UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES

"ARAGON"

"Controlador Lineal Programable de Temperatura"

T E S I S

Que para obtener el Título de:

INGENIERO EN COMPUTACION

Presentan:

MARCO ANTONIO CASTILLO ALVAREZ

JOSE MARTIN CENDEJAS HERNANDEZ

Asesor: Ing. Silvia Vega Muytoy

México, D. F. 1994

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

CONTROLADOR LINEAL PROGRAMABLE DE TEMPERATURA

AGRADECIMIENTOS

Primero a Dios, por la vida y la licencia de alcanzar nuestra meta.

A nuestros Padres por la bendición de su cariño y apoyo incondicional.

A nuestros Maestros y Asesores por el privilegio de compartirnos sus conocimientos.

MCA/MCH

"Levántate, ataviate, ponte de pie,
goza del hermoso lugar:
la casa de tu madre, tu padre, el Sol.
Allí hay dicha, hay placer, hay felicidad.
Condúcete, sigue a tu madre, a tu padre, el Sol..."

Códice Florentino, lib. VI, fol. 141, v.; AP I, 37.

"El que hace sabio los rostros ajenos,
hace a los otros tomar una cara,
los hace desarrollarla...
Pone un espejo delante de los otros, los hace
cuerdos, cuidadosos,
hace que en ellos aparezca una cara...
Gracias a él la gente humaniza su querer
y recibe una estricta enseñanza..."

Textos de los informantes, vol. VIII,
fol. 118, v.; AP I,8.

INDICE

INTRODUCCION

El diseño de un circuito implica conocer el tipo de aplicación así como las necesidades del usuario. Es igualmente importante conocer las posibles alternativas que se tienen para resolver el problema, tratando siempre que el diseño cubra todas las expectativas y de ser posible incorpore mejoras que puedan ser de utilidad para el usuario, estableciendo una interface amigable.

Las alternativas de solución del problema implica llevar a cabo una investigación de lo que ofrece el mercado, estudiar las ventajas de un tipo de tecnología con respecto a otra, la compatibilidad de las mismas y la posibilidad de introducir mejoras en un futuro sin tener que desechar lo que ya se tiene, y por lo tanto realizar un gasto adicional elevado.

Si el diseño se realizó sin cubrir los puntos anteriores y agregamos a ésto con una tecnología que ha dejado de ser útil, resulta más conveniente elaborar un nuevo diseño. El avance de la microelectrónica nos proporciona integrados que incluyen un gran número de funciones incluidas en una sola pastilla, cosa que no podemos pasar por desapercibido.

La elaboración del Controlador Lineal Programable de Temperatura estaba planeada desde el año de 1982, por parte del Instituto Mexicano del Petróleo, y para su implementación se había pensado en el microprocesador Z80. El proyecto fue retomado en Enero de 1992, es decir, diez años después, por lo que el Z80 resultaba, sin subestimar sus potencialidades, nada práctico. El circuito ocupaba mucho espacio y se hacía un abuso de recursos, ya que no se empleaba el computador Z80 a toda su potencialidad.

Se estudiaron las alternativas para sustituir el Z80 y se pensó en el uso de algún microcontrolador. La familia de microcontroladores MC68705 de Motorola resultó la más adecuada a lo que se quería. Se pensó en el uso del MC68705R3, por poseer una capacidad de memoria, en RAM y EPROM, superior a la de los demás, y por incluir un convertidor analógico digital, lo cual significaba una reducción importante en el circuito. Sin embargo su comercialización resultó escaza. Finalmente se optó por el MC68705P3, el cual se acerca más en características al R3, sin incluir el ADC.

Los microcontroladores o microcomputadores de la serie MC68705 pertenecen a la serie de microcomputadoras de la familia M6805 de bajo costo en un solo chip. Incluyen CPU, Unidad de Reloj, EPROM, RAM, Puerto de Entrada/Salida, TIMER y ROM con un programa de fábrica para autograbación de la EPROM, además de un conjunto muy completo de mnemotécnicos fáciles de utilizar.

La capacidad de programación de los microcontroladores incorpora la característica mencionada anteriormente: cualquier cambio que quiera hacerse, basta con modificar el programa y extender algún puerto para realizar tareas de multiplexaje. Las mejoras se limitan al grado de desembolso que quiera hacerse y siempre y cuando el circuito no deje de ser práctico.

El objetivo del Controlador Lineal Programable de Temperatura se basa en mantener una temperatura constante dentro de un horno, empleado para calentar diferentes tipos de sustancias utilizadas en la obtención y elaboración de diferentes petroquímicos. En ciertas ocasiones basta alcanzar la temperatura requerida y en situaciones especiales, no sólo alcanzar la temperatura fijada sino mantenerla por un tiempo establecido.

De lo anterior surgió la elaboración de tres tipos de controles: Control ON-OFF, Control Proporcional y Control Proporcional Automático por Pasos Temporizados, incluyendo una opción para funcionar como un simple termómetro.

En Control ON-OFF se establece una temperatura o SET POINT, si dicha temperatura está por arriba de la temperatura actual, el horno se enciende y se apaga una vez alcanzada. En Control Proporcional, al igual que en ON-OFF, se fija una temperatura, la cual una vez alcanzada, se mantiene hasta que el usuario decida abortar el proceso. Por último, en Control Proporcional por Pasos Automáticos Temporizados, se establece un incremento de temperatura, el período de tiempo durante el cual se mantendrá cada nueva temperatura y la temperatura final.

El presente trabajo se ha dividido para su estudio en tres capítulos. En el primer capítulo, titulado "Hardware", se trata la configuración global del circuito así como el desglose de cada uno de los módulos que lo integran. También se incluye una sección especial dedicada al estudio del microcontrolador MC68705P3, donde se exponen cada una de las partes que lo integran.

El segundo capítulo lleva como nombre "Software", en él se hace un estudio sobre las características de programación del MC68705P3, incluyendo una explicación del ensamblador y de la estructura general de los programas. Se presenta el programa completo en donde recide la lógica de funcionamiento del circuito, haciendo un desglose de cada una de las rutinas elaboradas para tal efecto.

Finalmente se tiene un capítulo dedicado a exponer las pruebas y resultados obtenidos en forma experimental, tal capítulo lleva por nombre "El Sistema Integrado: Pruebas y Resultados".

El diseño no representa en ningún momento un trabajo terminal y queda abierto a mejoras, esperando que resulte de utilidad para todos aquellos que se interesen en el desarrollo de sistemas automáticos de control mediante el empleo de microcontroladores.

MCA/MCH

CAPITULO 1
HARDWARE

CAPITULO 1. HARDWARE.

1.1. CONCEPTOS BASICOS.

El desarrollo experimentado en los últimos años en el campo de la electrónica ha hecho de los circuitos digitales una opción de diseño más rápida y eficiente, aunada a una reducción significativa de cada uno de los elementos integrantes. El empleo de los microprocesadores y microcomputadoras ha revolucionado la industria de la electrónica y ha tenido un tremendo impacto en todas las actividades desarrolladas por el ser humano pues aún las actividades de carácter humanístico y artístico involucran el uso de la computadora como un elemento auxiliar.

Las capacidades y los costos de los microprocesadores son muy variados y con un crecimiento muy rápido. La elección de alguno depende del tipo de aplicación. Para la realización de funciones sencillas de bajo volumen, el empleo de un microprocesador puede resultar costoso en comparación con el empleo de una lógica ordinaria, y en algunos casos contraproducente, pues además del desperdicio de sus capacidades puede resultar en una aplicación demasiado lenta.

Como una opción alternativa a los microprocesadores se han desarrollado integrados con capacidades similares a las de áquellos pero cuyos costos resultan más accesibles. Los microcomputadores o microcontroladores, como se les ha llamado a este tipo de circuitos, cuentan con los registros básicos suficientes incluidos en un microprocesador y son capaces de realizar las mismas tareas que aquéllos aunque en forma más reducida, lo cual lejos de ponerlos en desventaja, les da una mayor flexibilidad haciéndolos más fáciles de manejar con respecto a los mismos microprocesadores. Existe un tipo de microcontrolador a la medida de cada necesidad del cual no se desaprovecha nada. Su aplicación está enfocada principalmente al diseño de aquellos circuitos destinados a realizar tareas fijas con una cierta capacidad de programación. Dicha capacidad está limitada por los circuitos periféricos y de memoria que se hayan incluido dentro del diseño y por el mismo programa desarrollado en el microcontrolador.

Un microcontrolador es una computadora en pequeño programable por el usuario, con Unidad Central de Procesamiento (CPU) y memoria EPROM incluida. La EPROM del microcontrolador se graba con el programa de aplicación que será ejecutado cada vez

que el circuito entre en funcionamiento. Una vez que el programa ha dejado de ser útil, puede borrarse de la memoria para incluir otro versión mejorada o bien para destinarlo a otra aplicación diferente. Sin embargo, el número de veces que un microcontrolador puede ser borrado y regrabado está limitado y varía de una versión a otra.

La aplicación de los microcontroladores se enfoca principalmente al diseño de circuitos controladores en los cuales se requiere de un monitoreo constante de las condiciones de un proceso, mediante la lectura de una o varias variables involucradas dentro del mismo. Como respuesta, el controlador produce las señales necesarias para corregir las desviaciones experimentadas por el sistema o bien, en situaciones críticas, las señales de aviso oportunas.

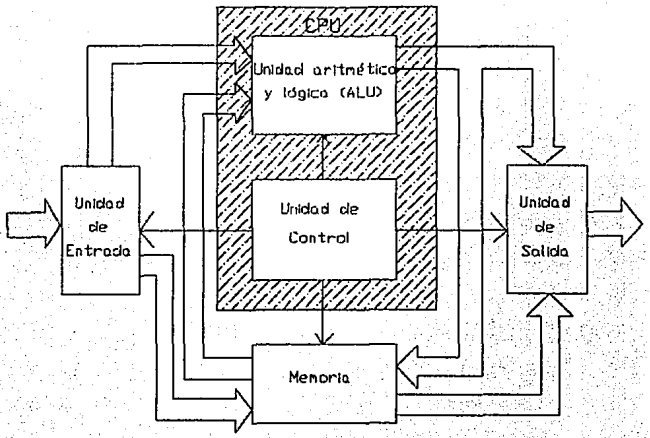
1.1.1. ORGANIZACION BASICA DE UN MICROCOMPUTADOR.

Al igual que un microprocesador, un microcomputador o microcontrolador cuenta con dos unidades básicas: la unidad aritmética y lógica y la unidad de control. Internamente cuenta con varios registros que emplea para almacenamiento y transferencia de datos entre los propios circuitos del microcontrolador, o entre el microcontrolador y la memoria o los puertos de entrada/salida. Véase **Fig. 1.1.**

1.1.1.1. Unidad Aritmética y Lógica.

La Unidad Aritmética y Lógica (UAL) como su nombre lo indica, permite realizar operaciones binarias aritméticas y lógicas con los datos. La Unidad de Control indica a la UAL que operación específica se debe llevar a cabo. Los datos empleados por la UAL pueden provenir de la unidad de memoria, de algún registro del microcomputador o bien de la unidad de entrada. Los resultados de las operaciones realizadas en la UAL pueden transferirse en forma inversa a la memoria, a algún registro interno o bien, hacia algún registro externo a través de la unidad de salida.

Fig. 1.1.– Estructura básica de un microcomputador.



1.1.1.2. Unidad de Control.

La Unidad de Control se encarga de dirigir la ejecución de todas las operaciones realizadas por las demás unidades mediante señales de distribución y control. La unidad de control se alimenta con las instrucciones del programa en curso y su trabajo es la ejecución, una a una, de estas instrucciones.

La Unidad de Control busca y trae una instrucción de la memoria. Esta instrucción, en código binario, es decodificada por los circuitos lógicos de la unidad de control para determinar la acción a ejecutar.

1.1.1.3. Unidad de memoria.

La Unidad de Memoria almacena grupos de palabras que pueden representar datos o instrucciones para el microcomputador. La operación de la memoria está controlada por la unidad de control la cual se encarga de habilitar el funcionamiento de la misma, indicando la operación a realizar. La operación realizada sobre la memoria puede ser una operación de lectura o escritura. Una localidad dada en la memoria se accesa por la unidad de control, ofreciendo el código de dirección adecuado. La memoria también funciona como un medio de almacenamiento temporal de los datos.

. Existen dos tipos de memoria implementadas en los sistemas microcomputadores: la memoria de acceso aleatorio (RAM) y la memoria de solo lectura (ROM). La memoria RAM se utiliza para almacenar datos, parámetros variables y resultados intermedios que necesitan actualizarse y están sujetos a cambio. La ROM se utiliza para almacenar programas y tablas de constantes que no cambian su valor una vez grabadas.

El espacio direccionable de un procesador es la cantidad de memoria que puede direccionar sin ayuda de un banco externo de selección. El direccionamiento de la memoria puede establecerse por medio de un tabla en donde se especifica la dirección de memoria asignada a cada una de las pastillas RAM y ROM utilizadas por el microcomputador. La tabla, denominada "mapa de dirección de memoria", es una representación gráfica del espacio de dirección asignado para cada una de las pastillas en el sistema.

1.1.1.4. Unidad de entrada.

La Unidad de Entrada involucra todos los dispositivos empleados para la lectura de datos externos al microcomputador para almacenarlos en la unidad de memoria o en la unidad aritmético lógica. La unidad de control se encarga una vez más de determinar el lugar de almacenamiento de los datos de entrada. Entre los dispositivos de entrada se pueden mencionar el teclado, los convertidores analógico digitales y los interruptores de reestablecimiento e interrupción.

1.1.1.5. Unidad de salida.

La unidad de salida consta de los dispositivos que se usan para transferir datos e información del microcomputador al mundo exterior. La unidad de control se encarga de dirigir los datos de salida, provenientes de la memoria o de la unidad aritmético lógica, hacia el dispositivo adecuado. Entre los dispositivos de salida se cuentan las pantallas de cristal líquido, las luces indicadoras LED, los convertidores Digital Analógicos y las señales de tipo audible como las bocinas.

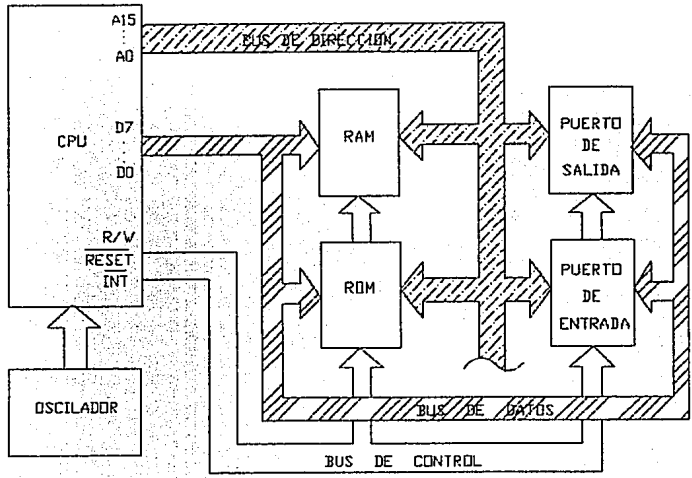
1.1.2. LINEAS DE COMUNICACION DEL SISTEMA

El microcomputador cuenta con tres líneas que transportan toda la información y señales implicadas en la operación del sistema, tal como puede apreciarse en la **Fig. 1.2.** Estas líneas comunican a la Unidad Central de Procesamiento (CPU) con cada uno de los elementos de memoria y entrada/salida de forma que los datos puedan fluir del microcomputador a cualquiera de estos elementos.

