    DISPLA
         LDA     ,S
         LBSR    HEXDEC
         LBSR    BDISP
         PULS    A
         RTS
```

II.17 SUBRUTINA PROCES.

-PROCES- es la subrutina más importante ya que se encarga de efectuar correctamente el proceso de control con auxilio de todas las subrutinas anteriores.

Recordando la función del control; el usuario introduce el valor de la temperatura máxima que es el punto de desición del sistema, inmediatamente después aparecerá en Display el valor de esta temperatura más el valor de la temperatura actual, y el proceso empezará a funcionar, es decir, mientras la temperatura actual no iguale a la máxima establecida no se activará la señal de control, en caso contrario se activará esta señal y se mantendrá el tiempo necesario hasta que de nuevo la temperatura actual baje del valor establecido.

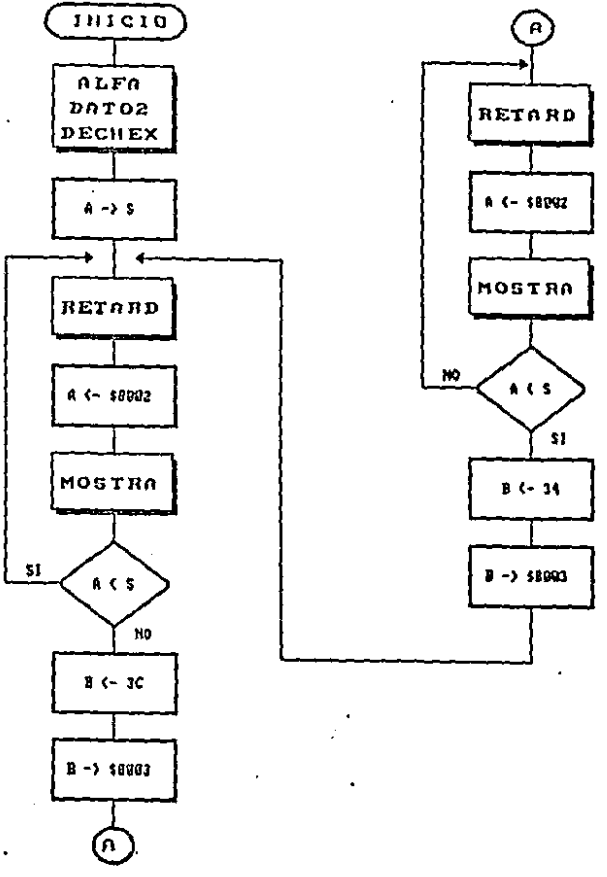
Este es un control porque desde luego la señal activaría algún dispositivo que provocara el descenso de la temperatura, con lo cual estamos controlándola; el dispositivo de enfriamiento no se incluye en el diseño de este control por lo que no se menciona en ningún momento.

Una vez teniendo correctos todos los anteriores modulos de programación la subrutina -PROCES- puede tener infinidad de variantes, es decir, el proceso puede ser tan diferente como las necesidades lo requieran, pueden existir más señales de control tratando independiente a cada una, o pueden recibirse varias otras de gran cantidad de dispositivos sensores, en fin es esto lo - que hace al microporcesador superior a cualquier circuito lógico, ya que solamente modificando el módulo de desición tendremos un nuevo control tomando en cuenta que todos los módulos ya existentes no necesitan cambio; ésta es - la razón por la que todas las subrutinas fueron independientes. Para nuestro caso es un proceso sencillo donde sólo manejamos una sola señal de control, pero ilustrativo para cualquier otra aplicación.

Entrando en el análisis de esta subrutina haremos referencia al diagrama de flujo que se encuentra después.

Iniciamos llamando a -ALFA- para que aparezcan los mensajes que están al final de la subrutina; seguimos con -DATO2- para introducir el valor de temperatura máxima, inmediatamente después ese valor se manda a -DECHEX- - para trabajarlo en hexadecimal y se almacena en el stack de máquina. Sigue un ciclo donde se compara la temperatura actual con la máxima, primero se llama a -RETARD- para hacer tiempo de 1.17 seg, ya que el convertidor entrega una conversión nueva aprox. cada 2 seg., por lo que no necesitamos que - el microprocesador esté leyendo el puerto todo ese tiempo; después de -- -RETARD- se lee el puerto donde está el convertidor, y ese valor es enviado a -MOSTRA- para que aparezca en Display, enseguida se realiza la comparación,

SUBRUTINA PROCES.



A<S, si la temperatura actual es menor que la máxima, la cual está almacenada en el stack se entra nuevamente al ciclo; de lo contrario se carga al registro B el número 3C y lo mandamos a la dirección $8003 para poner en uno el bit 4 del registro de control B del PIA 1, que es el bit de puerto de la terminal de control CB2, activándose la señal de control; entramos después a otro ciclo semejante al anterior; si la temperatura actual es igual o mayor a la máxima se continua en este segundo ciclo, donde la señal de control permanece activa hasta salir de él; si la temperatura actual ya es menor cargamos al registro B el número 34 y lo mandamos a la dirección $8003 con esto ponemos en cero la terminal CB2 desactivando la señal de control. Para terminar se regresa a iniciar el primer ciclo. Esta subrutina permanece funcionando hasta que se recibe una señal de reestablecimiento.

Subrutina PROCES :

```
PROCES    LEAY    MSG1,PCR
          LBSR    ALFA
          LBSR    DATO2
          LBSR    DECHEX
          STA     ,S
          LEAY    MSG2,PCR
          LBSR    ALFA
LECT1     LBSR    RETARD
          LDA     $8002
          LBSR    MOSTRA
          CMPA    ,S
          BLO     LECT1
          LDB     #$3C
          STB     $8003
LECT2     LBSR    RETARD
          LDA     $8002
          LBSR    MOSTRA
          CMPA    ,S
          BHS     LECT2
          LDB     #$34
          STB     $8003
          BRA     LECT1
```

```
MSG1    FCB     $0D
        FCC     \TEMP MAX_\
        FCB     03
MSG2    FCB     $0D
        FCC     \TEMP ACT____\
        FCB     03
```

Para grabar todas estas subrutinas en la EPROM primero es necesario - entrar a un sistema que contenga el ensamblador 6809, introducimos todas las subrutinas correctamente en el orden que sea, es decir, -RESET- puede estar al principio, al centro o al final, igual las demás subrutinas, en - samblamos el programa y queda listo para grabarlo a la EPROM, desde luego- mediante un grabador de memorias EPROM; recordando establecer en FFFE y FF FF de la EPROM la dirección exacta donde se encuentra el inicio de la su - brutina -RESET-.

CONCLUSIONES

A través de este trabajo de tesis se puede observar como el microprocesador juega un papel muy importante en el diseño de controladores, presentando grandes ventajas sobre técnicas de diseño basadas en otros dispositivos, como transistores, amplificadores operacionales, sistemas digitales; entre muchas otras encontramos que con la utilización del microprocesador se pueden construir controladores bastante complicados como los controles PROPORCIONAL-INTEGRAL-DERIVATIVO, (PID), algo demasiado difícil con otros métodos, debido a toda una serie de cálculos matemáticos que se realizan para obtener una óptima respuesta del sistema, resultando con ello más rápidos, exactos y precisos. Aunque este trabajo fue un control todo o nada, que en teoría es más fácil que el control PID, se pudo observar la gran cantidad de cálculos que se realizaron para la elaboración del proceso, así como toda una serie de estrategias para el correcto manejo de la información del teclado y Display.

Este controlador sólo manejó una variable, temperatura, pero se puede trabajar con una gran variedad de ellas, así como de transductores y señales de control, tantas como se requieran.

Es importante hacer notar que una gran ventaja con la utilización del microprocesador, por más simple que su aplicación sea, es la versatilidad y rapidez para modificar la secuencia de control que se esté llevando a cabo con el simple hecho de modificar la subrutina que se encarga del proceso de control, tomando en cuenta que todas las demás subrutinas de soporte están -

correctas, algo mucho más sencillo que el diseño de una nueva etapa que se le adaptará a un circuito de sistemas digitales por cada variación en las necesidades del control.

BIBLIOGRAFIA

1.- 8-BIT MICROPROCESORS AND PERIPHERAL DATA.
MOTOROLA INC., 1983
CAPITULO 3, DATA SHEETS.
PP. 233-265, 307-316.

2.- MEMORY COMPONENTS HANDBOOK.
INTEL CORPORATION, 1989.
CAPITULO 4, EPROMS.
PP. 1-5

3.- RONLD J. TOCCI.
SISTEMAS DIGITALES, TERCERA EDICION.
PRENTICE HALL, 1987.
CAPITULO 9, CIRCUITOS LOGICOS MSI.
PP. 380-382.

4.- LINEAR DATA BOOK 2.
NATIONAL SEMICONDUCTOR CORPORATION, 1988.
CAPITULO 3, ANALOG-TO-DIGITAL CONVERTERS.
PP. 16, 28-34.
CAPITULO 6, TEMPERATURE SENSORS.
PP. 12-15

5.- ROBERT F. COUGHLIN / FREDERICK F. DRISCOLL.
CIRCUITOS INTEGRADOS LINEALES
Y AMPLIFICADORES OPERACIONALES, SEGUNDA EDICION.
PRENTICE HALL.
CAPITULO 3, AMPLIFICADORES INVERSORES Y NO INVERSORES.
PP. 42-45
CAPITULO 8, AMPLIFICADORES DIFERENCIAL,
DE INSTRUMENTACION Y PUENTE.
PP. 162-164

6.- TIMOTHY J. MALONEY
ELECTRONICA INDUSTRIAL, DISPOSITIVOS Y SISTEMAS.
PRENTICE HALL, 1988.
CAPITULO 9, SISTEMAS REALIMENTADOS Y SERVO MECANISMOS.
PP. 302-320

7.- LANCE A. LEVENTHAL.
6809 ASSEMBLY LANGUAGE PROGRAMMING
Mc GRAW-HILL, 1981.

8.- MICROCOMPUTER HARDWARD HANDBOOK.
ELCOMP, 1982, GERMANY
PP. 205-209

APENDICE

Descripción de los dispositivos utilizados en los circuitos CONTROLADOR y SENSOR-CONVERTIDOR.

I. MICROPROCESADOR 6809.

Considerado por muchos como el mejor microprocesador de 8 bits, es un híbrido ya que permite el manejo de registros de 16 bits con instrucciones poderosas.

El 6809 es un desarrollo nacido del 6800. Sin embargo no son compatibles conexión por conexión y presentan algunas diferencias en su arquitectura. Esto es debido a que la filosofía de los diseñadores del 6809 fué mejorar la capacidad del 6800 pero manteniendo compatibilidad con el sistema 6800. Como consecuencia el conjunto de instrucciones es más poderoso, por la utilización de nuevas formas de direccionamiento y la optimización de los códigos de operación manteniendo en su mayoría las diferencias del ensamblador 6800.

I.1. ARQUITECTURA.

Específicamente las características en arquitectura del 6809 son las siguientes:

-Dos acumuladores de 8 bits, que juntos forman uno de 16.

-Dos registros índice de 16 bits.

-Dos apuntadores stack con capacidad de indexado.

-Registro de página directa.

-59 Mnemónicos de instrucciones.

-1464 instrucciones en diferentes modos de direccionamiento (inherente, directo, extendido, inmediato, indexado, indirecto, relativo).

-Multiplicación sin signo de 8*8 bits.

-Aritmética de 16 bits (cargar, almacenar, sumar, restar y comparar).

-Instrucciones poderosas para el manejo del stack.

-Transferencia e intercambio de registros.

-Instrucciones de manipulación de direcciones.

-Brincos de rango extendido.

En la figura 1.1. se muestra el modelo de los registros de programación. Los registros apuntadores U,S son de 16 bits al igual que los registros índice X,Y. El registro acumulador D es de 16 bits y está formado por la unión de los acumuladores A y B. Estos tres acumuladores están ligados con la ejecución de operaciones lógicas y aritméticas. El registro de página directa y el de código de condición de 8 bits sirven de apoyo a la programación.

Los registros índice son usados en el direccionamiento indexado. La dirección de 16 bits contenida en ambos registros X,Y es usada para apuntar a un dato directamente, o puede ser modificada por una constante o registro llamado offset. Los registros X,Y son equivalentes en uso, por lo que tienen el mismo número de instrucciones.

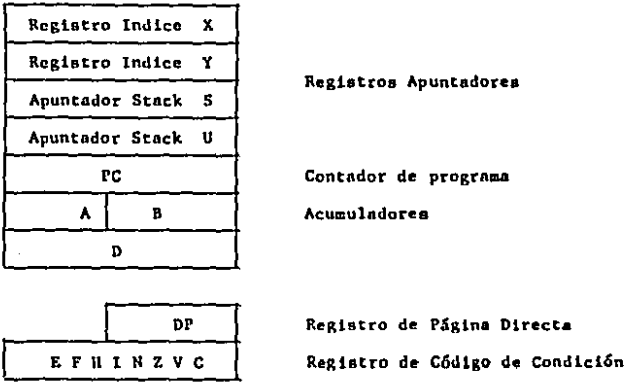


Figura I.1.- Registros de programación.

Los registros apuntadores U,S también pueden ser usados como registros - índice, pero el apuntador S es usado por el microprocesador para formar el - llamado stack de máquina. El apuntador U es para el uso del programador y permite transferir argumentos de y hacia subrutinas con facilidad. Estos apuntadores tienen el mismo número de instrucciones que los registros X,Y más las instrucciones de control PUSH y PULL.

El siguiente registro es el contador de programa (PC), es de 16 bits y es usado por el micro para indicar la dirección de la siguiente instrucción a ejecutar. El direccionamiento relativo permite usar el PC como un registro índice en algunas situaciones.

Los registros A, B son acumuladores de propósito general usados para - cálculos aritméticos y para movimiento de datos de una palabra. La unión de A y B forma el acumulador D de 16 bits, donde A es la parte más significativa y B la menos significativa.

El registro de página directa (DP) define la palabra más significativa - usada en el direccionamiento directo. El registro DP está concatenado con la palabra que sigue al código de operación de direccionamiento directo para formar la dirección efectiva. Esto permite el uso del direccionamiento directo en cualquier lugar de las 64K palabras de memoria.

El último registro es el de código de condición. El formato de este re - gistro se muestra a continuación.

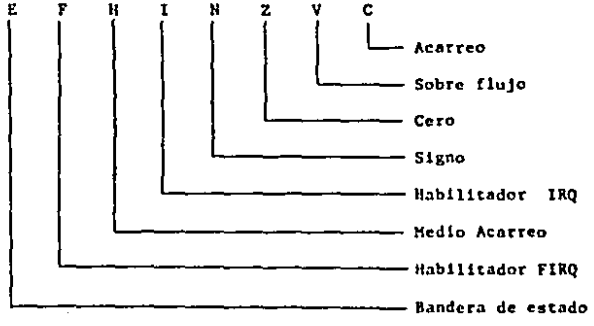


Figura 1.2. registro del código de condición.

Bit 0 (C) este bit representa usualmente un acarreo de la unidad lógica y aritmética (ALU). Específicamente C es puesto en uno por el acarreo binario del bit más significativo (MSB) de las operaciones de suma (ADC y ADD). También es usado para representar un préstamo (Borrow) de una instrucción de substracción (CMP, NEG, SUB, SBC). Sólo las operaciones aritméticas modifican a C.

Bit 1 (v). Este bit es puesto en uno por la operación que cause un sobreflujo aritmético de 2 complemento. (operaciones con signo).

Bit 2 (Z). Este bit es puesto en uno si el resultado de la operación previa es igual a cero.

Bit 3 (N). Este bit indica el signo y contiene el valor del MSB del resultado de la operación previa. Si ocurre un sobreflujo de 2 complemento, el signo del resultado será incorrecto.

Bit 4 (I). Este es el bit de condicionamiento de la interrupción condicionada (IRQ). Si este bit está en 1, el microprocesador ignora cualquier interrupción IRQ. Las interrupciones NMI, FIRQ, RESET y SWI pondrán este bit en 1 mientras que SWI2 y SWI3 no afectan a 1.