Todas las operaciones de transferencia están dirigidas hacia la CPU. Cuando la CPU envía datos a otro elemento del microcomputador se lleva a cabo una operación de escritura (WRITE). En el caso contrario, cuando la CPU recibe datos de algún dispositivo, se lleva a cabo una operación de lectura (READ).

Las líneas implicadas en la transferencia y control de datos son la línea de direcciones, la línea de datos y la línea de control.

Fig. 1.2.- Líneas de comunicación del sistema.



1.1.2.1. Línea de direcciones.

Constituye una línea unidireccional, es decir que la información fluye en una sola dirección, de la CPU a la memoria o a los puertos de entrada/salida. Cada una de las direcciones implicadas corresponde a una localidad de memoria o a un elemento de entrada/salida.

Siempre que se lleva a cabo una operación de comunicación de la CPU con algún otro dispositivo, la CPU coloca el código de dirección adecuado en la línea de direcciones decodificado para seleccionar el elemento de comunicación requerido.

1.1.2.2. Línea de datos.

Constituye una línea bidireccional a través de la cual viaja el dato a ser transferido o recibido por la CPU o algún otro registro del microcomputador. La longitud de esta línea es igual al tamaño de palabra del microcomputador.

1.1.2.3. Línea de control.

Esta línea se emplea para enviar cada una de las señales que se usan para sincronizar las actividades de los elementos separados del microcomputador como son las señales de reestablecimiento e interrupción.

1.1.3. REGISTROS DEL MICROCOMPUTADOR.

La arquitectura de un microcomputador está formada por los registros implementados en el mismo. El número y tipo de registros de un microcomputador tiene un efecto marcado en el esfuerzo de programación que se requiere para una aplicación dada.

Los registros se emplean por lo general para transferencia de información y también como almacenamiento temporal de datos. El número de registros implementados en cada microcomputador varía de un fabricante a otro, sin embargo por lo general se incluyen los registros detallados en los párrafos subsecuentes.

1.1.3.1. Contador de programa (PC).

El contador de programa contiene la dirección de memoria de la siguiente instrucción a ejecutar por parte de la CPU. Cada vez que el sistema se reinicializa el PC se fija con el valor de la primera instrucción a ser ejecutada, incrementándose a medida que las instrucciones del programa se van concluyendo.

El tamaño del PC depende del número de bits de la línea de direcciones y por lo general se encuentra fragmentado en dos partes, PCH y PCL. El motivo de esta división se debe a que generalmente el valor del PC necesita almacenarse en la memoria, cuyo tamaño de palabra es menor al número de bits de la línea de direcciones, por lo que se tiene que almacenar en dos localidades de memoria consecutivas.

1.1.3.2. Acumulador.

El acumulador es un registro empleado como almacenamiento temporal de las operaciones realizadas por la unidad aritmético lógica. En muchas instrucciones de la UAL, el acumulador es la fuente de uno de los operandos y el destino del resultado.

El tamaño del acumulador es igual al tamaño de palabra del microcomputador. Entre otras aplicaciones, el acumulador sirve como fuente de recepción de los datos provenientes de un dispositivo de entrada y como fuente de transmisión de los datos hacia un dispositivo de salida.

1.1.3.3. Registros de uso general.

Este tipo de registros se utilizan para realizar funciones de almacenamiento temporal o como índices. El número de estos registros varía fuertemente de un fabricante a otro. Algunos no tienen ninguno mientras que otros llegan a tener 12 o más.

1.1.3.4. Registro índice.

Este registro puede utilizarse como medio de almacenamiento temporal, sin embargo su uso esta más enfocado al uso de tablas. Para la ubicación de un elemento dentro de la tabla, a la dirección apuntada por el PC se le suma el valor del registro índice. Este tipo de operación constituye un modo de direccionamiento de tipo indexado.

1.1.3.5. Registro de condiciones.

Este registro consta de un conjunto de bits individuales con diferentes significados. Estos bits se emplean para indicar un cierto estado del microcomputador. El valor de cada uno de estos bits puede ser examinado mediante programa y dependiendo del estado en que se encuentre realizar una operación determinada. Dos de la señales más comunes son la señal cero (Z) y la señal de acarreo (C). La señal cero indica si la instrucción precedente produjo o no un resultado igual a cero. La señal de acarreo por su parte indica si la instrucción precedente rebasó el tamaño de palabra del microcomputador.

1.1.3.6. Registro apuntador de pila.

Una pila es un dispositivo de almacenamiento que almacena información de tal manera que el dato almacenado al último es el primero en salir. Una pila puede existir como una unidad sola o puede ser implementada en una memoria de acceso aleatorio asignada al CPU. La implementación de una pila en la CPU se hace asignando una porción de la memoria del microcomputador a una operación de la pila y utilizando un registro procesador dedicado como indicador de la pila el cual se denomina registro apuntador de pila (SP).

El registro apuntador de pila actúa como un registro especial de direcciones de memoria que se usa sólo para la porción de la pila en RAM. Siempre que se vaya a almacenar una palabra en la pila, ésta se almacena en la dirección contenida en el apuntador de pila. Análogamente, siempre que se va a leer una palabra de la pila, ésta se lee de la dirección especificada por el apuntador de pila. El contenido del apuntador de pila es inicializado al tope de la pila al inicio del programa. Por lo tanto, el SP se disminuye automáticamente después de que una palabra es almacenada en la pila y se incrementa antes de que una palabra sea leída de la pila.

1.2. CONFIGURACION GLOBAL DEL CIRCUITO

El Hardware del equipo, tal como se muestra en la **Fig. 1.3**, se basa en el Microcontrolador MC68705P3 perteneciente a la familia de procesadores M6805 de Motorola. En este circuito radica el programa monitor encargado de administrar todas las tareas de control y manipulación de periféricos.

Conectados a sus tres puertos se encuentran los circuitos de entrada y salida de datos así como los elementos de sincronización de señales, juntos constituyen la interface entre el sistema y el usuario.

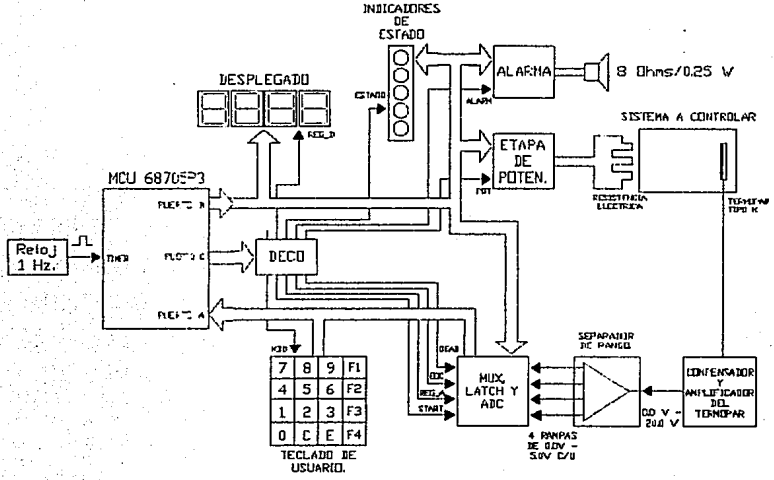
El diseño se llevó a cabo en forma modular aprovechando la capacidad de programación de los puertos del microcontrolador conectados internamente a un bus común de datos. De esta forma se consiguió aislar cualquier problema concentrando la atención en el módulo en cuestión sin la necesidad de afectar el diseño de las demás partes.

Aprovechando al máximo las potencialidades del microcontrolador se logró un diseño reducido, pues se evitó el uso de un mayor número de integrados así como de memoria externa, enfocando una mayor atención en la programación del microcontrolador.

En la **Fig. 1.3** se muestra la configuración global del circuito en la cual podemos apreciar el microcontrolador MC68705P3 y todos los circuitos periféricos interconectados al mismo a través de sus puertos. El puerto B se ha configurado como un puerto de salida. A este puerto se encuentran conectados la pantalla de visualización de datos, los leds indicadores de estado, la alarma, la etapa de potencia y el circuito de interface del Convertidor Analógico Digital (ADC).

La pantalla de visualización constituye uno de los elementos más importantes del controlador ya que establece el punto de unión, junto con el teclado, entre el usuario y el sistema. El usuario puede constatar los datos introducidos a través del teclado y corregirlos en caso de detectar alguna anomalía la cual no se podría descubrir si el usuario no pudiera observar lo que está introduciendo. A través de la pantalla de visualización, el sistema puede informar al usuario de los cambios que se están

Fig. 1.3.- Controlador de Temperatura.



suscitando y éste, tomar las medidas necesarias para corregir las posibles desviaciones que pudieran presentarse.

Los Leds indicadores de estado, como su nombre lo indica, muestran el estado presente del sistema con fines informativos para el usuario, como lo es la secuencia en la captura de los datos, el tipo de control que se está ejecutando o bien, la existencia de algún error en los datos introducidos o del propio sistema.

La alarma constituye un elemento importante en cualquier sistema de control por sus características indicativas, por lo general se responde con más rapidez a una señal audible que a una visual en un ambiente en el cual el usuario desempeña funciones variadas. Gracias a la automatización de procesos y al uso creciente de la computadora y otros elementos de control, el hombre puede dedicarse a otras actividades más productivas e intervenir en los procesos sólo en condiciones extremas, y que otra mejor forma de llamar la atención que el empleo de una alarma.

La etapa de potencia es el medio de conexión entre el sistema controlador y el elemento a controlar, ya que finalmente se va a encargar de trasmitir las señales necesarias para que el horno empiece a calentar o bien, deje de hacerlo en caso de que así se requiera de acuerdo a la temperatura establecida.

El puerto A, configurado como puerto de entrada, constituye el punto de comunicación complementario del usuario con el sistema. A este puerto se encuentran conectados el Teclado y el Convertidor Analógico Digital.

El teclado es el elemento a través del cual el usuario va a introducir los datos necesarios para la operación del sistema. Permite tanto la captura como la corrección de errores detectados en tiempo real gracias a la pantalla de visualización de datos.

El convertidor analógico digital es quizá uno de los elementos más importantes del sistema ya que es el encargado de transformar una señal analógica de entrada a una señal digital equivalente de salida. A partir del voltaje equivalente a la temperatura del horno (señal analógica) el convertidor se encarga de generar una señal digital equivalente que puede ser manipulada por el sistema. Esta señal, constantemente monitoreada, constituye la variable de control del sistema.

Finalmente, conectado al microcontrolador MC68705P3, se puede distinguir un reloj de tiempo real con una frecuencia de operación de 1 Hz, útil en tareas de rampa múltiple con tiempos de espera definidos.

Cada uno de los módulos mencionados, se habilita mediante programa haciendo uso del puerto C, fijado como puerto de salida y auxiliado por un decodificador.

1.3. UNIDAD MICROCONTROLADORA MC68705P3.

1.3.1. DESCRIPCION DE PINES

A continuación se describen cada uno de los pines de entrada salida del microcontrolador MC68705P3, tal como se puede apreciar en la **Fig. 1.4**:

Vcc y Vss Constituyen los pines de alimentación del circuito. Vcc corresponde a 5.0 Volts y Vss a 0.0 Volts (tierra del sistema).

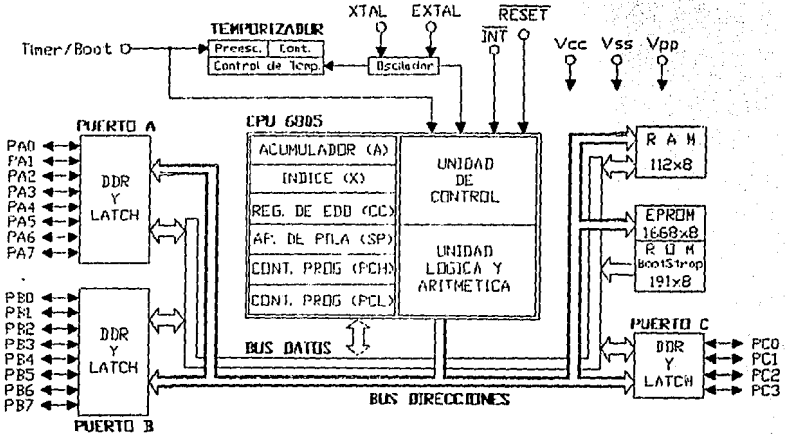
$\overline{\text{INT}}$ Este pin permite generar una interrupción del sistema en forma asíncrona mediante una señal externa. También puede usarse como una línea de entrada controlada mediante las instrucciones BIL y BIH del conjunto de Mnemotécnicos del microcontrolador.

XTAL, EXTAL A través de estos pines se establece la fuente de reloj del sistema. Dependiendo del valor del bit CLK en el Registro de Opción por Máscara (MOR) puede elegirse entre una resistencia, un cristal o bien una fuente externa.

TIMER/BOOT Este pin se emplea como una entrada externa para controlar la circuitería del temporizador propio del sistema. Así mismo se encarga de detectar un nivel alto de voltaje empleado para inicializar el Programa de Carga y Autograbación de Datos en la EPROM.

$\overline{\text{RESET}}$ Este pin cuenta internamente con un disparador Schmitt y una resistencia de fijación para efectos de reinicialización del integrado, por lo cual puede ser reestablecido durante el encendido primario del circuito o bien mediante un nivel bajo de voltaje aplicado en este pin.

Fig. 1.4.- Diagrama a Bloque del MCU68705P3.



Vpp Este pin se emplea con el fin de proporcionar un voltaje de programación requerido para la grabación de la EPROM. Durante la operación normal del circuito este pin debe estar conectado a Vcc.

PA0-PA7, PB0-PB7, PC0-PC3 Estas 20 líneas se encuentran dispuestas en dos puertos de ocho bits (A y B) y uno de cuatro (C) y constituyen líneas de Entrada/Salida mediante las cuales el microcontrolador se comunica con el exterior. Todos los puertos son programables como entradas o salidas bajo control de programa del Registro de Dirección de Datos correspondiente (DDR).

1.3.2. ORGANIZACION DE LA MEMORIA DEL MC68705P3.

Como se muestra en la **Fig. 1.5**, el MCU es capaz de direccionar 2048 bytes de memoria y registros de Entrada/Salida por medio de su Contador de Programa (PC). De las 2048 localidades mencionadas siete no son empleadas, teniendo acceso realmente a solo 2041, las cuales se dividen como sigue:

a) 1804 bytes de EPROM para programa de usuario.

b) 115 bytes de ROM para el programa de arranque y autograbación de la EPROM.

c) 112 bytes de RAM.

d) 1 byte para el Registro de Opción Enmascarable (MOR, Mask Option Register).