Bit 5 (H). Este bit es usado para indicar el acarreo del bit 3 en las operaciones de suma de 8 bits (ADC y ADD). Esta indicación es necesaria para la instrucción DAA (ajuste decimal) en las sumas BCD. El estado de este bit

está indefinido en cualquier substracción.

Bit 6 (F). El bit F está asociado con la interrupción rápida condiciona da (FIRQ). Si este bit está en 1 el microprocesador ignorará las instrucciones que se presenten en la línea FIRQ. Las interrupciones NMI, FIRQ, SWI y RESET pondrán a F en 1. IRQ, SWI2 y SWI3 no afectan a F.

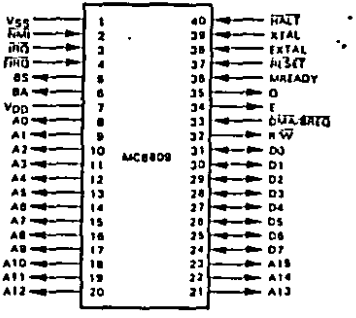
Bit 7 (E). Este bit es reservado para indicar el tipo de interrupción - que se trabaja. El bit E es usado por la instrucción retorno de interrupción (RTI) para determinar el número de los datos guardados en el stack. Esta función permite el manejo de las subrutinas de interrupción que trabajan con interrupciones lentas y rápidas. La interrupción FIRQ pondrá en cero a E, mientras que IRQ, NMI, SWI2 y SWI3 pondrán a E en 1.

1.2. DESCRIPCION DE SEÑALES.

En la figura 1.3. se muestra la distribución de terminales del 6809 a la cual se hará referencia a continuación.

Alimentación (Vss,Vcc), contactos 1 y 7. Vss es tierra, o cero volts, mientras que Vcc es + 5 volts con una tolerancia del 5%.

Extal, Xtal contactos 38, 39. Estos contactos de entrada son usados para conectar un oscilador interno con un cristal resonante-paralelo. El contacto extal puede ser usado como una entrada TTL para un reloj exterior, co-



Pin Name	Description	Type
*A0-A15	Address Lines	Tri-state, Output
*D0-D7	Data Bus Lines	Tri-state, Bi-directional
*E, Q	Clock Signals	Output
*R/W	Read/Write	Tri-state, Output
*BA	Bus Available	Output
*BS	Bus State	Output
EXTAL, XTAL	Crystal	Input
*MRDY	Memory Ready	Input
*DMA/BREQ	DMA/Bus Request	Input
*HALT	Halt	Input
*RESET	Reset	Input
NMI	Non-Maskable Interrupt	Input
*FIRQ	Fast Interrupt Request	Input
*IRQ	Interrupt Request	Input
V_{DD}, V_{SS}	Power and Ground	

*These signals connect to the System Bus

FIGURA I.3. TERMINALES 6809.

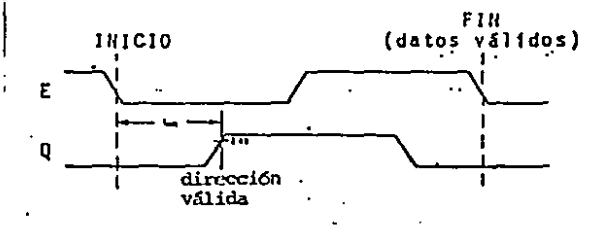


FIGURA I.4. SEÑALES DE RELOJ.

nectando a tierra xtal. Ambos, el cristal o la señal externa, debe ser cuatro veces la frecuencia del reloj que se desee.

E, Q contactos 34, 35. E es una señal de sincronización a la frecuencia de operación y se conecta en la mayoría de las aplicaciones a la entrada habilitadora de la familia 68XX de periféricos.

Tal como se puede apreciar en la figura 1.4. la salida Q es una señal de cuadratura (90°) de E. El inicio de este pulso indica que la dirección presente en las líneas de direccionamiento, es correcta y estable. El fin de este pulso indica que la información presente en las líneas de datos es correcta y estable. Los datos son atrapados al final del pulso E.

Línea de datos (D7-D0) contactos 24 a 31. Los 8 contactos, designados para los datos, proporcionan la comunicación con el sistema bidireccional de datos. Cada contacto puede manejar una carga TTL schottky y típicamente 130 pF.

Línea disponible, estado de línea (BA,BS) contactos 5 y 6. La salida línea disponible (BA) es una indicación de la señal de control interno que pone los acopladores de las líneas del microprocesador en alta impedancia.

La presencia de esta señal no implica que las líneas estarán disponibles por más de un ciclo. Para saber el estado del microprocesador es necesario tomar en cuenta además la salida de estado de línea (BS, BUS STATUS). El estado del microprocesador será entonces indicado por la decodificación de las

líneas BA y BS como se muestra a continuación. Esta decodificación es valida al final del pulso Q.

Estado del microprocesador.

BA	BS	Interpretación
0	0	Normal (corriendo)
0	1	Reconocimiento de interrupción
1	0	Reconocimiento de SYNC
1	1	Alto o líneas en disposición

Lectura/escritura (R/$\overline{W}$) contacto 32. Esta señal indica la dirección de transferencia en las líneas de datos. Un nivel bajo indica que el microprocesador está "escribiendo" un dato en la línea de datos. La salida R/$\overline{W}$ es válida en el inicio de la señal Q.

Reestablecer ($\overline{RESET}$) contacto 37. Esta entrada está acoplada internamente a un disparador SCHMITT (TRIGGER) en donde una señal de bajo nivel que dure más de un ciclo de reloj reestablecerá al microprocesador.

El vector (dirección) de reestablecimiento, es buscado en los lugares - FFFE y FFFF cuando el reconocimiento de la interrupción sea cierta (BA*BS=1) mientras que se estabiliza la energía al ser aplicada inicialmente, esta línea deberá mantenerse en nivel bajo hasta que el oscilador del reloj esté - funcionando completamente.

Debido a que el contacto de reestablecimiento tiene una entrada disparadora SCHMITT de un voltaje de umbral mayor que el de los periféricos stándar, - se puede usar un simple circuito R-C para reestablecer el circuito completo. Este voltaje de umbral alto, nos asegura que los periféricos dejarán el estado de reestablecimiento antes que el microprocesador.

MRDY, contacto 36. Esta terminal de entrada de control permite ampliarla duración del pulso E para extender el tiempo de acceso de datos. Cuando - la terminal MRDY está en el estado alto, la señal E estará en operación normal; cuando tenga nivel lógico cero, el pulso de reloj E puede ser extendido en múltiplos de un cuarto de ciclo de línea, permitiendo así acoplarse con - memorias de tiempo de acceso largo. El máximo de duración del pulso E es de 10uS. Durante un acceso de duración E. Esto impide que la velocidad del microprocesador disminuya durante accesos de línea sin importancia.

$\overline{DMA}/\overline{BREQ}$, contacto 33. La entrada $\overline{DMA}/\overline{BREQ}$, se utiliza para suspender - la ejecución y tener disponibles las líneas del microprocesador para otro - uso.

Sus usos típicos son el direccionamiento directo de memorias (DMA). y para refrescar memorias dinámicas.

Línea de direcciones (A0-A15) contactos 8 a 23. Se usan 16 contactos de salida para dar la información de direcciones en la línea de direcciones. -

Cuando el microprocesador no requiere esta línea, Para manipular datos, la salida de direcciones será FFFF, R/$\overline{W}$=1 y BS=0. La dirección es válida en - el inicio del pulso Q. Todos los acopladores de línea de direcciones se ponen en alta impedancia cuando la salida BA está en estado 1. Cada línea puede manejar una carga TTL schottky y típicamente 90 pF.