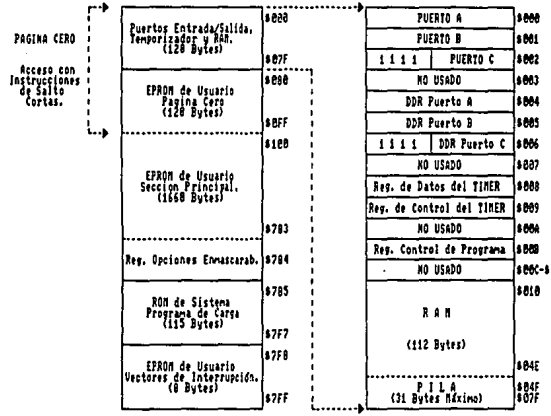
e) 1 byte para el Registro de Control de Programa (PCR, Program Control Register).

f) 6 bytes para los puertos de Entrada/Salida y sus correspondientes Registros de Dirección de Datos (DDR's, Data Direction Registers).

g) 1 byte para el Registro de Datos del Timer (TDR, Timer Data Register).

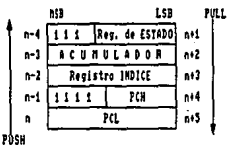
h) 1 byte para el Registro de Control del Timer (TCR, Timer Control Register).

Fig. 1.5.- Mapa de memoria del MC68705P3.



CUIDADO: Los registros de Dirección de Datos (DDR's) son Solo Escritura, por lo que son leídos como $FF.

Fig. 1.6.- Orden de Almacenamiento en la Pila (Interrupción).



NOTA: Durante la llamada a una Subrutina solo PCH y PCL son almacenadas.

En el mapa de memoria se puede apreciar la localización de la EPROM en dos áreas distintas, la primera ubicada de la dirección $080 a la $783, correspondiente a la memoria de usuario y la segunda de la dirección $7F8 a la $7FF correspondiente a los Vectores de Interrupción.

El MCU usa 9 de las 16 localidades más bajas (de los 128 bytes de RAM) para control del programa y tareas de E/S tales como los puertos, los Registros de Dirección de Datos y el Temporizador.

Es importante mencionar que aún cuando el MCU no tiene pila, ni los mnemotécnicos para manejarla, durante saltos a subrutinas aprovecha los 31 bytes más altos de la RAM para almacenar la dirección precedente en una especie de PseudoPila. También durante una requisición de interrupción, se vacía en estas localidades el estado presente del CPU para realizar la subrutina de servicio y al terminar poder regresar al estado anterior y continuar con el flujo normal del programa de usuario. Véase la **Fig. 1.6.**

1.3.3. UNIDAD CENTRAL DE PROCESAMIENTO

La Unidad Central de Procesamiento (CPU) de los procesadores de la familia M6805 se encuentra implementada de manera independiente de los registros de E/S y la memoria. La comunicación con los periféricos se establece mediante los buses internos de Direcciones, Datos y Control. Por lo cual el MCU es internamente un sistema µComputador completo encapsulado en un solo chip.

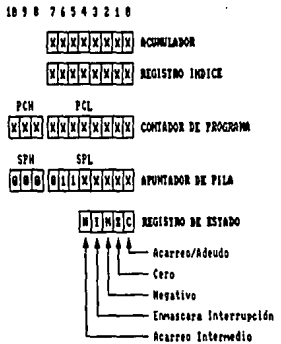
1.3.3.1. Registros.

La **Fig. 1.7** muestra la organización de la CPU cuyos registros se describen en los párrafos siguientes.

ACUMULADOR (A).- El acumulador es un registro de propósito general de 8 bits empleado para almacenar temporalmente operandos y resultados de operaciones aritméticas y de manipulación de datos.

REGISTRO INDICE (X).- El registro índice es un registro de 8 bits empleado en el modo de direccionamiento indexado. El valor

Fig. 1.7.- Registros del MC68705P3.



almacenado en este registro puede ser sumado a otro para generar una dirección efectiva, aún cuando está es su principal aplicación también puede ser usado en la manipulación de datos o como área de almacenamiento temporal en apoyo al acumulador.

CONTADOR DE PROGRAMA (PC).- El contador de programa es un registro de 11 bits, capaz de direccionar 2048 localidades de memoria, el cual contiene la dirección de la siguiente instrucción a ser ejecutada.

APUNTADOR DE PILA (SP).- El apuntador de pila es un registro de 11 bits que contiene la dirección de la siguiente localidad libre en la pila. Los seis bits más significativos se han fijado en 000011. Durante un reestablecimiento del sistema o la ejecución de una instrucción de reestablecimiento del apuntador de pila (RSP, Reset Stack Pointer), el SP se modifica para apuntar a la localidad más alta de la RAM ($07F).

El valor del SP se decrementa antes de la ejecución de una subrutina o servicio de interrupción y se incrementa una vez terminada la serie de instrucciones correspondientes. La pila en RAM esta limitada a 31 bytes, dirección $061, lo cual permite hasta 15 niveles posibles de anidamiento (llamadas a subrutinas e interrupciones).

REGISTRO DE CONDICIONES (CC).- El registro de condiciones es un registro de 5 bits de los cuales cuatro son usados para indicar la naturaleza del resultado de una instrucción ejecutada previamente. Estos bits pueden ser probados individualmente a través de las instrucciones correspondientes y de acuerdo al estado presente llevar a cabo una acción específica. La descripción estos bits se muestra en los siguientes párrafos.

Acarreo Intermedio (H).- Un nivel alto en este bit indica la ocurrencia de un acarreo intermedio entre los bit 3 y 4 durante la ejecución de una instrucción de suma (ADD) o suma con acarreo (ADC).

Interrupción (I).- Cuando este bit se encuentra en un nivel alto, tanto la interrupción externa (INT) como la del temporizador se deshabilitan (se establece una máscara). Si se genera una interrupción mientras este bit está en 1, la interrupción se almacena en buffer y se ejecuta tan pronto como el bit de interrupción se establece a un nivel bajo.

Negativo (N).- Un nivel alto en este bit indica que el resultado de la última operación aritmética, lógica o de manipulación de datos produjo un número negativo (bit 7 igual a 1).

Cero (Z).- Un nivel alto en este bit indica que el resultado de la última operación aritmética, lógica o de manipulación de datos fue cero.

Acarreo/Adeudo (C).- Un nivel alto en este bit indica que el resultado de la instrucción anterior produjo un resultado que excedió el tamaño de palabra del MCU. La señal C puede considerarse como el noveno bit de cualquier resultado aritmético también afectada por instrucciones de desplazamiento y rotación de bits.

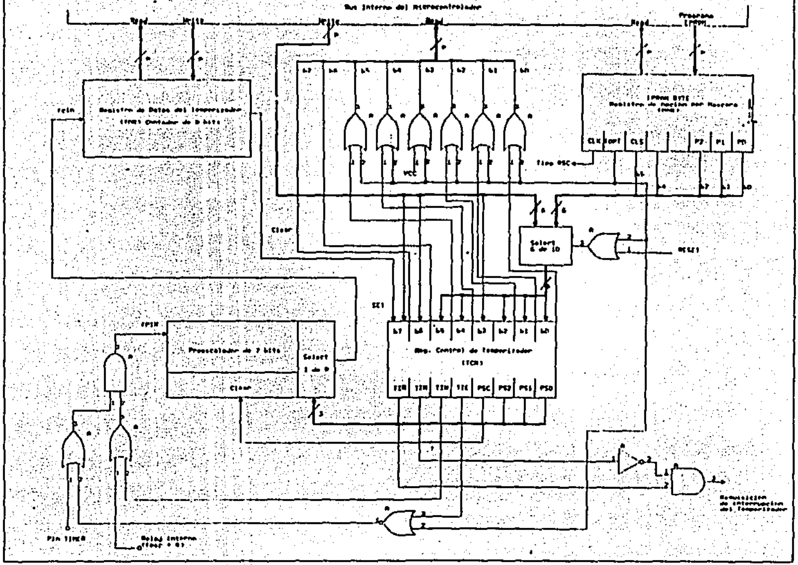
1.3.4. TEMPORIZADOR.

El temporizador del MCU consiste de un contador de 8 bits, programable por el usuario, impulsado por un preescalador de 7 bits con rangos selectivos. Pueden seleccionarse diferentes fuentes de reloj para impulsar el Contador-Preescalador. La selección puede hacerse mediante el Registro de Control del Temporizador (TCR, Timer Control Register) y/o el Registro de Opción por Máscara (MOR, Mask Option Register). El TCR cuenta con los bits necesarios para control de interrupciones.

El circuito del temporizador se muestra en la **Fig. 1.8.** El contador de 8 bits puede ser modificado bajo control del programa cuya cuenta es decrementada hasta cero de acuerdo a la frecuencia fijada por la entrada F_{cin} (procedente del preescalador). Una vez que el contador es decrementado hasta cero, el bit 7 del TCR, Requisición de Interrupción por Temporizador (TIR, Timer Interrup Request), se fija en un nivel alto. Dicha interrupción puede pasar o no al procesador de acuerdo al estado del bit de Habilitación de Interrupción por Temporizador (TIM, Timer Interrup Mask), bit 6. Un nivel alto en el TIM inhibe cualquier interrupción generada por el Temporizador mientras que un nivel bajo la habilita.

Por otra parte, cuando el bit de Interrupción (I) del registro de condiciones (CC) es cero, el procesador acepta la interrupción del temporizador. El CPU responde a esta

Fig. 1.8.- Diagrama de bloque del Temporizador.



interrupción guardando el estado actual del PC en la pila, toma el vector de interrupción correspondiente (Vector de Interrupción del Temporizador en las direcciones $FF8 y $FF9) y ejecuta la rutina establecida. Una vez concluida la rutina de interrupción, el CPU retorna a la posición del programa almacenada en la pila continuando con la ejecución normal del programa.

El procesador es sumamente sensible a la señal de Requisición de Interrupción por Temporizador (TIR) aún cuando se cuenta con una máscara para inhibir dicha señal, por este motivo es aconsejable reestablecer el estado de la señal TIR mediante una instrucción BCLR (por ejemplo, al concluir la rutina de servicio de interrupción), evitando saltos en falso.

El contador continua la cuenta (decrementándose) haciéndolo una vez más desde $FF, o bien desde el valor establecido mediante programa para el Registro de Datos del Timer (TDR, Timer Data Register) hasta cero. El valor del contador puede leerse en cualquier momento sin afectar el proceso de conteo, lo cual es útil para determinar el tiempo faltante mediante programa para que ocurra una requisición de interrupción.

La fuente de reloj puede ser externa (el decremento del contador ocurre durante una transición positiva de la fuente externa) aplicada al pin de entrada TIMER, o bien la señal interna Ò2. Para diversas tareas puede requerirse una fuente "pura" o modulada, en el primero de los casos se usa únicamente la fuente externa o la interna y en el segundo ambas (siendo la fuente interna la moduladora de la fuente externa) según se requiera. La fuente de reloj es seleccionada a través del TCR o el MOR. Véase la **Fig. 1.8** para mayor información.

Cualquiera de las opciones de preescalamiento puede ser aplicada a la entrada de reloj del contador, con lo que puede extenderse el intervalo de la frecuencia hasta 128 pulsos como máximo antes de que el contador se decremente. La opción de preescalamiento selecciona uno de 8 rangos sobre un divisor binario de 7 bits. Para evitar errores de truncamiento, el preescalador es reestablecido cuando el bit b3 de el TCR es igual a 1.

Después de un reestablecimiento, el preescalador y el contador son inicializados a "1s" lógicos, el bit de Requisición de Interrupción por Temporizador (TCR, b7) a "0" lógico y el bit de Habilitación de Interrupción del Temporizador (TIM, b6) a "1" lógico. Los bits b0 hasta b5 del TCR son inicializados a los

valores correspondientes de los bits en el Registro de Opción por Máscara (MOR), los cuales pueden ser modificados posteriormente por programa.

Note que el diagrama a bloques de la **Fig. 1.8** refleja dos posibles configuraciones de control para el temporizador:

1) Control por Programa, mediante el Registro de Control del Temporizador (TCR).

2) Control Programado, mediante el MOR almacenado en EPROM y con posibilidades de emulación para versiones de ROM con máscara (MC6805P2/P4).

En el modo de control por programa, todos los bits del TCR son de lectura/escritura con excepción del bit b3 de solo escritura. En el modo de control vía el MOR, los bit b7 y b6 son de lectura/escritura, el bit b3 es de solo escritura y los otros cinco no tienen efecto (todos 1's).

El bit de Opción del Temporizador (TOPT, b6) en el Registro de Opción con Máscara es programado en EPROM como "0" lógico para seleccionar el modo de control por programa. Los bits del TCR, b5, b4, b3, b2, b1 y b0 dan al programa el control directo del preescalador y las opciones de entrada.

La entrada del temporizador del preescalador (f_{PIN}) puede configurarse para trabajar en tres modos diferentes de operación, más un modo de deshabilitación, dependiendo del valor de los bits de control b4 y b5 (TIE y TIN).

Cuando los bit TIE y TIN son programados a "0", la entrada del temporizador se establece como el reloj interno ($\overline{O2}$) y el pin de entrada TIMER se deshabilita. Este modo de operación puede ser usado para la generación periódica de interrupciones.

Cuando TIE = 1 y TIN = 0, la señal del reloj interno y la señal del pin de entrada TIMER son multiplicadas lógicamente para formar la entrada del temporizador f_{PIN}. Este modo de operación puede ser usado para modular una señal de entrada externa.

Cuando TIE = 0 y TIN = 1, la entrada f_{PIN} no se aplica al preescalador y el temporizador se deshabilita.

Finalmente, cuando TIE y TIN son ambas "1", la entrada del temporizador se establece como el reloj externo. Este modo de operación puede ser usado para el conteo de eventos externos así como para la generación periódica de interrupciones.

Los bit b0, b1 y b2 en el TCR son controlados por programa para seleccionar la salida apropiada del preescalador. El preescalador divide la frecuencia f_{PIN} en múltiplos binarios hasta 128 produciendo la frecuencia f_{PIN} para el contador. El procesador no puede leer del ni escribir en el preescalador.

El modo de control del temporizador vía el MOR se selecciona cuando el bit de Opción del Temporizador (TOPT, b6) en el MOR se programa como "1" lógico, emulando el preescalador programable con máscara del MC6805P2 y MC6805P4. El circuito del temporizador es igual al descrito anteriormente, sin embargo, el Registro de Control del Temporizador (TCR) se configura en forma distinta.

El nivel lógico para los bits b0, b1, b2 y b5 en el TCR son establecidos en el momento que se programa la EPROM, controlados por los bits correspondientes en el Registro de Opción por Máscara (MOR, $F38). Los bits b0, b1, b2 y b5 controlan el preescalador y la selección de la fuente de reloj. El bit b4 (TIE) se encuentra fijo en "1" lógico. El bit b3, en el modo controlado por el MOR, siempre se lee como "0" lógico y puede ser fijado a "1" para limpiar el preescalador. Los bits b7 y b6 funcionan en la misma forma que lo hacen en el modo de control por programa.

1.3.5. REESTABLECIMIENTO DEL MCU.

El MCU puede ser reestablecido de dos maneras: mediante el encendido del sistema o a través de una señal externa aplicada a la terminal RESET. Durante el encendido se requiere un retardo de t_{RHL} segundos antes de que la entrada RESET se reestablezca a un nivel alto. Esto proporciona el tiempo suficiente para que el reloj interno y el sistema se estabilicen. Mediante la conexión de un capacitor en la entrada RESET y tierra (tal y como se aprecia en la **Fig. 1.9**) se obtiene dicho retardo.