Alto ($\overline{HALT}$) contacto 40. Un nivel bajo en este contacto de entrada causará que el uP pare de correr al final de la instrucción que esté ejecutando, y permanezca en alto indefinidamente, sin pérdida de datos. Cuando esto sucede, la salida BA tiene nivel lógico alto indicando que las líneas se encuentran en alta impedancia; la salida BS también se encuentra en nivel alto indicando que el uP está en el estado de alto o de líneas en disposición. Mientras el uP esté en alto, no responderá a peticiones de interrupción en tiempo real (FIRQ, IRQ). Sin embargo, si aceptará a $\overline{DMA}/\overline{BREQ}$ y si se activará $\overline{NMI}$ o $\overline{RESET}$, se guardarán para una respuesta posterior. Durante el estado de alto las salidas Q y E continuan trabajando normalmente.

Interrupción incondicionada ($\overline{NMI}$) contacto 2. La interrupción incondicionada se provoca cuando existe una transición negativa en el contacto de entrada 2. Cuando es reconocida, el estado de máquina completo es guardado en una parte especial de la memoria llamada stack. Esta interrupción no puede ser inhibida por el programa y tiene mayor prioridad que $\overline{IRQ}$, $\overline{FIRQ}$ o - cualquiera de las interrupciones del programa. Sin embargo, si el microprocesador está en reestablecimiento la $\overline{NMI}$ no es reconocida hasta que el primer programa indique la ubicación del stack. La duración del nivel lógico

cero debe ser cuando menos un ciclo E. El programa de interrupción se inicia a partir de la dirección contenida en las memorias FFFC y FFFD.

Interrupción condicionada ($\overline{IRQ}$) contacto 3. Cuando esta línea es puesta a nivel lógico 0 por algún dispositivo externo, y si el bit (I) del registro de código de condición está en 0, el microprocesador completará la ejecución de la instrucción que está realizando e inicia una secuencia de interrupción.

Cuando se inicia la secuencia de interrupción se almacenan en el stack los registros índice, el contador de programa, los acumuladores y los registros de código de condición y página directa. El bit (I) del registro de condición se pone en uno para que no ocurran interrupciones posteriores ($\overline{IRQ}$). Hasta que termine la presente interrupción. El programa de interrupción IRQ inicia a partir de la dirección marcada en las memorias FFF8 y FFF9. Al encontrar la instrucción RTI en el programa de interrupción, el uP regresará a su estado inicial.

Interrupción rápida condicionada ($\overline{FIRQ}$) contacto 4. Un nivel bajo en este contacto iniciará una secuencia de interrupción rápida si el bit (F) del registro de condición está en cero. $\overline{FIRQ}$ tiene prioridad sobre $\overline{IRQ}$ y es más rápida que ésta en el sentido de que sólo almacena el contenido del registro de condición y el contador del programa. El programa de interrupción se inicia a partir de la dirección marcada en las memorias FFF6 y FFF7 y deberá eliminar el origen de la interrupción antes de ejecutar la instrucción RTI (retorno de interrupción).

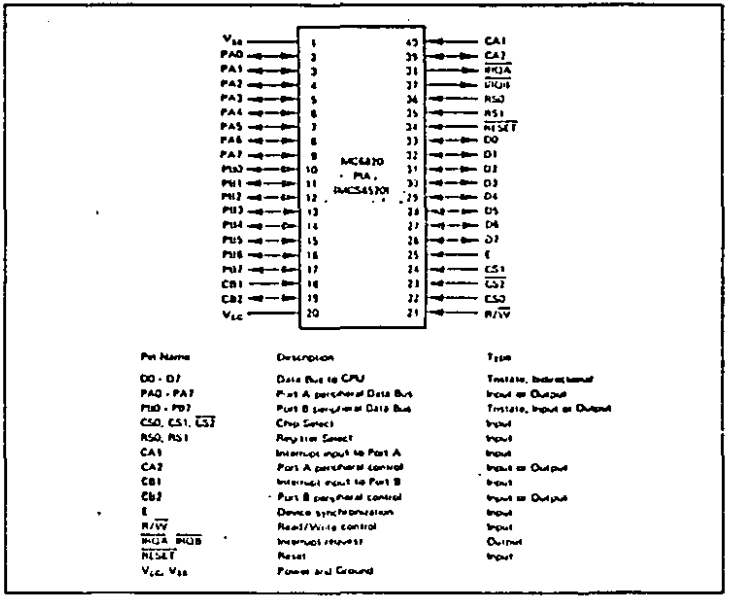
2. ADAPTADOR PERIFERICO 6821.

El adaptador periférico (PIA) 6820 o 6821 es capaz de proveer la interface necesaria entre el microprocesador (uP) y los periféricos a través de dos líneas de datos bidireccionales de 8 bits y 4 líneas de control.

La configuración funcional del PIA es programada por el uP durante la inicialización. Cada uno de los contactos de las líneas de datos periféricos puede ser programado para actuar como salida o como entrada, y cada una de las 4 líneas de control pueden ser programadas para uno de los varios modos de control. Esto permite un alto grado de flexibilidad en la operación como interface.

En la figura 2.1. se muestra el diagrama de la distribución de terminales del adaptador periférico 6821.

Controles internos. Existen 6 lugares dentro del PIA accesibles a las líneas de datos del uP; dos registros periféricos, dos registros de dirección de datos y dos registros de control. La selección de estos registros está controlada por las entradas RS0 y RS1 junto con el bit 2 del registro de control como se muestra en la tabla 2.1.



Pin Name	Description	Type
D0 - D7	Data Bus to CPU	Tristate, bidirectional
PA0 - PA7	Port A peripheral Data Bus	Input or Output
PB0 - PB7	Port B peripheral Data Bus	Tristate, Input or Output
CS0, CS1, $\overline{CS2}$	Chip Select	Input
RS0, RS1	Register Select	Input
CA1	Interrupt input to Port A	Input
CA2	Port A peripheral control	Input or Output
CB1	Interrupt input to Port B	Input
CB2	Port B peripheral control	Input or Output
E	Device synchronization	Input
R/$\overline{W}$	Read/Write control	Input
$\overline{IRQA}$, $\overline{IRQB}$	Interrupt request	Output
$\overline{RESET}$	Reset	Input
V_{CC}, V_{SS}	Power and Ground	

FIGURA 2.1. TERMINALES 6820.

RS1	RS0	CRA-2	CRB-2	Lugar Seleccionado
0	0	1	X	Registro Periférico A
0	0	0	X	Reg. de Dirección de Datos A
0	1	X	X	Registro de Control A
1	0	X	1	Registro Periférico B
1	0	X	0	Reg. de Dirección de Datos B
1	1	X	X	Registro de Control B

X = Sin importancia
CRA-2 = Bit 2 del Registro de Control A
CRB-2 = Bit 2 del Registro de Control B

Tabla 2.1. Direccionamiento interno.

Un nivel bajo en la línea $\overline{\text{RESET}}$ pone en ceros todos los registros. Esto significa que los contactos PA0-PA7, PB0-PB7, CA2 y CB2 están como entradas y todas las interrupciones deshabilitadas.

La configuración funcional del PIA deberá establecerse en el programa que sigue cuando la línea $\overline{\text{RESET}}$ cambia a nivel alto.

Registros de dirección de datos (DDRA y DDRB). Estos dos registros permiten al uP controlar la dirección de datos através de cada contacto periférico de datos. Un bit puesto en cero en el registro de dirección de datos pone el correspondiente contacto periférico como entrada; puesto en uno resulta como salida.

Registros de control (CRA y CRB). Estos registros permiten al uP controlar la operación de las 4 líneas de control periféricas CA1, CA2, CB1, CB2. Además permiten al uP habilitar las líneas de interrupción y monitorear el estado de las banderas de interrupción. Los bit del 0 al 5 en ambos registros pueden ser escritos o leídos por el uP cuando se aplica la dirección adecuada. Los bit 6 y 7 de los 2 registros sólo se pueden leer y son modificados por interrupciones externas en las líneas de control CA1, CA2, CB1, CB2. El formato de los registros de control se muestra en la tabla 2.2.

	7	6	5 4 3	2	1 0
CRA	IRQA1	IRQA2	Control de CA2	Acceso a DDRA	Control de CA1
CRB	IRQB1	IRQB2	Control de CB2	Acceso a DDRB	Control de CB1

Tabla 2.2. Formato del Registro de Control.

Bit de acceso a los registros de dirección de datos (CRA-2 y CRB-2).- el bit 2 en cada registro de control permite la selección del registro periférico o el registro de dirección de datos cuando se aplica la selección apropiada en RS0 y RS1.

Banderas de interrupción (CRA-6, CRA-7, CRB-6, CRB-7). Estas banderas de interrupción son activadas por transiciones activas de señales en las 4 líneas de control periféricas cuando están programadas para funcionar como entradas. Estos bits no pueden ser activados directamente por el uP y son

puestos en cero indirectamente por una instrucción de lectura de periférico en la sección apropiada.

Control de las líneas de entrada de interrupción CA1 y CB1 (CRA-0, - CRA-1, CRB-0 y CRB-1). Los dos bits menos significativos del registro de control son usados para controlar las líneas de entrada de control (interrupción) CA1 y CB1. Los bits CRA-0 y CRB-0 son usados para habilitar las señales de interrupción en IRQA e IRQB, respectivamente. Los últimos bits CRA-1 y CRB-1 determinan la transición activa de la señal de entrada CA1- y CB1.

CRA-1 (CRB-1)	CRA-0 (CRB-0)	Entrada de Interrupción CA1 (CB1)	Bandera de Interrupción CRA-7 (CRB-7)	Línea IRQA (IRQB)
0	0	↓ Activa	A 1 en la ↓	Permanece en alto.
0	1	↓ Activa	A 1 en la ↓	Puesta en nivel bajo.
1	0	↑ Activa	A 1 en la ↑	Permanece en alto.
1	1	↑ Activa	A 1 en la ↑	Puesta en nivel bajo.

Tabla 2.3. Control de las entradas CA1 y CB1.

Notas: 1.- ↑ Indica transición positiva.

2.- ↓ Indica transición negativa.

3.- La bandera de interrupción bit CRA-7 es puesto en cero por la lectura del uP del registro periférico A; CRB-7 es puesto en cero por la lectura del uP al registro B.

Control de las líneas periféricas CA2 y CB2 (CRA-3, CRA-4, CRA-5, CRB-3, CRB-4, CRB-5). Los bits 3, 4 y 5 de los registros de control son usados para determinar el modo de operación de las líneas CA2 y CB2. Estos bits indican si las líneas de control serán entradas de interrupciones o salidas de control. Si el bit CRA-5 tiene nivel bajo, CA2 (CB2) es una entrada de interrupción exactamente igual a CA1 (CB1); con los bits CRA-4 (CRB-4) y CRA-3 (CRB-3) respectivamente en la tabla 2.3.

Cuando CRA-5 (CRB-5) está en alto, CA2 (CB2) se torna en una salida que puede usarse para controlar periféricos. Cuando se tiene esta condición, CA2 y CB2 tienen algunas diferencias que pueden observarse en las tablas 2.4 y 2.5.

CRB5	CRB4	CRB3	Puesto en cero	Puesto en uno
1	0	0	En el inicio del primer pulso E que siga a una instrucción de escritura en el periférico B del microprocesador.	Cuando el bit CRB-7 es puesto en 1
1	0	1		En el inicio del primer pulso E después de la desactivación.
1	1	0	Cuando CRB-3 se pone en 0 como resultado de la escritura del uP en el registro de control B.	Cuando en una instrucción de escri - del uP al registro de control B se cambie el bit CRB-3 a 1.
1	1	1		

Tabla 2.4. Control de CB2 como salida.

CRA5	CRA4	CRA3	Puesto en cero	Puesto en uno
1	0	0	En la transición negativa del primer pulso E que siga a una instrucción de lectura del registro periférico A.	Cuando el bit CRB-7 es puesto en 1.
1	0	1		En la transición negativa de E durante la desactivación.
1	1	0	Cuando CRA-3 se pone en nivel bajo como resultado de una instrucción de escritura del uP al registro de control A.	Cuando una instrucción de escritura del uP al registro de control A pone el bit CRA-3 en 1.
1	1	1		

Tabla 2.5. Control de CA2 como salida.

Para la utilización del PIA es necesario realizar la programación adecuada del mismo en el procedimiento de inicialización que sigue al RESET en todo microcomputador.

Para la programación del PIA es necesario que los registros de control se encuentren en ceros. Esto se da automáticamente con el RESET de Hardware. Sin embargo si se va a reinicializar se debe limpiar a los registros de control (dirección IMPAR).

En este punto las direcciones pares accesan al registro de dirección.- por estar el bit 2 del registro de control en cero, y se debe programar la dirección de trabajo, Es decir, se deberá escribir "UNOS" a los bits que se deseen como salidas y "CEROS" a los que funcionarán como entradas.

Seguidamente se programan las direcciones impares (registro de control)

detallando la forma en que trabajarán las líneas de control CX1 (bits 0 y - 1) y CX2 (bits 3, 4 y 5) y poniendo el bit 2 a uno para que los accesos posteriores a las direcciones pares accesen al registro periférico, ya sea como entrada o salida según se programó.

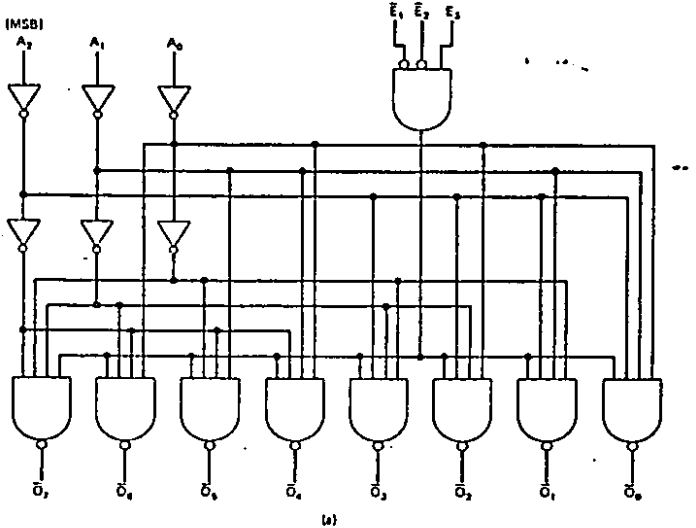
3. DECODIFICADOR 74138.

La figura 3.1. muestra el diagrama lógico del decodificador 74LS138 - tal y como aparece en el manual TTL de Fairchild.

Si se examina este diagrama cuidadosamente podemos llegar a determi - nar exactamente la forma en que este decodificador 74138 funciona. Primero, obsérvese que tiene salidas de compuerta NAND, de modo que sus salidas son bajas activas. Otra indicación es la rotulación de las salidas como $\overline{O7}$, $\overline{O6}$ $\overline{O5}$ etc.. La barra de inversión superpuesta indica que se trata de salidas- bajas activas.

El código de entrada se aplica en A2, A1, A0, donde A2 es el MSB. Con tres entradas y ocho salidas, éste es un decodificador de 3 a 8 o bien, - equivalente, un decodificador de 1 a 8.

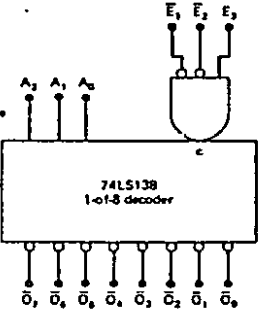
Las entradas $\overline{E1}$, $\overline{E2}$ y E3 son entradas activas separadas que se combi- nan en la compuerta AND. Al fin de activar las compuertas NAND de salida para responder al código de entrada en A2, A1, A0 esta salida de compuerta AND tiene que ser alta, lo cual ocurrirá sólo cuando $\overline{E1} = \overline{E2} = 0$ y E3=1. - Si una o más de las entradas activadas se encuentran en su estado inactivo, la salida de AND será baja, forzando a estado inactivo todas las salidas - de NAND.



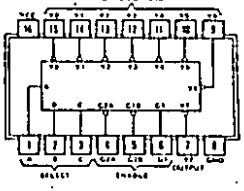
(a)

$\bar{E}_1$	$\bar{E}_2$	E_3	Salidas
0	0	1	Responder al código de entrada $A_2A_1A_0$
1	X	X	Desactivada — todos ALTOS
X	1	X	Desactivada — todos ALTOS
X	X	0	Desactivada — todos ALTOS

Decodificador 1 de 8
74LS138



Diag. 115 16-Pin DIP See Fig. D8
ECG74LS138, ECG74S138



3-Line-to-8-Line Decoder/Demultiplexer

FIGURA 3.1. DIAGRAMA Y TERMINALES 74LS138.