El circuito interno conectado a la terminal $\overline{\text{RESET}}$ consiste en un disparador Schmitt, el cual checa el nivel lógico de la

Fig. 1.9.- Circuito para RESET Inicial (Power Up)

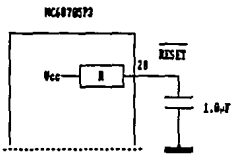
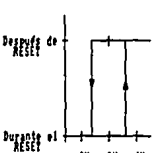


Fig. 1.10.- Histéresis Típico del disparador Schmitt del RESET.



entrada. Cuando se detecta un nivel lógico "0" se produce un reestablecimiento del sistema. En la **Fig. 1.10** se muestra una curva de histéresis típica del disparador y en la **Fig. 1.14** un diagrama completo del proceso de reestablecimiento.

1.3.6. OPCIONES DE GENERACION INTERNA DE RELOJ

El circuito oscilador interno del MCU está diseñado para requerir un mínimo de componentes externos. Un Cristal, una resistencia, un alambre o un generador externo pueden ser usados como fuente de reloj del sistema. El Registro de Opción por Máscara (EPROM) se programa para seleccionar un cristal o una resistencia como fuente de reloj. La frecuencia del oscilador se divide internamente entre cuatro para producir el sistema de reloj interno.

En la **Fig. 1.11** se muestran los diferentes métodos de conexión, en los cuales hay que considerar los siguientes puntos:

1) Cuando el pin TIMER/BOOT está en el rango de V_{IHTP} (Modo de Programación de la EPROM) la opción de cristal es obligada. Para cualquier otro voltaje la opción del generador de reloj está determinada por el bit 7 del Registro de Opción por Máscara (CLK).

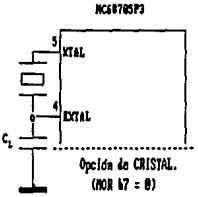
2) El valor recomendado para C_L con un cristal de 4 MHz es de 27 pF como máximo. Existe una capacitancia aproximada de 25 pF en XTAL. Para frecuencias diferentes a 4 MHz, la capacitancia para cada pin deberá ser calculada como el inverso de la frecuencia. Por ejemplo, con un cristal de 2 MHz, deberá emplearse un capacitor de 50 pF para EXTAL y uno de 25 pF para XTAL. El valor exacto depende de los parámetros del cristal empleado.

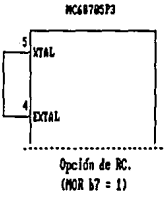
Las especificaciones para un oscilador de cristal y una resistencia están dadas en las **Fig. 1.12** y **1.13** respectivamente.

1.3.7. ROM DE PROGRAMACION.

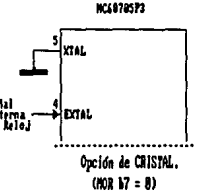
La memoria ROM de arranque (BOOTSTRAP ROM) contiene un programa de fábrica el cual permite al MCU realizar una búsqueda

Fig. 1.11.- Opciones de Generación de Reloj.





NOTA: t_{cyc} típico = 1.25μseg.



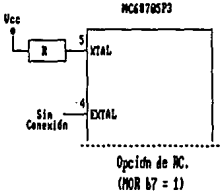
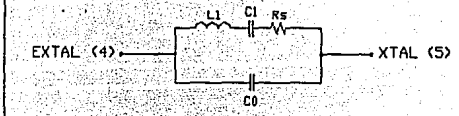
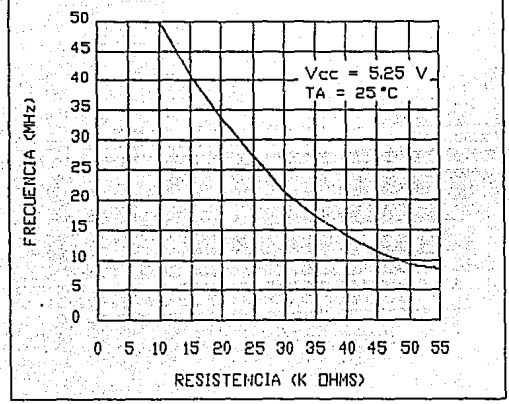


Fig. 1.12.- Circuito Equivalente de Cristal.



C0 = 7 pF Max.
Frec. = 4.0 MHz & Cl = 27 pF
Rs = 100 ohms Max.

Fig. 1.13.- Selección de la frecuencia típica mediante una resistencia.



de datos desde un dispositivo externo y transferirlos hacia la EPROM del MC68705P3. El programa incluido en la ROM de arranque establece los pulsos de tiempo necesarios para la programación, el tiempo de entrada para Vpp y la verificación de los datos de programación de la EPROM. Más adelante se abordará más profundamente el tema en la sección denominada PROGRAMACION DEL FIRMWARE.

1.3.8. REGISTRO DE OPCION POR MASCARA (MOR).

El Registro de Opción por Máscara (MOR, Mask Option Register), consiste de un registro de 8 bits programable por el usuario (EPROM) del cual sólo seis bits son usados. Los bits de este registro son empleados para seleccionar el tipo de reloj del sistema, la opción del temporizador, la fuente de reloj del temporizador/preescalador y la opción del preescalador.

1.3.9. INTERRUPCIONES.

Una interrupción en el MCU MC68705P3 puede llevarse a cabo de tres formas distintas: a través de el pin de entrada de interrupción externa (INT), a través de una requisición de interrupción por temporizador, o bien, mediante una instrucción de interrupción por programa (SWI). Cuando se suscita una interrupción la instrucción corriente se concluye (incluyendo SWI) y el proceso normal se suspende, el estado actual del CPU se guarda en la pila, el bit de interrupción en el Registro de Condiciones se establece en "1", la dirección de la rutina de interrupción se obtiene del vector de interrupción correspondiente y la rutina de interrupción se ejecuta. Todo este proceso requiere de 11 períodos t_{cyc} para completarse. Un diagrama de flujo acerca de este proceso se muestra en la **Fig. 1.14.**

La rutina de servicio de interrupción deberá finalizar con una instrucción de retorno de interrupción (RTI) la cual permite al CPU continuar con la ejecución normal del programa. La **tabla 1** muestra los diferentes tipos de interrupciones, el grado de prioridad de las mismas y la dirección del vector que contiene la dirección de inicio de la rutina correspondiente de interrupción. La prioridad de las interrupciones se aplica para aquellas que están pendientes cuando el CPU está listo para aceptar una nueva interrupción. Cuando el bit I en el Registro de Condiciones se

Fig. 1.14.- Procedimiento de interrupción y reset.

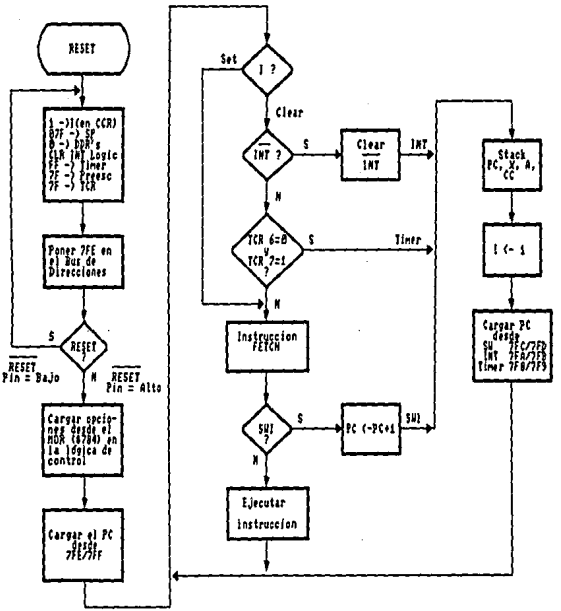
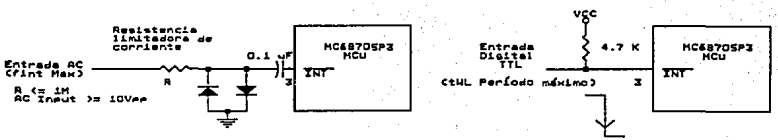


Tabla 1 - Niveles de interrupción.

Interrupción	Prioridad	Vector de dirección
$\overline{\text{RESET}}$	1	07FE y 07FF
SWI	2 *	07FC y 07FD
$\overline{\text{INT}}$	3	07FA y 07FB
TIMER	4	07F8 y 07F9

* La prioridad 2 se aplica solamente cuando el bit I del Registro de Condiciones se encuentra en 1. Si I=0, SWI tendrá prioridad 4 mientras que $\overline{\text{INT}}$ y TIMER tendrán prioridad 2 y 3 respectivamente.

Fig. 1.15.- Circuitos de interrupción típicos.



a.- Interrupción empleando un detector de cruce por cero.

b.- Interrupción con señal digital

establece en "1", la interrupción es almacenada para su posterior ejecución.

La interrupción externa es internamente sincronizada y aceptada en el flanco de bajada de INT. Una señal de entrada sinusoidal (f_{INT} máxima) se puede usar para generar una interrupción externa, como se muestra en la **Fig. 1.15a**, mediante el uso de un detector de cruce por cero. Para aplicaciones digitales, el pin INT puede ser impulsado por una señal digital con un período máximo de t_{WL} como se muestra en la **Fig. 1.15b.**

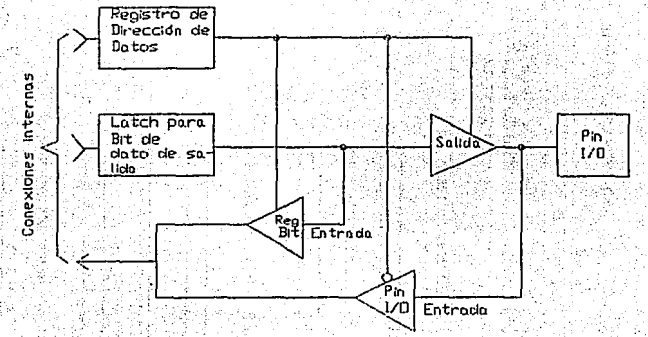
Una interrupción por software (SWI) es una instrucción ejecutable la cual se lleva a cabo no importando el estado del bit I en el Registro de Condiciones. La instrucción SWI se emplea generalmente como punto de ruptura o llamadas al sistema.

1.3.10. ENTRADA/SALIDA.

Existen 20 pines de entrada/salida. Todos los pines (puertos A, B y C) son programables ya sea como entradas o salidas bajo control de programa de el correspondiente Registro de Dirección de Datos (DDR, Data Direction Register). La programación de los puertos de entrada/salida se complementa por la escritura del bit correspondiente en el DDR del puerto, "1" para salida y "0" para entrada. Durante un reestablecimiento del sistema todos los DDR's son inicializados a un nivel lógico "0" estableciendo los puertos en modo de entrada. Los registros de salida de los puertos no son inicializados durante el reestablecimiento pero pueden ser establecidos al mismo nivel que sus correspondientes DDR's con el fin de evitar niveles indefinidos. Cuando se programa un puerto como salida, el dato de salida es también leído como un dato de entrada a pesar del nivel lógico en el pin de salida tal y como se muestra en la **Fig. 1.16.**

Todas las líneas de entrada/salida son compatibles con TTL ya sea como entradas o salidas. Las líneas del puerto A son compatibles con CMOS como salidas mientras que el puerto B y C son compatibles con CMOS como entradas. El mapa de memoria en la figura 6 muestra las direcciones de los registros de datos y de sus correspondientes registros de direcciones (DDR's). La **Fig. 1.17** muestra la configuración de los registros y la **Fig. 1.18** algunos ejemplos de conexión de los puertos.

Fig. 1.16.- Circuito típico para puertos de E/S.



Bit del Registro de Dirección de Datos	Bit de Datos de Salida	Estado de la Salida	Entrada al MCU
1	0	0	0
1	1	1	1
0	X	Triestado	Pin

Fig. 1.17.- Configuración de registros del MC68705P3.

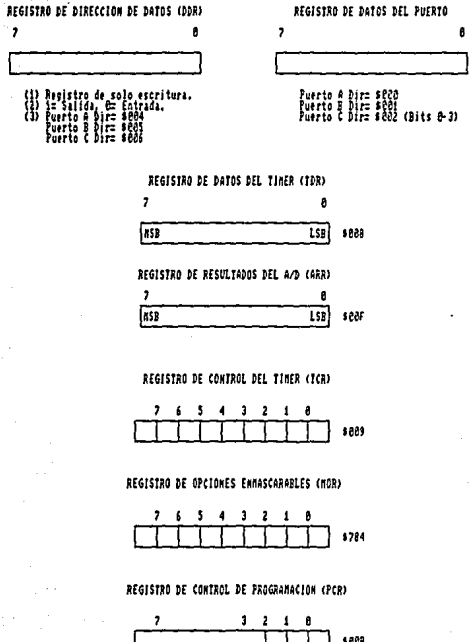
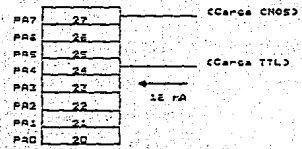
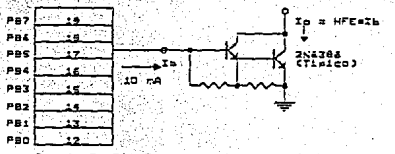


Fig. 1.18.- Conexion tipica de los puertos.

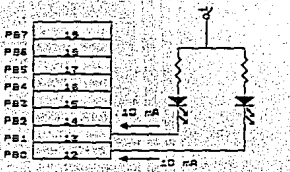
(a) Modos de Salida



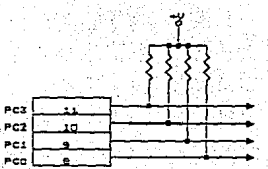
Puerto A. Bit 7 programado como salida.



Puerto B. Bit 5 programado como salida.



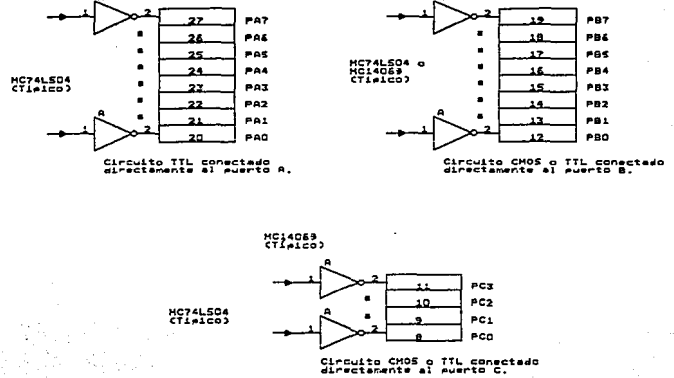
Puerto B. Bit 0 y Bit 1 programado como salida.



Puerto C. Bits 0-3 programados como salida.

Fig. 1.18.- Conexion típica de los puertos.

(b) Modos de Entrada



Los DDR's de los puertos A, B y C son registros de sólo escritura (registros en $004, $005 y $006). Una operación de lectura para estos registros no está definida. Dado que BSET y BCLR son funciones de lectura/modificación/escritura, no pueden ser empleadas para poner o limpiar un simple bit del DDR. Es recomendable que todos los bits de los DDR's se inicialicen utilizando una instrucción de almacenamiento simple.