4. MEMORIA RAM MCM6116.

La MCM6116 es una memoria estática de acceso aleatorio 16,384 bits organizada en 2048 palabras de 8 bits.

Algunas de sus principales características son las siguientes:

- Capacidad de 2048 palabras de 8 bits.
- Tecnología HCMOS.
- Completamente estática, no requiere reloj.
- Máximo tiempo de acceso: MCM6116-12 - 120 nS.
  MCM6116-15 - 150 nS.
  MCM6116-20 - 200 nS.
- Retención de información de bajo voltaje 50 uA máximo.

El diagrama de distribución de terminales se encuentra en la figura 4.1.

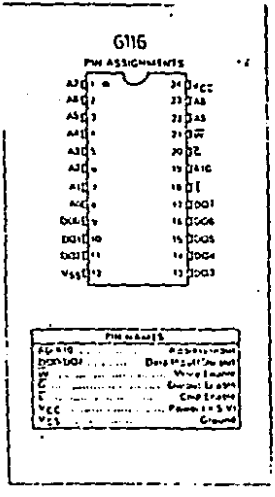


FIGURA 4.1. TERMINALES 6116.

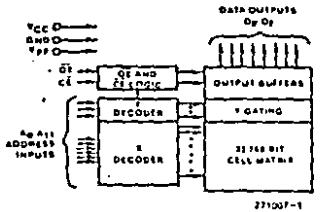
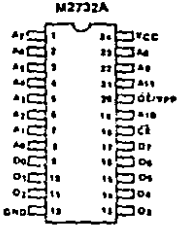


Figure 1. Block Diagram



271007-2

Figure 2. Configuration

Mode Selection

Pins / Mode	CE (18)	OE/Vpp (20)	Vcc (24)	Outputs (9-11, 13-17)
Read	V_{IL}	V_{IL}	+5	Dout
Standby	V_{IH}	Don't Care	+5	High Z
Program	V_{IL}	Vpp	+5	Din
Program Verify	V_{IL}	V_{IL}	+5	Dout
Program Inhibit	V_{IH}	Vpp	+5	High Z

Pin Names

A_0-A_{11}	Addresses
CE	Chip Enable
OE	Output Enable
O_0-O_7	Outputs

FIGURA 5.1. TERMINALES 2732A.

5. MEMORIA EPROM M2732A.

La M2732A es una memoria sólo de lectura programable eléctricamente y borrable con luz ultravioleta, tiene una capacidad de almacenamiento de 32,768 bits.

Consta de dos entradas ENEBLE $\overline{CE}$ y $\overline{OE}$ separadas una de la otra las cuales tienen tiempos de acceso diferentes, dependiendo de la necesidad del tiempo de acceso.

Entre sus principales características se encuentran las siguientes:

- 2 líneas de control.
- Máximo tiempo de acceso 250 nS.
- Tecnología HMOS.
- Compatible a alta velocidad de 5 MHz del MIAPX 86/10 uP.
- Contiene dos líneas de control.
- Rango de temperatura - 55°C a +125°C.

El diagrama de su distribución de terminales se encuentra en la figura 5.1.

6. DISPLAY DE CRISTAL LIQUIDO -DMC SERIES- OPTREX.

Este Display es fácilmente conectable a microprocesadores de 8 bits de datos, teniendo disponibles 4 bits o los 8 bits de interface al microprocesador.

Posee 160 tipos de caracteres entre alfabéticos, numéricos, 32 caracteres especiales y símbolos que pueden ser mostrados por el generador interno de caracteres (ROM).

Este dispositivo puede realizar gran cantidad de operaciones como — "CLEAR DISPLAY" , "HOME CURSOR" , "ON/OFF CURSOR", "BLINK CARACTER", "SHIFT DISPLAY". "SHIFT CURSOR", "READ/WRITE DISPLAY DATE", etc..

A continuación se encuentran las figuras 6.1. que especifica cada terminal del dispositivo Display, la figura 6.2. que muestra las terminales del dispositivo para trabajar con él en forma independiente, la tabla 6.1 — que muestra todo el conjunto de instrucciones que se pueden realizar, y la tabla 6.2. que contiene todos los caracteres disponibles.

PIN ASSIGNMENT

Pin No.	Symbol	Level	Function	
1	Vss	—	Power Supply	0V(GND)
2	Vcc	—		+ 5 V
3	Vee	—		for Liquid Crystal Drive
4	RS	H/L	Register H:Data Input Select L:Instruction Input	
5	R/W	H/L	H:Data Read (Module→MPU) L:Data Write (Module←MPU)	
6	E	H,H→L	Enable Signal	
7	DB0	H/L	Data Bus Line	
8	DB1	H/L		
9	DB2	H/L		
10	DB3	H/L		
11	DB4	H/L		
12	DB5	H/L		
13	DB6	H/L		
14	DB7	H/L		

FIGURA 6.1. TERMINALES DMC 16207.

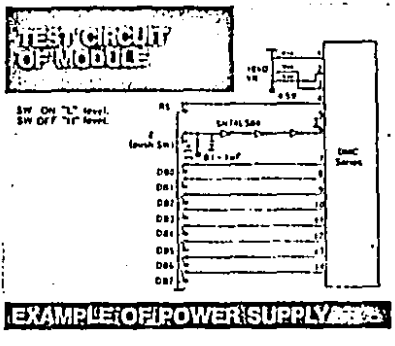


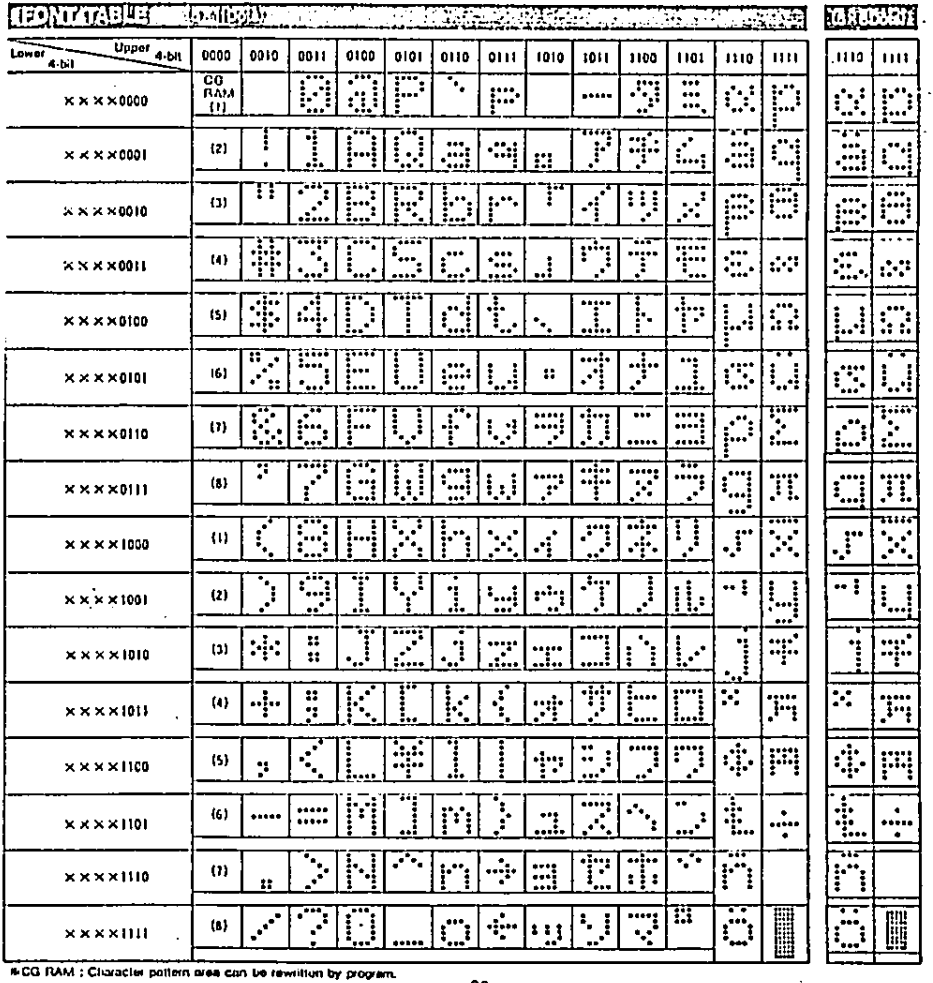
FIGURA 6.2. MODULO DE OPERACION.