1.3.11. REGISTRO DE CONTROL DEL TEMPORIZADOR (TCR).

La configuración del Registro de Control del Temporizador (TCR, Timer Control Register) está determinada por el nivel lógico de el bit 6 (Opción del Timer, TOPT) en el Registro de Opción por Máscara (MOR). En la **Fig. 1.19** se muestran dos configuraciones del TCR, una para TOPT = 1 y la otra para TOPT = 0. TOPT = 1 configura el TCR para emular el MC6805P2 o MC6805P4. Cuando TOPT = 1, las opciones del preescalador son programables por el usuario vía el MOR. En los párrafos siguientes se da una descripción de cada bit del TCR.

b7, TIR Requisición de Interrupción por Temporizador.
Este bit se usa para iniciar una interrupción vía el temporizador o señalar un estado de todos ceros en el Registro de Datos del Temporizador cuando es igual a "1".

1 = Fijado cuando el Registro de Datos del Temporizador (TDR) cambia a todos ceros.
0 = Fijado mediante un reestablecimiento o bajo control de programa.

b6, TIM Habilitación de Interrupción del Temporizador.
Este bit se usa para deshabilitar la interrupción por temporizador cuando es igual a "1".

1 = Fijado mediante un reestablecimiento externo o bajo control de programa.
0 = Fijado mediante control de programa.

b5, TIN Externo o Interno.
Este bit se usa para seleccionar la fuente de reloj de entrada, ya sea externa (pin 7, TIMER) o interna (Ò2).

Fig. 1.19.- Registros de Control del Timer

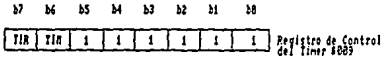


Fig. 1.20.- Modos TIN-TIE.

TIN	TIE	CLOCK
0	0	Reloj interno (o2)
0	1	Operacion AND de reloj interno y externo
1	0	Reloj deshabilitado
1	1	Reloj externo

Fig. 1.21.- División del Preescalador.

PS2	PS1	PS0	División del Preescalador
0	0	0	1 (Preescalador Bypass)
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

1 = Selecciona la fuente externa.
0 = Selecciona la fuente interna $\div 2$ (fosc+4).

b4, TIE Habilitación externa.
Este bit se usa para habilitar el pin 7 (TIMER, temporizador externo), o para habilitar el reloj interno (si TIN=0) sin importar el estado del pin del temporizador externo. Cuando TOPT=1, TIE se fija automáticamente en "1". Véase la **Fig. 1.20.**

1 = Habilita el pin del temporizador externo.
0 = Deshabilita el pin del temporizador externo.

b3, PSC Borrado del Preescalador.
Este es un bit de sólo escritura. Se lee como cero lógico (cuando TOPT=0), por lo que BSET y BCLR funcionan correctamente sobre el TCR. Cuando PSC = 1 se genera un pulso que borra el preescalador.
Cuando TOPT = 1 este bit permanece en "1" lógico y no tiene efecto sobre el preescalador.

b2, PS2
b1, PS1
b0, PS0 Selección del preescalador.
Estos bits son decodificados para seleccionar uno de los ocho rangos del preescalador.
En la **Fig. 1.21** se muestran los resultados de la división del preescalador por decodificación de estos bits.

Cuando los bits PS2-0 son modificados por programa, el PSC deberá de ponerse en "1" en el mismo ciclo de escritura. Cambiar los bits PS sin limpiar el preescalador puede causar problemas en el Registro de Datos del Temporizador.

1.3.12. OPCIONES POR MASCARA.

El Registro de Opción por Máscara del MC68705P3 está implementado en la EPROM. Como todos los bytes de la EPROM, el MOR mantiene todos sus bits en ceros antes de su programación.

Cuando se usa para emular el MC6805P2 o MC6805P4, cinco de los ocho bits del MOR son usados en conjunto con el preescalador. De los restantes, el bit b7 es usado para seleccionar el tipo de reloj oscilador, y los bits b3 y b4 no son usados. Los bits b0, b1 y b2 determinan la división de el preescalador del temporizador y el bit b5 determina la fuente de reloj. El valor de el bit TOPT (b6) se programa para configurar el TCR (un "1" lógico para emulación del MC6805P2/P4).

Si el TOPT = 0, los bits b5, b4, b2, b1 y b0 fijan el valor de los bits correspondientes en el TCR durante un reestablecimiento. Después de la inicialización el TCR es controlable por software.

Descripción de los bits del MOR, Véase la **Fig. 1.22**:

b7, CLK Tipo de Reloj Oscilador.

1 = RC
0 = Cristal

V_{INTP} en el pin (7) TIMER/BOOT forza a modo de cristal.

b6, TOPT Opción del Timer.

1 = Temporizador/preescalador tipo MC6805P2/P4. Todos los bits, excepto el 6 y 7, de el TCR no son accesibles al usuario. Los bits 5, 2, 1 y 0 del MOR determinan la opción enmascarable equivalente al MC6805P2/P4.

0 = Todos los bits del TCR son implementados por programa. El estado de los bits del MOR b5, b4, b2, b1 y b0 establecen los valores iniciales de sus respectivos bits en el TCR. El TCR es controlado por programa después de la inicialización.

b5, CLS Fuente de reloj del Temporizador/preescalador.

1 = Pin del temporizador externo (TIMER).
0 = Reloj interno *Ò2*.

Fig. 1.22.- Registro del MOR.

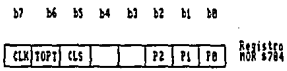


Fig. 1.23.- División del Preescalador.

PS2	PS1	PS0	División del Prescalador
0	0	0	1 (Preescalador Bypass)
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

b4 No usado si TOPT=1 (emulación del MC6805P2/P4). Se establece igual al valor del TIE si TOPT=0.

b3 No usado.

b2, P2
b1, P1
b0, P0 Opciones del preescalador.
Los niveles lógicos de estos bits, una vez decodificados, seleccionan uno de los ocho rangos del preescalador. Véase la **Fig. 1.23**.

1.3.13. SOFTWARE PROGRAMADO EN CHIP.

El Registro de Control de Programación (PCR, Programming Control Register) en la localidad $00B es un registro de 8 bits de los cuales sólo se emplean los tres menos significativos (los más significativos están en "1"). Este registro provee los bits de control necesarios para la programación de la EPROM de el MC68705P3. El programa de arranque de programación manipula el PCR.

Descripción de los bits del PCR, Véase la **Fig. 1.24a**:

b0, $\overline{\text{PLE}}$ Habilitación de la programación.
Cuando este bit es cero, permite que la dirección y el dato sean cargados en la EPROM. Cuando es igual a "1" los datos pueden ser leídos desde la EPROM.

1 = Leer EPROM.

0 = Direcciones y datos son cargados en la EPROM (Lectura deshabilitada).

Durante un reestablecimiento, el bit $\overline{\text{PLE}}$ se cambia a "1" pero puede ser establecido a cero en cualquier momento. Sin embargo, su efecto sobre la EPROM es inhibido si VPON es igual a "1".

Fig. 1.24.- Registro del PCR.

b7	b6	b5	b4	b3	b2	b1	b0
1	1	1	1	1	$\overline{VPON}$	$\overline{PGE}$	$\overline{PLE}$

Registro de control de programación $9008

(a)

$\overline{VPON}$	$\overline{PGE}$	$\overline{PLE}$	Condiciones de Programación
0	0	0	Modo de programación de la EPROM
1	0	0	$\overline{PGE}$ y $\overline{PLE}$ deshabilitados desde el sistema
0	1	0	Programación deshabilitada (latch de direcciones y datos en EPROM)
1	1	0	$\overline{PGE}$ y $\overline{PLE}$ deshabilitados desde el sistema
0	0	1	Estado inválido, $\overline{PGE}$ = 0 si $\overline{PLE}$ = 0
1	0	1	Estado inválido, $\overline{PGE}$ = 0 si $\overline{PLE}$ = 0
0	1	1	Voltaje alto en Vpp
1	1	1	$\overline{PGE}$ y $\overline{PLE}$ deshabilitados desde el sistema (Modo de Operación)

(b)

b1, $\overline{\text{PGE}}$ Habilitación de programa.
Cuando PGE = 0, se habilita la programación de la EPROM. El bit PGE puede ser limpiado solamente cuando PLE = 0. PGE deberá tener un valor "1" cada vez que se cambia la dirección y el dato, por ejemplo al poner el byte a ser programado.

1 = Inhibe la programación de la EPROM.

0 = Habilita la programación de la EPROM (si PLE = 0).

PGE se establece en "1" durante un reestablecimiento del sistema, sin embargo, no tiene efecto sobre la EPROM si VPON es igual a "1".

b2, $\overline{\text{VPON}}$ (Vpp ON) $\overline{\text{VPON}}$ es un bit de sólo lectura y cuando está en "0" lógico indica que un voltaje alto está presente en el pin Vpp.

1 = Ausencia de un voltaje alto.

0 = Voltaje alto en el pin Vpp.

Cuando $\overline{\text{VPON}}$ = 1 se inhibe el efecto de $\overline{\text{PGE}}$ y $\overline{\text{PLE}}$. Véase la **Fig. 1.24b.**

1.3.14. BORRADO DE LA EPROM.

La EPROM del MC68705P3 puede ser borrada por la exposición a la luz ultravioleta (UV) con una longitud de onda de 2537 Å. La dosis recomendada (Intensidad de UV x tiempo de exposición) es de 15 Ws/cm^2. Las lámparas deberán ser usadas sin filtros de onda corta y el MC68705P3 deberá estar colocado a una pulgada de los tubos de UV.

1.3.15. PROGRAMACION DEL FIRMWARE.

El MC68705P3 tiene 115 bytes de ROM enmascarable que contiene el programa de arranque de programación de la EPROM. El vector en la dirección $7F6 y $7F7 se usa para empezar la ejecución del programa. El valor de este vector se carga cuando

V_{IHTP} es aplicado al pin 7 (TIMER/BOOT) de el MC68705P3 y el pin RESET está cerca de alcanzar $V_{IRES}+$. La **Fig. 1.25** muestra el diagrama básico de programación de la EPROM del MCU donde se especifican cada uno de los pasos a seguir.

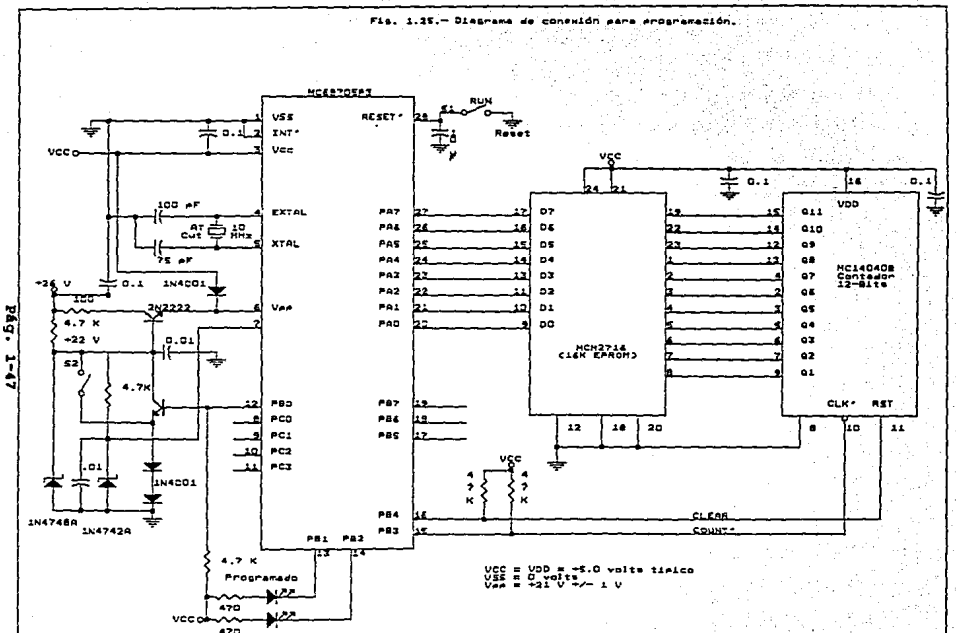
1.3.16. PASOS DE PROGRAMACION DE LA EPROM.

La EPROM MCM2716 UV deberá programarse primero con un duplicado exacto de la información que será transferida al MC68705P3. Dado que el MC68705P3 y el MCM2716 serán insertados y quitados del circuito deberán ser montados en bases. Además se debe de tener la siguiente precaución: asegurarse de que S1 y S2 estén cerrados y que tanto Vcc como +26V no estén conectadas cuando se vaya a insertar el MC68705P3 y el MCM2716 en sus respectivas bases. Esto asegura que el RESET permanezca bajo mientras inserta los dispositivos.

Hecho lo anterior, sólo basta aplicar Vcc y +26V, abrir el switch S2 (para aplicar Vpp y V_{IHTP}), y después S1 (para remover reset). Una vez que los voltajes son aplicados y ambos interruptores han sido abiertos, la línea de control de salida CLEAR (PB4) se encarga de reiniciar el contador y la salida PB3 (COUNT) de fijar los pulsos de reloj. El contador selecciona la dirección del byte de la EPROM del MCM2716 a cargar en la dirección correspondiente de la EPROM del MC68705P3 establecida por el programa de arranque. Una vez que la localidad de la EPROM ha sido cargada, COUNT impulsa el contador a la siguiente dirección. Este proceso continua hasta que el MC68705P3 ha sido programado completamente a la vez que el led indicador de terminación de programación se enciende. El contador se reinicia y el ciclo se repite, esta vez para verificar los datos programados. El led de verificación se enciende si la programación fue exitosa.

Una vez que el MC68705P3 ha sido programado y verificado, es necesario cerrar el interruptor S2 (para remover Vpp y V_{IHTP}) y S1 (para reiniciar el integrado). Finalmente sólo queda desconectar +26V y Vcc y desmontar el MC68705P3 de su base.

Fig. 1.25.- Diagrama de conexión para programación.



Proceso de programación:

1 Antes de colocar el MC68705P3 y la memoria MCM2716, asegúrese de que tanto S1 como S2 están cerrados y que VCC y +26 V no están aplicados.
2 Para iniciar la programación, asegúrese de que tanto S1 como S2 están cerrados, aplique entonces VCC y +26 V. Ya aplicada la alimentación abra S2 seguido de S1.
3 Antes de remover el MC68705P3, primero cierre S2 y después S1, desconecte VCC y +26 V.

1.3.17. SOFTWARE INTERNO.

1.3.17.1. Manipulación de bits.

El MC68705P3 MCU posee la capacidad de fijar el valor de un simple bit de la RAM o de algún puerto de entrada/salida mediante una instrucción simple (BSET o BCLR). El estado de cualquier bit en la página cero de memoria puede ser examinado, usando las instrucciones BRSET y BRCLR, llevando a cabo un brinco por parte del programa de acuerdo al estado en que se encuentre dicho bit. Una instrucción de rotación puede usarse para acumular datos de entrada en serie en una localidad de la RAM o registro. Estas capacidades permiten la posibilidad de trabajar con bits en RAM, ROM o en los puertos de entrada/salida como banderas individuales o bien para usarse como líneas de control de entrada/salida de datos.

El ejemplo de la **Fig. 1.26** ilustra el uso de la manipulación de bits.

Haga de cuenta que el MCU se comunica con un dispositivo serie externo. El dispositivo externo cuenta con una señal de dato listo, una línea de salida de datos, y una línea de reloj para impulsar los datos un bit a la vez. El bit menos significativo (LSB) es el primero en salir del dispositivo.

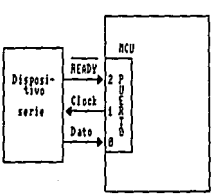
El MCU espera a que el dato esté listo, impulsa al dispositivo externo, recoge el dato en el bit de acarreo, limpia la línea de reloj y finalmente guarda el bit de dato en una localidad de la RAM.

1.3.17.2. Modos de direccionamiento.

El MC68705P3 tiene 10 modos de direccionamiento los cuales se explican en los siguientes párrafos.

El término dirección efectiva (EA, Effective Address) se emplea en la descripción de los modos de direccionamiento. La Dirección Efectiva se define como la dirección desde la cual, un operando para una instrucción, es traído o almacenado.

Fig. 1.26.- EJEMPLO DE MANIPULACION DE BITS.




```
SELF   BRSET 2, PUERTO, SELF
       .
       .
       .
       BSET  1, PUERTO
       BRCLR 0, PUERTO, CONT
CONT   BCLR  1, PUERTO
       ASR   AANLOC
```

INMEDIATO.- En el modo de direccionamiento inmediato, el operando está contenido en el byte siguiente al código de operación. Se usa para accesar constantes las cuales no cambian durante la ejecución del programa (por ejemplo una constante usada para inicializar un contador en un ciclo).

DIRECTO.- En el modo de direccionamiento directo, la dirección efectiva del operando está contenida en un byte simple que sigue al código de operación. Este modo permite direccionar los 256 bytes más bajos en memoria con una instrucción simple de dos bytes. Esta área de direccionamiento incluye la RAM, los registros de entrada/salida y los 128 bytes de EPROM.

EXTENDIDO.- En el modo de direccionamiento extendido, la dirección efectiva del operando está contenida en los dos bytes que siguen al código de operación. Las instrucciones que usan este tipo de direccionamiento son capaces de referenciar operandos en cualquier parte de la memoria con una instrucción simple de 3 bytes.

RELATIVO.- El modo de direccionamiento relativo se utiliza sólo en instrucciones de brinco. El contenido del byte con signo que sigue al código de operación (el desplazamiento) se suma al PC, si y solamente si, la condición de brinco es verdadera. En otro caso, el control procede con la siguiente instrucción. El alcance de este modo se encuentra entre -126 y +129 desde la dirección del código de operación. El programador no necesita preocuparse del cálculo correcto del desplazamiento ya que el ensamblador de Motorola lo hace automáticamente.

INDEXADO, SIN DESPLAZAMIENTO.- En este modo de direccionamiento, la dirección efectiva del operando está contenida en el registro índice. De esta forma se puede acceder a las primeras 256 localidades de memoria. Estas instrucciones son de un solo byte. Este modo de direccionamiento se usa con frecuencia para mover un apuntador en una tabla o para almacenar la dirección de una localidad de RAM o de entrada/salida frecuentemente referenciada.

INDEXADO, CON DESPLAZAMIENTO DE 8 BITS.- En este caso la dirección efectiva es la suma del byte sin signo del registro índice y el byte sin signo que sigue al código de operación. Se usa en la selección de un k-ésimo elemento en una tabla de n elementos. Con este tipo de instrucciones de 2 bytes, k deberá estar contenido en X con la dirección de inicio de la tabla contenida en la instrucción. Como tales tablas pueden empezar en

cualquiera de las 256 primeras localidades direccionables su alcance puede extenderse hasta la localidad 511 ($1FE).

INDEXADO CON DESPLAZAMIENTO DE 16 BITS.- La dirección efectiva es la suma de los contenidos del registro índice y los dos bytes sin signo que siguen al código de operación. Es similar al anterior, excepto que estas instrucciones de 3 bytes permiten tablas con longitud equiparable a toda la memoria.

BIT SET/CLEAR.- En este caso el bit a ser puesto o limpiado es parte del código de operación y el byte que sigue al código de operación especifica la dirección directa del byte en el cual se encuentra el bit especificado. De esta manera cualquier bit de lectura/escritura en las primeras 256 localidades de memoria, incluyendo los puertos de entrada/salida, puede ser puesto o limpiado con una instrucción simple de 2 bytes.

CHEQUEO DE BIT Y BRINCO.- Es una combinación del direccionamiento directo y relativo. El bit checado y la condición (set o clear) está incluida en el código de operación, y la dirección del byte a checar es el byte que sigue al código de operación. El desplazamiento relativo con signo está en el tercer byte y es sumado al valor del PC si la condición de brinco es verdadera. Esta instrucción de 3 bytes permite al programa brincos basados en la condición de algún bit de lectura en las primeras 256 localidades de memoria. El alcance es de -125 a +130 desde la dirección del código de operación. El estado del bit checado se transfiere al bit de acarreo del Registro de Condiciones.

INHERENTE.- Toda la información necesaria para ejecutar la instrucción está contenida en el código de operación. Estas operaciones se llevan a cabo solamente sobre el registro índice o el acumulador así como las instrucciones de control que no necesitan argumentos por lo que sólo requieren de 1 byte de longitud.

1.3.17.3. Instrucciones de registros y memoria.

El MCU cuenta con 59 instrucciones básicas, que al combinarse con los 10 modos de direccionamiento producen 207 códigos de operación. Un operando puede ser el acumulador o el registro índice. El otro operando se obtiene de la memoria mediante algún modo de direccionamiento. El brinco incondicional

(JMP) y el brinco a subrutina (JSR) no tienen operandos. Véase la **tabla 2**.

1.3.17.4. Instrucciones de lectura, modificación y escritura.

Estas instrucciones leen una localidad de memoria o un registro, modificando o examinando su contenido y escribiendo el resultado en la misma localidad o registro. La instrucción para negativo o cero está incluida dentro de esta categoría. Véase la **tabla 3**.

1.3.17.5. Instrucciones de brinco.

Las instrucciones de brinco causan que el programa realice un brinco cuando cierta condición es detectada. Véase la **tabla 4**.

1.3.17.6. Instrucciones de manipulación de bits.

Este tipo de instrucciones son usadas sobre algún bit específico localizado en los primeros 256 bytes de memoria. Un grupo de bits puede ser fijado a todos "1's" o a todos "0's" o bien, dependiendo del estado de algún bit ubicado en una localidad de memoria específica realizar un brinco. Véase la **tabla 5**.

1.3.17.7. Instrucciones de control.

Las instrucciones de control controlan las operaciones del MCU durante la ejecución del programa. Véase la **tabla 6**.

En la **tabla 7** puede consultarse el juego completo de instrucciones del MC68705P3.

TABLA 2.- INSTRUCCIONES PARA MANEJO DE REGISTROS Y MEMORIA

		Modos de direccionamiento																		
		Inmediato			Directo			Extendido			Indexado (No Offset)			Indexado (8 bits Offset)			Indexado (16 bits Offset)		
Función	Mnemo	Código Op.	# Bytes	# Ciclos	Código Op.	# Bytes	# Ciclos	Código Op.	# Bytes	# Ciclos	Código Op.	# Bytes	# Ciclos	Código Op.	# Bytes	# Ciclos	Código Op.	# Bytes	# Ciclos
Cargar A desde memoria	LDA	A6	2	2	B6	2	4	C6	3	5	F6	1	4	E6	2	5	D6	3	6
Cargar X desde memoria	LDX	AE	2	2	BE	2	4	CE	3	5	FE	1	4	EE	2	5	DE	3	6
Almacenar A en memoria	STA	-	-	-	B7	2	5	C7	3	6	F7	1	5	E7	2	6	D7	3	7
Almacenar X en memoria	STX	-	-	-	BF	2	5	CF	3	6	FF	1	5	EF	2	6	DF	3	7
Sumar memoria a A	ADD	AB	2	2	BB	2	4	CB	3	5	FB	1	4	EB	2	5	DB	3	6
Sumar memoria y acarreo a A	ADC	A9	2	2	B9	2	4	C9	3	5	F9	1	4	E9	2	5	D9	3	6
Sustraer memoria	SUB	A0	2	2	B0	2	4	C0	3	5	F0	1	4	E0	2	5	D0	3	6
Sustraer memoria de A con prestamo	SBC	A2	2	2	B2	2	4	C2	3	5	F2	1	4	E2	2	5	D2	3	6
Memoria AND A	AND	A4	2	2	B4	2	4	C4	3	5	F4	1	4	E4	2	5	D4	3	6
Memoria OR A	ORA	AA	2	2	BA	2	4	CA	3	5	FA	1	4	EA	2	5	DA	3	6
Memoria OR exclusiva A	EOR	A8	2	2	B8	2	4	C8	3	5	F8	1	4	E8	2	5	D8	3	6
Comparación aritmética de A con memoria	CMP	A1	2	2	B1	2	4	C1	3	5	F1	1	4	E1	2	5	D1	3	6
Comparación aritmética de X con memoria	CPX	A3	2	2	B3	2	4	C3	3	5	F3	1	4	E3	2	5	D3	3	6
Comparación lógica de A con Memoria	BIT	A5	2	2	B5	2	4	C5	3	5	F5	1	4	E5	2	5	D5	3	6
Brinco Incondicional	JMP	-	-	-	BC	2	3	CC	3	4	FC	1	3	EC	2	4	DC	3	5
Brinco a Subrutina	JSR	-	-	-	BD	2	7	CD	3	8	FD	1	7	ED	2	8	DD	3	9

TABLA 3.- INSTRUCCIONES DE LECTURA, ESCRITURA Y MODIFICACION

		Modos de direccionamiento														
		Inherente (A)			Inherente (X)			Directo			Indexado (No Offset)			Indexado (8 bits Offset)		
Función	Mnemo	Código Op.	# Bytes	# Ciclos	Código Op.	# Bytes	# Ciclos	Código Op.	# Bytes	# Ciclos	Código Op.	# Bytes	# Ciclos	Código Op.	# Bytes	# Ciclos
Incremento	INC	4C	1	4	5C	1	4	3C	2	6	7C	1	6	6C	2	7
Decremento	DEC	4A	1	4	5A	1	4	3A	2	6	7A	1	6	6A	2	7
Clear	CLR	4F	1	4	5F	1	4	3F	2	6	7F	1	6	6F	2	7
Complemento	COM	43	1	4	53	1	4	33	2	6	73	1	6	63	2	7
Complemento a 2 Negación	NEG	40	1	4	50	1	4	30	2	6	70	1	6	60	2	7
Rotación a la izquierda (C)	ROL	49	1	4	59	1	4	39	2	6	79	1	6	69	2	7
Rotación a la derecha (C)	ROR	46	1	4	56	1	4	36	2	6	76	1	6	66	2	7
Desplazamiento lógico a la izquierda	LSL	48	1	4	58	1	4	38	2	6	78	1	6	68	2	7
Desplazamiento lógico a la derecha	LSR	44	1	4	54	1	4	34	2	6	74	1	6	64	2	7
Desplazamiento aritmético a la derecha	ASR	47	1	4	57	1	4	37	2	6	77	1	6	67	2	7
Test para negativo o cero	TST	4D	1	4	5D	1	4	3D	2	6	7D	1	6	6D	2	7

TABLA 4.- INSTRUCCIONES DE BRINCO.

FUNCION	MNEMOTECNICO	Modo de direccionamiento relativo		
		CODIGO OP.	# BYTES	# CICLOS
Salto incondicional	BRA	20	2	4
Salto no habilitado	BRN	21	2	4
Salto si es mayor	BHI	22	2	4
Salto si es menor o igual	BLS	23	2	4
Salto si no hay acarreo	BCC	24	2	4
(Salto si es mayor o igual)	(BHS)	24	2	4
Salto si hay acarreo	BCS	25	2	4
(Salto si es menor)	(BLO)	25	2	4
Salto si no es igual	BNE	26	2	4
Salto si es igual	BEQ	27	2	4
Salto si no hay acarreo intermedio	BHCC	28	2	4
Salto si hay acarreo intermedio	BHCS	29	2	4
Salto si es positivo	BPL	2A	2	4
Salto si es negativo	BMI	2B	2	4
Salto si bit de interrupción clear	BMC	2C	2	4
Salto si bit de interrupción set	BMS	2D	2	4
Salto si línea de interrupción está en bajo	BIL	2E	2	4
Salto si línea de interrupción está en alto	BIH	2F	2	4
Salto a subrutina	BSR	AD	2	8

TABLA 5.- INSTRUCCIONES DE MANIPULACION DE BITS.

FUNCION	MNEMOTECNICO	Modos de direccionamiento					
		Bit Set/Clear			Tets de Bit y Brinco		
		CODIGO OP.	# BYTES	# CICLOS	CODIGO OP.	# BYTES	# CICLOS
Brinco si bit n es 1	BRSET n (n = 0...7)	-	-	-	2 * n	3	10
Brinco si bit n es 0	BRCLR n (n = 0...7)	-	-	-	01 + 2 * n	3	10
Poner bit n a 1	BSET n (n = 0...7)	10 + 2 * n	2	7	-	-	-
Poner bit n a 0	BCLR n (n = 0...7)	11 + 2 * n	2	7	-	-	-

TABLA 6.- INSTRUCCIONES DE CONTROL.