INSTRUCTIONS

Instruction	Code										Description	Execute Time(max.) (NOTE 1)
	RS	R/W	DB 7	DB 6	DB 5	DB 4	DB 3	DB 2	DB 1	DB 0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears all display and returns the cursor to the home position (Address 0).	1.64mS
Cursor At Home	0	0	0	0	0	0	0	0	1	*	Returns the cursor to the home position (Address 0). Also returns the display being shifted to the original position. DDRAM contents remain unchanged.	1.64mS
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Sets the cursor move direction and specifies or not to shift the display. These operations are performed during data write and read.	40μS
Display On/Off Control	0	0	0	0	0	0	1	D	C	B	Sets ON/OFF of all display (D) cursor ON /OFF (C) and blink of cursor position character (B).	40μS
Cursor/Display Shift	0	0	0	0	0	1	S/C	R/L	*	*	Moves the cursor and shifts the display without changing DDRAM contents.	40μS
Function Set	0	0	0	0	1	DL	N	F	*	*	Sets interface data length (DL) number of display lines (L) and character font (F).	40μS
CGRAM Address Set	0	0	0	1	ACG						Sets the CGRAM address. CGRAM data is sent and received after this setting.	40μS
DDRAM Address Set	0	0	1	ADD							Sets the DDRAM address. DDRAM data is sent and received after this setting.	40μS
Busy Flag/ Address Read	0	1	BF	AC							Reads Busy flag (BF) indicating internal operation is being performed and reads address counter contents.	0μS
CGRAM/DDRAM Data Write	1	0	Write Data								Writes data into DDRAM or CGRAM.	40μS
CGRAM/DDRAM Data Read	1	1	Read Data								Reads data from DDRAM or CGRAM.	40μS

Code	Description	Execute Time (max.)
I/D = 1 : Increment I/D = 0 : Decrement S = 1 : With display shift S/C = 1 : Display shift S/C = 0 : Cursor movement R/L = 1 : Shift to the right R/L = 0 : Shift to the left DL = 1 : 8-bit DL = 0 : 4-bit N = 1 : 1/16Duty N = 0 : 1/8Duty, 1/11Duty F = 1 : 5×10dots F = 0 : 5×7dots BF = 1 : Internal operation is being performed BF = 0 : Instruction acceptable	DDRAM : Display Data RAM CGRAM : Character Generator RAM ACG : CGRAM Address ADD : DDRAM Address Corresponds to cursor address. AC : Address Counter, used for both DD RAM and CGRAM * : Invalid	fcp or fosc = 250kHz However, when frequency changes, execution time also changes Ex When fcp or fosc = 270kHz, $40nS \times \frac{250}{270} = 37\mu S$

TABLA 6.1.



*CG RAM : Character pattern area can be rewritten by program.

TABLA 6.2.

Pasos para inicializar el Display :

1.- Encender el módulo Display.

2.- Ajustar el contraste hasta que los cuadros del Display sean visibles.

3.- Poner RS y R/$\overline{W}$ en cero lógico.

4.- Enviar los siguientes códigos a la línea de datos.

38 HEX

0F HEX

01

06

Cada uno de los códigos con un pulso enable de 500 nseg.

Después de ser enviados estos códigos a Display, el cursor aparecerá en la esquina superior izquierda o HOME POSITION.

Para enviar caracteres a Display poner RS en 1 y enseguida mandar el carácter deseado en su código ASCII a la línea de datos.

Para posicionar el cursor en cualquier lugar del Display que consta de dos líneas de 16 lugares se logra enviando a la línea de datos el código del lugar deseado, estando RS en cero ya que corresponde a instrucciones. Los códigos son los siguientes y corresponden al lugar seleccionado:

80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F

C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF

7. CONVERTIDORES A/D DE 8 BITS ADC0802, ADC0803, ADC0804, ADC0805, COMPATIBLES CON MICROPROCESADORES.

7.1.- DESCRIPCION GENERAL-.

Son convertidores A/D de aproximación sucesiva y de tecnología C-MOS- con una resolución de 8 bits los cuales usan una escala diferencial potenciométrica similar a la de los productos 256R. Estos convertidores han sido diseñados para permitir una operación con el bus de control derivativo- NSC800 o INS8080A. El microprocesador toma a estos dispositivos como localidades de memoria o como puertos de entrada/salida y no se requiere de - ningún tipo de lógica para la conexión.

Un nuevo voltaje analógico diferencial de entrada permite aumentar el rechazo de modo común así como balancear el valor de entrada cero de voltaje de entrada.

Caracterfsticas:

- Compatible con microP 8080 y derivados- no es necesaria una lógica- de interface con todos los microprocesadores.
- Voltajes analógicos de entrada diferenciales.
- Trabaja con 2.5V (LM336) de voltaje de referencia.
- Rango de voltaje analógica de entrada 0V a 5V con suministro de 5V.
- No requiere de ajuste de cero.

Especificaciones:

Resolución	8bits
Error Total	±1/4LSB,±1/2LSB y ±1 LSB
Tiempo de Conversión	100 useg.

7.2. DESCRIPCION FUNCIONAL.-

La serie ADC0801 contiene un circuito equivalente a la red 256R. Switches analógicos están secuenciados por lógica de aproximación sucesiva para acompañar la diferencia de voltaje de entrada [Vin(+)-Vin(-)] con su correspondiente conexión en la red R. El bit más significativo es probado y después de 8 comparaciones (64 ciclos de reloj) un código binario digital de 8 bits (1111 1111= escala completa) es transferido a un cerrojo de salida y luego una interrupción es asegurada ($\overline{INTR}$ realiza una transición de alto a bajo). Una conversión en proceso puede ser interrumpida mediante la introducción de un segundo comando start. El dispositivo puede ser operado en modo de operación libre conectando $\overline{INRT}$ a la entrada $\overline{WR}$ con $\overline{CS}$=0. Para asegurar un inicio alto, bajo todas las posibles condiciones, un pulso externo debe ser aplicado a $\overline{WR}$ durante el primer ciclo de encendido.

Cuando ocurre la transición de alto a bajo en la entrada $\overline{WR}$ los estados de los cerrojos internos del -SAR- y del registro de corrimientos son reestablecidos.

Mientras las entradas $\overline{CS}$ y $\overline{WR}$ se encuentren en el estado bajo, el convertidor analógico-digital permanecerá en estado de reestablecimiento.

Una conversión comenzará en al menos de 1 a 8 períodos de reloj después de la primera transición de bajo a alto de alguna de las entradas $\overline{WR}$ o $\overline{CS}$.

El convertidor es inicializado cuando $\overline{CS}$ y $\overline{WR}$ pasan al estado bajo simultáneamente. Con lo cual se activa el flip-flop (pasa al estado uno) de inicio y el nivel "1" resultante reinicia el registro de corrimiento de 8 bits, reinicia la interrupción (INTR) F/F y proporciona de un "1" a la entrada del flop D, F/F1, el cual es la entrada final del registro de corrimiento de 8 bits. La señal del reloj interno transfiere entonces este "1" hacia la salida Q del F/F1. La compuerta AND combina esta salida "1" con la señal de reloj para proveer una señal de inicio al F/F. Si esta señal de inicio no se encuentra presente por mucho tiempo (ya sea que $\overline{WR}$ o $\overline{CS}$ sean "1") el F/F de inicio es inicializando y entonces el registro de corrimiento de 8 bits obtiene su señal de reloj, con lo cual da inicio el proceso de conversión. Si la señal de inicio aún se mantuviera presente, este pulso de inicio no tendría efecto (ambas salidas del F/F estarían momentaneamente en el nivel "1") y el registro de corrimiento de 8 bits continuaría en estado de RESET. Esta lógica permite que el convertidor sólo se inicialice hasta que al menos una de las señales, $\overline{CS}$ o $\overline{WR}$, regresen al estado alto y que el reloj interno provea de una señal de reinicio al F/F de inicio.

Después de que el "1" pase por el registro de corrimiento de 8 bits, - aparecerá como la entrada del cerrojo tipo D, LATCH 1; la compuerta AND, G2, causa que la nueva palabra digital sea transferida a los cerrojos de salida, los cuales son de tipo Tri-estado. Cuando LATCH 1 es subsecuentemente habilitado, la salida Q realiza una transición de alto a bajo lo cual causa que - INTR F/F sea inicializado. Un buffer inversor proporciona la señal de entrada $\overline{\text{INTR}}$.

Nótese que este control de inicio ($\overline{\text{SET}}$) de INTR F/F se mantiene en el estado bajo durante 8 períodos del reloj externo (ya que el reloj interno trabaja a una velocidad de 1/8 de la frecuencia del reloj externo). Si el dato de salida es permanentemente habilitado ($\overline{\text{CS}}$ y $\overline{\text{RD}}$, ambos se mantienen en el - estado bajo), la salida $\overline{\text{INTR}}$ indicará el fin de una conversión (por medio de una transición de alto a bajo).

Cuando el convertidor opera en "modo libre" o modo de conversiones continuas ($\overline{\text{INTR}}$ conectado a $\overline{\text{WR}}$ y $\overline{\text{RD}}$ conectado a tierra) entonces el F/F de inicio es inicializado mediante la transición de alto a bajo de la señal $\overline{\text{INTR}}$.- Esto reinicia el registro de corrimiento lo cual causa que la entrada al cerrojo D LATCH 1, vaya al estado bajo. Como la entrada habilitadora del cerrojo se mantiene presente, la salida $\overline{Q}$ pasará al estado alto lo cual permite - que el INTR F/F sea reinicializado. Esto reduce el ancho del pulso resultante de la salida $\overline{\text{INTR}}$ a unas pocas demoras de propagación (aproximadamente 30 nS).

Cuando los datos van a ser leídos, la combinación $\overline{CS}$ y $\overline{RD}$ igual a cero causa que el INTR F/F sea reinicializado y los cerrojos tipo tri-estado de salida sean habilitados para proveer la salida digital de 8 bits.

7.3. - ENTRADAS DIGITALES DE CONTROL.-

Las entradas digitales de control ($\overline{CS}$, $\overline{RD}$ y $\overline{WR}$)son bajas activas para- permitir una fácil interface con las líneas de control de los microprocesadores. Para aplicaciones en las que no se desee realizar una interface con microprocesadores, la entrada $\overline{CS}$ puede ser conectada a tierra, la función - de inicio del convertidor puede ser llevada a cabo aplicando un pulso bajo a la entrada $\overline{WR}$ y, por último, la función habilitar salida se logra mediante un pulso bajo a la entrada $\overline{RD}$.

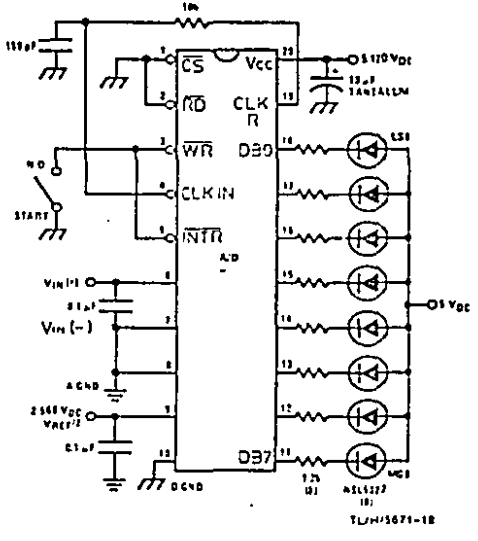


FIGURA 7.1. Diagrama de pruebas.

8. SENSOR DE TEMPERATURA (DE PRESICION) LM35D.

8.1. -DESCRIPCION GENERAL-

La serie LM35 son circuitos sensores de temperatura de presición, cuya salida de voltaje es linealmente proporcional a la temperatura en grados - centígrados. En este tipo de circuitos no es necesaria una calibración ex - terna o el sustraer un voltaje constante desde la salida para obtener la - conveniente escala de grados. La salida de baja impedancia del LM35, su res puesta lineal, y su precisa calibración hacen que sea una interface de lec- tura especialmente fácil para circuitos de control.

CARACTERISTICAS:

- Calibrado directamente en grados centígrados.
- Respuesta lineal de +10.0mV por cada grado.
- Operado desde 4 hasta 30 volts.
- Menos de 60uA de corriente drenada.
- Baja impedancia de salida, 0.1 OHMS por 1mA cargado.

Connection Diagrams

TO-46
Metal Can Package*

*Case is connected to negative pin

Order Number LM35H, LM35AH,
LM35CH, LM35CAH or LM35DH
See NS Package Number H03H

TO-92
Plastic Package

Order Number LM35CZ or LM35DZ
See NS Package Number Z03A

Typical Applications

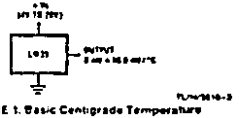


FIGURE 1. Basic Centigrade Temperature Sensor (+2°C to +150°C)

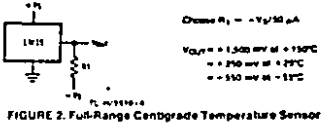


FIGURE 2. Full-Range Centigrade Temperature Sensor

FIGURA 8.1. DIAGRAMA DE CONEXION.

9. SEGUIDOR DE VOLTAJE.

9.1. - DESCRIPCION GENERAL -

El circuito de la figura 9.1 se llama seguidor de voltaje pero también se conoce como: amplificador seguidor de fuente, amplificador de ganancia 1 o amplificador de aislamiento . El voltaje de entrada Ei, se aplica directamente a la entrada (+). Ya que el voltaje entre las terminales (+) y (-) del OPAM puede considerarse 0, Vo = Ei.

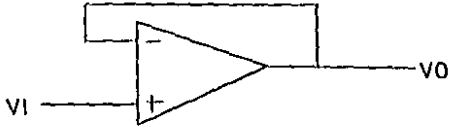


Figura 9.1. Seguidor de voltaje.

Obsérvese que el voltaje de salida iguala al voltaje de entrada tanto en magnitud como en signo. Por tanto, tal como el nombre del circuito lo dice, el voltaje de salida sigue al voltaje de entrada o fuente. La ganancia del voltaje es 1 como se muestra por: Acl = Vo / Ei = 1.

9.2 - USO DEL SEGUIDOR DE VOLTAJE -

Con frecuencia surge la pregunta: por qué preocuparse en usar un amplificador con ganancia de 1 ? la respuesta puede comprenderse mejor si se compara el seguidor de voltaje con un amplificador inversor. El interés -

principal no se centra en la polaridad del voltaje de ganancia, sino más - bien en el efecto de voltaje de entrada.

El seguidor de voltaje se utiliza porque su resistencia de entrada es alta (en megaOHMS). Por tanto extrae corriente despreciable de una fuente - de señal.

La caida de voltaje atraves de Rin es 0 V. El voltaje que se aplique a la entrada en (+) llega a ser el voltaje de entrada al amplificador y es - igual al generado E gen. Vo = Ei= E gen

Si la misma fuente de señal se conecta a un inversor, la R de entrada es Rin y Ri (Rin esta dentro del generador). Esto proporciona que el voltaje del generador E gen se divida entre Rin y Ri, esto es, Ei = Ri / (Rint + Ri) * E gen, dandonos un valor de Vo menor al Ei pero invertido (-).

Si se debe amplificar e invertir una fuente de señal de un circuito de alta impedancia y se desea no drenar corriente de la señal, primero amortigüese la fuente con un seguidor de voltaje.