FUNCION	MNEMOTECNICO	Inherente		
		CODIGO OP.	# BYTES	# CICLOS
Transferencia de A a X	TAX	97	1	2
Transferencia de X a A	TXA	9F	1	2
Poner Bit de Acarreo	SEC	99	1	2
Limpiar Bit de Acarreo	CLC	98	1	2
Poner Bit de Interrupción Enmascarable	SEI	9B	1	2
Limpiar Bit de Interrupción Enmascarable	CLI	9A	1	2
Interrupción por Sofware	SWI	83	1	11
Retorno desde Subrutina	RTS	81	1	6
Retorno desde Interrupción	RTI	80	1	9
Reinicialización del Apuntador de Pila	RSP	9C	1	2
No-Operación	NOP	9D	1	2

Símbolos del código de condiciones:

H	Bit de acarreo intermedio
I	Bit de interrupción enmascarable
N	Negativo (Bit del signo)
Z	Cero
C	Acarreo/Prestamo
A	Se establece en 1.En otro a 0.
•	No afectado
?	Carga del CC a partir de la pila
1	Set
0	Clear

TABLA 7.- LISTA DE INSTRUCCIONES

Mnemotécnico	Modos de direccionamiento										Códigos de condición				
	Inherente	Inmediato	Directo	Extendido	Relativo	Indexado (No Offset)	Indexado (8 bits)	Indexado (16 bits)	Bit Set/ Clear	Bit test y Brinco	H	I	N	Z	C
ADC		X	X	X		X	X	X			A	•	A	A	A
ADD		X	X	X		X	X	X			A	•	A	A	A
AND		X	X	X		X	X	X			•	•	A	A	•
ASL	X		X			X	X				•	•	A	A	A
ASR	X		X			X	X				•	•	A	A	A
BCC					X						•	•	•	•	•
BCLR									X		•	•	•	•	•
BCS					X						•	•	•	•	•
BEQ					X						•	•	•	•	•
BHCC					X						•	•	•	•	•
BHCS					X						•	•	•	•	•
BHI					X						•	•	•	•	•
BHS					X						•	•	•	•	•
BIH					X						•	•	•	•	•
BIL					X						•	•	•	•	•
BIT		X	X	X		X	X	X			•	•	A	A	•
BLO					X						•	•	•	•	•
BLS					X						•	•	•	•	•
BMC					X						•	•	•	•	•
BMI					X						•	•	•	•	•
BMS					X						•	•	•	•	•
BNE					X						•	•	•	•	•
BPL					X						•	•	•	•	•

TABLA 7.- LISTA DE INSTRUCCIONES

Mnemotécnico	Modos de direccionamiento										Códigos de condición				
	Inherente	Inmediato	Directo	Extendido	Relativo	Indexado (No Offset)	Indexado (8 bits)	Indexado (16 bits)	Bit Set/ Clear	Bit test y Brinco	H	I	N	Z	C
BRA					X						•	•	•	•	•
BRN					X						•	•	•	•	•
BRCLR										X	•	•	•	•	A
BRSET										X	•	•	•	•	A
BSET									X		•	•	•	•	•
BSR					X						•	•	•	•	•
CLC	X										•	•	•	•	0
CLI	X										•	0	•	•	•
CLR	X		X			X	X				•	•	0	1	•
CMP		X	X	X		X	X	X			•	•	A	A	A
COM	X		X			X	X				•	•	A	A	1
CPX		X	X	X		X	X	X			•	•	A	A	A
DEC	X		X			X	X				•	•	A	A	•
EOR		X	X	X		X	X	X			•	•	A	A	•
INC	X		X			X	X				•	•	A	A	•
JMP			X	X		X	X	X			•	•	•	•	•
JSR			X	X		X	X	X			•	•	•	•	•
LDA		X	X	X		X	X	X			•	•	A	A	•
LDX		X	X	X		X	X	X			•	•	A	A	•
LSL	X		X			X	X				•	•	A	A	A
LSR	X		X			X	X				•	•	0	A	A
NEQ	X		X			X	X				•	•	A	A	A
NOP	X										•	•	•	•	•

TABLA 7.- LISTA DE INSTRUCCIONES

Mnemotécnico	Modos de direccionamiento										Códigos de condición				
	Inherente	Inmediato	Directo	Extendido	Relativo	Indexado (No Offset)	Indexado (8 bits)	Indexado (16 bits)	Bit Set/ Clear	Bit test y Brinco	H	I	N	Z	C
ORA		X	X	X		X	X	X			•	•	A	A	•
ROL	X		X			X	X				•	•	A	A	A
RSP	X										•	•	•	•	•
RTI	X										?	?	?	?	?
RTS	X										•	•	•	•	•
SBC		X	X	X		X	X	X			•	•	A	A	A
SEC	X										•	•	•	•	1
SEI	X										•	1	•	•	•
STA			X	X		X	X	X			•	•	A	A	•
STX			X	X		X	X	X			•	•	A	A	•
SUB		X	X	X		X	X	X			•	•	A	A	A
SWI	X										•	1	•	•	•
TAX	X										•	•	•	•	•
TST	X		X			X	X				•	•	A	A	•
TXA	X										•	•	•	•	•

1.4. TRANSDUCCION Y DIGITALIZACION DE TEMPERATURA.

El uso de equipo electrónico en procesos industriales, requiere que éstos cuenten con módulos de adquisición de datos confiables. En nuestro caso particular, dicho módulo lo forman 4 etapas posteriores: Transducción, Amplificación, División y Conversión A/D.

Cada una de ellas, modifica la variable de voltaje-temperatura de acuerdo a las características que necesita la etapa subsecuente. A continuación se detalla cada etapa.

1.4.1. CIRCUITO TRANSDUCTOR.

Debido a las características del proyecto, fue necesario armar un circuito transductor de temperatura. Un transductor es un dispositivo capáz de convertir la energía de una forma a otra, conservando las variaciones de amplitud, frecuencia o fase de la onda que se convierte, tal como un micrófono, un altavoz, una fotocelda, un sensor de presión, etc.

En electrónica, las únicas variables mensurables son las diferencias de potencial o los flujos de corriente; por lo tanto se requería un dispositivo que convirtiera los incrementos de temperatura en incrementos de voltaje, preferentemente. Para ello nos apoyamos en la teoría termoeléctrica, principalmente en el efecto Seebeck. Este efecto, descubierto por el físico alemán Thomas J. Seebeck en 1821, muestra como al calentar un par de trozos de metales diferente puestos en contacto, producen una corriente eléctrica. Véase la **Fig. 1.27.**

Seebeck, pudo determinar que la diferencia de potencial originado por la corriente eléctrica era directamente proporcional al incremento de temperatura, quedando expresado por:

$$Vc - Vf = S \cdot (Tc - Tf) \qquad \ldots\ (1)$$

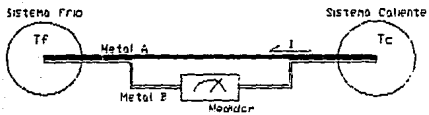


Fig. 1.27.- Experimento de Seebeck.

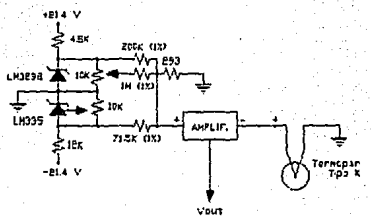


Fig. 1.28.- Circuito de Transducción.

Donde:

S representa el coeficiente de Seebeck dado en [V/°C] y que es distinto para cada unión de metales.

Vc es el voltaje del lado caliente y Vf el voltaje del lado frío.

Tc la temperatura del lado caliente y Tf la temperatura del lado frío.

La **Fig. 1.28** nos muestra un circuito para transducción de temperatura usando un termopar tipo K, él cual aprovecha el efecto Seebeck.

Un termopar, es un dispositivo similar al usado por Seebeck en su experimento, a excepción de que los metales están soldados solamente por un extremo, quedando las otras terminales abiertas. Por ello, se puede completar la cupla termoeléctrica con otro circuito y no necesariamente con otro termopar.

Tal como puede apreciarse en dicha figura, la cupla es completada con un circuito sensor de temperatura ajustable, recomendado por NSC (National Semiconductor Corporation). El termopar es introducido en el sistema a monitorear, mientras el circuito permanece ajustado a temperatura ambiente. Y aun cuando las variaciones de temperatura ambientales afectan al circuito, no son de importancia.

La respuesta obtenida con este transductor es de 40.8 ± 2 µV/°C de voltaje diferencial entre las terminales positiva y negativa que se observan en la **Fig. 1.28.**

Sin embargo, dicha respuesta es demasiado pequeña para ser medida con precisión, por lo cual se debe amplificar esta variable de voltaje-temperatura de salida a un nivel apropiado. Considerando que será aplicada a un circuito convertidor A/D, debemos conocer los requerimientos de entrada para este dispositivo.

El convertidor A/D es un dispositivo que cuantifica digitalmente la amplitud de una señal analógica. El usado en el proyecto es un ADC0808, cuya resolución es de 8 bits y la señal máxima permitida de entrada es 5.5 Volts. Además, el nivel mínimo de voltaje para un LSB (Least Significant Bit [bit menos

significativo de una expresión binaria]) es de 21 ± 1.5 mV. Por lo cual, si deseamos evitar cálculos innecesarios del µControlador debemos elevar la salida del transductor a 21 mV/°C, dándonos una relación Temperatura - Voltaje Digitalizado de 1:1.

1.4.2. CIRCUITO AMPLIFICADOR.

Observando las características del circuito mostrado en la **Fig. 1.28**, tenemos que el voltaje de salida se obtiene de una diferencia de los voltajes entre el circuito sensor y el termopar. Por tal razón el amplificador a usar debe ser uno de tipo de Diferencial.

Este tipo de amplificador tiene como principales características las siguientes:

1) Alta resistencia de entrada,
2) Ganancia Ajustable y
3) su elevado Rechazo en Modo Común.

Generalmente a este circuito lo constituyen 3 amplificadores, como puede verse en la **Fig. 1.29**. Es más comúnmente llamado Amplificador de Instrumentación.

Analizando este circuito, se tienen:

a) Dos OpAms (A1 y A2) en configuración de amplificador diferencial básico para cargas flotantes y
b) Un OpAm (A3) en configuración de amplificador no inversor.

Los cálculos de ganancias arrojan para los OpAms A1 y A2:

$$Av1 = 1 + (2/a);\ \text{donde } a = (aR/R) = (10K/47K) = 0.212766$$

$$\Rightarrow \quad Av1 = 1 + (2/0.212766) = \mathbf{10.40}$$

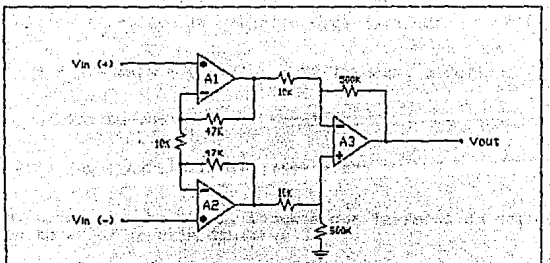


Fig. 1.29.- Amplificador de Instrumentación.

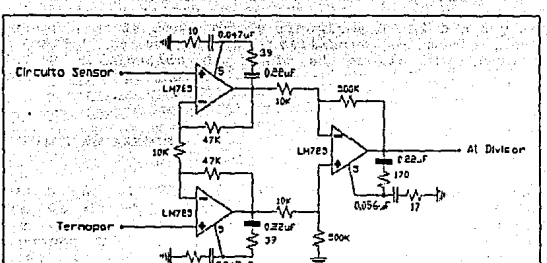


Fig. 1.30.- Circuito de Amplificación.

para A3, se tiene que la ganancia está dada por:

$$Av2 = m = mR/R = 500K/10K = 50 \text{ (ganacia diferencial)}$$

Dando por último una ganancia total de:

$$At = (Av1)(Av2) = (10.40)(50) = 520$$

Si partimos de este valor, tenemos que la señal de salida de este circuito es de aproximádamente:

$$Vo2 = 21.216 \text{ mV/°C}$$

Lo cual coincide perfectamente con el paso estándar de 1 LSB del circuito Convertidor (=21mV ± 1.5) ADC0808. Con esto relacionamos directamente la etapa de amplificación con la etapa de Digitalización, sin efectuar ningún cálculo posterior.

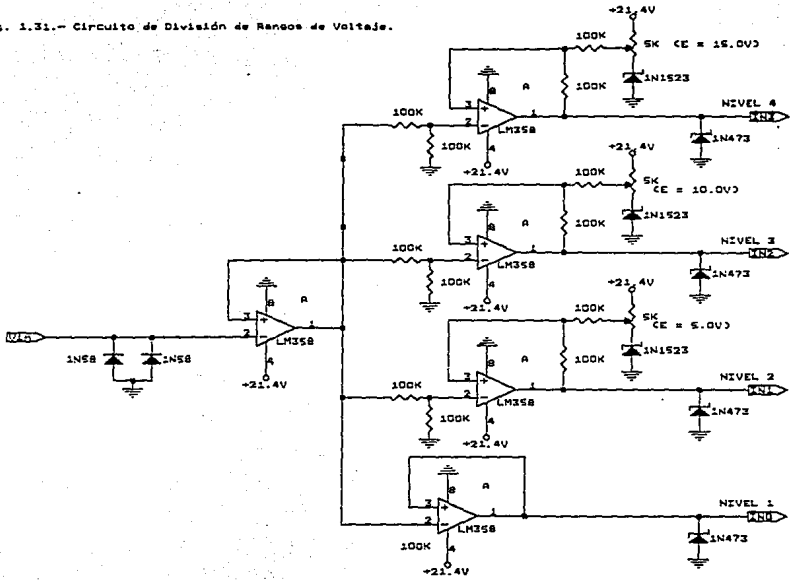
El circuito implementado hace uso de esta configuración tal y como se aprecia en la **Fig. 1.30.** En ella, se puede ver que la única diferencia es la conexión de circuitos RC al pin 5 y salida del OpAm LM725, los cuales son utilizados solamente para dar estabilidad y eliminación de ruido al propio integrado y no afectan la señal de salida.

1.4.3. DIVISION EN RANGOS.

Debido a que el voltaje máximo permitido como entrada al Convertidor es de +5.5 Volts y el mínimo es de -1.5 Volts, tenemos la necesidad de recortar la entrada en niveles de voltaje, puesto que de la etapa amplificadora el voltaje de salida puede alcanzar ±20 Volts, lo cual podría dañar nuestro convertidor. Véase la **Fig. 1.31.**

Inicialmente debemos eliminar los voltajes negativos, debido a que no tienen una razón lógica o física de existir en el diseño, puesto que los voltajes siempre se esperan positivos. Sin embargo, ya que la alimentación de la etapa amplificadora requiere también de un voltaje negativo y no se puede estar totalmente exento de una falla que provocará una saturación

Fig. 1.31.- Circuito de División de Rangos de Voltaje.



negativa que debe eliminarse. Con el fin de eliminar esta probabilidad de error hacemos uso de un par de diodos de Germanio, cuya caída de voltaje es de 0.3 Volts y una potencia de 5 Watts por diodo, conectados en inversa a la salida de la etapa amplificadora. De esta manera, al estar presente un voltaje negativo en dicha salida, encontrará un camino más corto a tierra por los diodos que por los circuitos posteriores, con lo cual es eliminada, quedando como remanente la caída propia de los diodos (-0.30 Volts).

Enseguida, debemos partir en niveles de voltaje la variable de voltaje-temperatura ingresada. Para ello, se dividió en múltiplos de 5 Volts la entrada, quedando de la siguiente manera:

1) En el primer nivel, se tienen voltajes que fluctúan entre -0.30 y 5.00 Volts, un recorrido nominal entre 0.0 y 5.0 Volts y una salida máxima fijada de 5.2 Volts.

2) En el segundo nivel, se tienen voltajes que fluctúan entre 5.00 y 10.00 Volts, un recorrido nominal entre 0.0 y 5.0 Volts y una salida máxima fijada de 5.2 Volts.

3) En el tercer nivel, se tienen voltajes que fluctúan entre 10.00 y 15.00 Volts, un recorrido nominal entre 0.0 y 5.0 Volts y una salida máxima fijada de 5.2 Volts.

4) En el cuarto y último nivel, se tienen voltajes que fluctúan entre los 15.00 y 20.00 Volts, con un recorrido nominal entre 0.0 y 5.0 Volts y una salida máxima fijada en los 5.2 Volts.

La única manera de efectuar lo anterior es usando otro circuito llamado circuito comparador de voltaje. Este circuito resta al voltaje ingresado en la entrada positiva (+) el voltaje ingresado en la entrada negativa (-) del amplificador con una ganancia unitaria. Si en la entrada negativa ponemos un voltaje fijo, se restará al voltaje de la entrada positiva un valor constante, que puede fijarse de acuerdo al nivel que se desea. Por otra parte, para evitar que el voltaje de salida superé los 5.5 Volts se fija mediante un diodo Zener de 5.2 Volts y 1 Watt de potencia.

Al efectuar un análisis del circuito comparador (Fig. 1.32) se tiene:

Mediante un análisis de corriente:

$$Vr1 = I1 \cdot R1 = (Vin/(Ri+R1)) \cdot R1 = (Vin \cdot R1)/(Ri+R1) \quad \ldots (1)$$

luego por tierra virtual, observamos que:

$$Ir = (Vr1-Er)/R2$$

y aplicando la ecuación (1), queda como

$$Ir = (((Vin \cdot R1/(Ri+R1)) - Er)/R2) \quad \ldots (2)$$

Ahora, la caída de voltaje en Rf es:

$$VRf = Ir \cdot Rf$$

entonces, usando la ec. (2), llegamos a:

$$VRf = (((Vin \cdot R1/(Ri+R1)) - Er)/R2) \cdot Rf \quad \ldots (3)$$

Finalmente, el voltaje de salida del amplificador está dado por:

$$Vo = VRf + VR1$$

por lo que, usando las expresiones (1) y (3), se tiene:

$$Vo = (((Vin \cdot R1/(Ri+R1)) - Er)/R2) \cdot Rf + (Vin \cdot R1)/(Ri+R1)$$

desarrollando la expresión anterior, se obtiene:

$$Vo = (Vin \cdot R1 \cdot (R2+Rf) - Er \cdot Rf \cdot (Ri+R1))/(R2 \cdot (Ri+R1)) \quad \ldots (4)$$

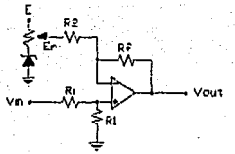
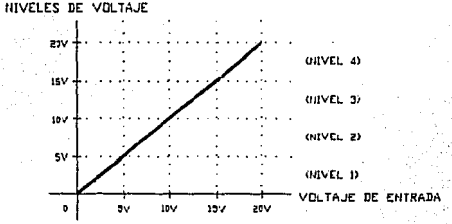


Fig. 1.32.- Cto. Comparador.

Fig. 1.33.- Gráfica Voltaje - Niveles.



Pero debido a que todas las resistencias son iguales, entonces:

$$Req = R1\cdot(R2+Rf) = Rf\cdot(Ri+R1) = R2\cdot(Ri+R1) \quad \ldots (5)$$

Por lo tanto, aplicando la ec. (5) en la ec. (4), tenemos:

$$Vo = (Req\cdot(Vin - Er))/Req = Vin - Er \quad \ldots (7)$$

Esta última expresión, nos indica que el voltaje de salida es la diferencia del voltaje de entrada menos el voltaje de referencia. Lo cual cimienta el concepto dado al respecto.

En la **Fig. 1.31**, se muestra el circuito divisor completo. En el observamos que para cada nivel de voltaje se usa un circuito comparador y además un seguidor de voltaje que impulsa la señal en los cuatro comparadores. La salida particular de cada circuito es tomada como una entrada individual al Convertidor.

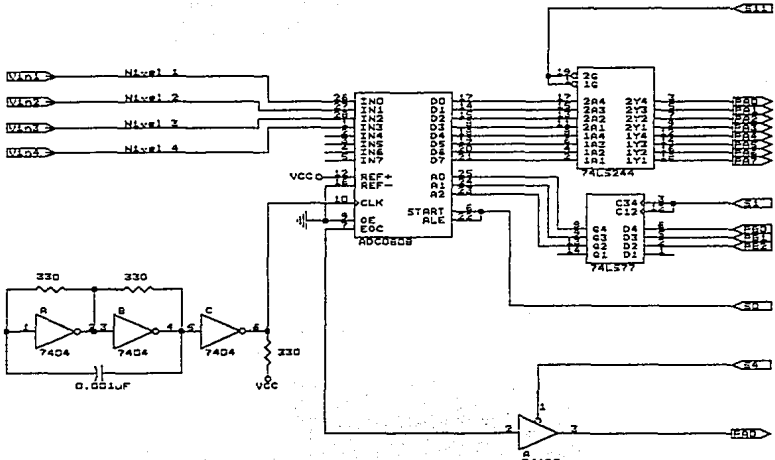
Finalmente, la **Fig. 1.33** esquematiza el concepto de voltaje contra niveles que manejamos en el divisor.

1.4.4. CONVERSION ANALOGICA - DIGITAL.

Una vez que la variable de voltaje-temperatura a sido amplificada, adecuada y dividida, la etapa que nos hace falta es la conversión de ésta en un dato digital. Tal etapa consiste en cuantificar la magnitud de un voltaje en una representación binaria única, la cual se lleva a cabo usando un circuito convertidor Analógico/Digital. Véase la **Fig. 1.34.**

En el mercado existen muchos dispositivos de este tipo, sin embargo, el que presenta más ventajas al mejor precio es el ADC0808 de National. Sus principales características son 8 entradas multiplexadas, una resolución de 8 bits (compatible con el bus de entrada del microcontrolador), ajustes internos automáticos de Cero y Escala Completa, un Latch de salida Tri-Estado, requerimientos de una sola fuente de alimentación (5.0 Volts regulados), un bajo gasto de potencia (alrededor de 15 mW) y una velocidad de conversión de 100 μseg para una frecuencia de entrada de 640 KHz.

Fig. 1.34.- Circuito del Convertidor Analógico Digital.



Ahora bien, todo el diseño anterior al ADC, fue elaborado para acoplar correctamente la variable de voltaje-temperatura del Termopar a este circuito. Puesto que sus pasos de incremento binario son de 20 mV ± 1.5 mV, obligada en la salida de la etapa amplificadora.

Continuando con el circuito, se puede apreciar en la **Fig. 1.34** que a las entradas del multiplexor se conectan las salidas del grupo divisor en forma ascendente y las cuales son controlados por el propio microcontrolador, mediante un latch al bus de Salida (Puerto B) y habilitado por el DECO de control a su entrada ALE.

El inicio de conversión (START) del ADC, es enviado por el microcontrolador en el momento que se requiere un monitoreo de temperatura y éste espera la señal de fin de conversión (EOC) proveniente del ADC, para permitirle tomar el bus de entrada e ingresar el dato de lectura obtenido, vía un lacth externo (74LS244). No se usa el lacth del ADC por razones de seguridad del bus.

Finalmente, resta comentar que un circuito de adquisición de datos como éste, resulta sumamente barato y confiable. Pero presenta algunas desventajas como el no poder ajustar automáticamente el coeficiente Seebeck y no poder checar el funcionamiento correcto de las partes que lo componen a través del microcontrolador.

1.5. TECLADO Y PANTALLA DE VISUALIZACION DE DATOS.

Todo sistema de control automático puede ser tan sencillo o complicado como se quiera de acuerdo al proceso o aplicación al cual será destinado, sin embargo, el principio de funcionamiento no deja de ser el mismo: a partir de una o varias entradas, producir las salidas correspondientes que se necesiten en algún punto del proceso.

De aquí se desprende la necesidad de contar con algún medio o interfaz entre lo que es el sistema y el usuario. El sistema requiere para su funcionamiento, una serie de valores o parámetros iniciales que le indiquen el comportamiento o acción correspondiente a tomar. Así mismo, el usuario debe de contar con algún medio que le permita determinar en cualquier momento el estado del sistema, con el fin de llevar un registro del comportamiento del mismo o bien para tomar la decisión adecuada en situaciones críticas. El comportamiento del sistema puede determinarse a través de algún medio visual o audible que indique el buen o mal funcionamiento del mismo.

En el caso del Controlador Lineal Programable de Temperatura, el elemento de entrada de datos está constituido por un teclado matricial de 16 teclas, cuatro de las cuales asumen funciones especiales, siendo las restantes los valores numéricos del 0 al 9, incluyendo la tecla intro (E, Enter) y de borrado (C, Clear). El teclado, como se observa en la **Fig. 1.35,** está conectado a un codificador, el C.I. 74C922 para arreglo matricial, cuyo rango de alimentación, por ser CMOS, se encuentra entre +3 y +18 Volts. Con el fin de evitar niveles de voltaje no deseados, tomando en cuenta la calidad del teclado, la alimentación del codificador se ha fijado en +15 volts.

El C.I. 74C922 cuenta con un circuito interno antirrebotes, así como con la posibilidad de evitar teclas fantasma. Está provisto de una salida que indica la validez de un dato (DA = Data Available) con la cual se puede controlar a su vez la habilitación de la salida. Ya que la alimentación general del circuito es de +5V, a la salida del codificador se conectaron divisores de tensión para reducir la salida al nivel adecuado.

Fig. 1.35.- Circuito para conexión del teclado.

<C> Tecla Clear/Abort. Usada para borrar datos de entrada o abortar el flujo del programa.

<E> Tecla Enter. Usada para aceptar datos de entrada.

<F1> Control Proporcional con Pasos Automáticos Temporizados.

<F2> Control Proporcional.

<F3> Control ON-OFF.

<F4> Termómetro Temporizado.

La corriente de salida para cada uno de los pines del codificador es de aproximadamente 35 mA, con lo cual para obtener 5 volts de un punto a tierra resulta:

Para R1:

$$I = 35\ mA$$
$$R1 = V1/I = 5V/35\ mA = 142.857\ Ohms$$

y para R2:

$$R2 = V2/I = 10V/35mA = 285.7143\ Ohms$$

Por lo tanto, considerando valores comerciales para resistencias, R1 se puede fijar en 150 Ohms y R2 en 270 Ohms sin afectar en gran medida los voltajes requeridos de un punto a otro.

El valor de la tecla pulsada se hace llegar al microcontrolador a través del puerto A, controlando su paso mediante un buffer (74LS244) habilitado por programa.

El elemento de salida lo forman cuatro display's de cátodo común. Estos display's permiten la visualización de los datos introducidos vía teclado así como de los valores que se obtienen, de tiempo y temperatura, a lo largo del proceso dependiendo del tipo de control.

Se emplea un buffer (74LS670) como medio de almacenamiento intermedio en donde se coloca el dato a ser visualizado así como el número de display habilitado, tal como se muestra en la **Fig. 1.36**. Este circuito constituye una memoria de 4 palabras, en donde cada palabra está formada de 4 bits, con pines individuales para habilitación de lectura/escritura y para la dirección de la palabra a leer o escribir. El pin de habilitación para lectura siempre se encuentra habilitado y las palabras almacenadas son barridas constantemente por medio de un contador módulo 3 (MOD 3), hecho a base del 74LS90 el cual a su vez está conectado a un 74LS139, decodificador doble de 2X4, con salidas bajo activas, utilizado para determinar el número de display a habilitar. El contador de barrido emplea una fuente de reloj de 1KHz de frecuencia a base de un 555 en configuración astable.

Fig. 1.38.- Pantalla de visualización de datos.

El dato a ser visualizado (escrito), así como la dirección en la cual se almacenará, son controlados por medio de programa a través del puerto B.

Para la decodificación de datos provenientes del 74LS670 se emplea un decodificador impulsor de BCD a 7 segmentos, específicamente el 74LS248.

Como medio informativo auxiliar se emplea una barra de diez leds indicadores de estado con un código específico asignado a cada uno de ellos.

El dato a desplegar en la barra de leds se envía desde el puerto B y se almacena temporalmente en un buffer, en este caso el C.I. 74LS77, para ser decodificado por un C.I. 74LS42 con salidas bajo activas, activándose el led correspondiente. Véase la **Fig. 1.37** para mayor información.

Los valores asignados a cada led incluye las señales Listo, para indicar que el sistema se encuentra en espera de algún dato o función; Termómetro Temporizado, Control ON-OFF, Control Proporcional y Control Proporcional por Pasos Automáticos Temporizados, para indicar la función que el sistema está llevando a cabo; Temperatura Final, Incremento de Temperatura y Tiempo, para indicar el tipo de dato a introducir como parámetro; Dato Erróneo, para indicar algún error en el dato introducido; y finalmente Mal Funcionamiento, empleado para indicar una falla general del sistema.

Cada uno de los leds empleados para informar sobre el estado del sistema se encuentra conectado a tierra a través de una resistencia de 330 Ohms.

Fig. 1.37.- Leds indicadores de estado.

1.6. ETAPA DE POTENCIA

La etapa de potencia constituye un elemento fundamental en todo sistema, es el elemento que proporciona la energía suficiente para que un motor funcione, un relevador se abra o se cierre, un horno empiece a calentar y en fin, toda una planta se eche a andar, a través del acoplamiento de cada una de sus partes, que de por sí, ya constituyen un proceso. Véase la **Fig. 1.38.**

Para comenzar, es necesario hablar un poco más sobre la aplicación, motivo por el cual fue elaborado el Controlador Lineal Programable de Temperatura.

El Instituto Mexicano del Petróleo (IMP), institución para la cual fue elaborado el controlador, requería de algún dispositivo electrónico capaz de poder ejercer el control sobre la temperatura de un horno. En este horno se lleva a cabo el calentamiento de diversas sustancias de las cuales se necesita conocer distintos parámetros a distintas temperaturas variables. En ciertos casos una temperatura específica, requiere mantenerse por un lapso determinado de tiempo.

Aunque la aplicación del controlador puede extenderse no sólo a hornos, éste debe ser capaz de ejercer el control directo sobre la alimentación de energía del horno en cuestión, el cual se alimenta con un voltaje de 127 Vca, permitiendo el calentamiento de la sustancia tratada o bien, el enfriamiento. A su vez, el controlador necesita conocer las diferentes temperaturas a las que se encuentra el horno en diferentes lapsos de tiempo con el fin de tomar una acción determinada sobre el horno. Dichas temperaturas son proporcionadas por la etapa de Transducción y Digitalización de Temperatura explicada en el apartado anterior.

El poder controlar la etapa de potencia es el fin fundamental del controlador, el cual determinará automáticamente los tiempos de encendido y apagado de dicha etapa, así como el tiempo que pertenecerá encendida o apagada, de acuerdo al tipo de control que se esté llevando a cabo. Esto es, cuando el controlador funcione como termómetro temporizado la etapa de potencia permanecerá encendida en todo momento, permitiendo con esto, el calentamiento del horno pero sin llevar a cabo ningún control sobre el mismo, sino simplemente indicando la temperatura

Fig. 1.38.- Etapa de potencia.

a la que se encuentra cada determinado tiempo, especificado a través del teclado.

En control ON\OFF, la etapa de potencia se enciende siempre y cuando la temperatura a alcanzar (SET POINT) esté por arriba de la temperatura actual del horno, en caso contrario permanece apagado con el fin de lograr el enfriamiento. Una vez que la temperatura del horno es igual a la temperatura establecida (SET POINT), la etapa de potencia deja de actuar.

Para control proporcional sucede un proceso similar al control de tipo ON/OFF, sólo que una vez alcanzada la temperatura, esta se mantiene mediante una conmutación a encendido/apagado. El horno se enciende cuando la temperatura cae 5°C por debajo del SET POINT o bien se apaga cuando la temperatura sube 5°C o más arriba del mismo, por lo cual la etapa de potencia está constantemente conmutando.

Finalmente, en el Control Proporcional por Pasos Automáticos Temporizados (PAT), se combinan el control ON/OFF y el control proporcional, a partir de la temperatura actual, se incrementa a la siguiente temperatura encendiendo la etapa de potencia y después, una vez alcanzada, se empieza a mantener por un tiempo establecido, encendiendo y apagando la etapa de potencia como se indicó anteriormente. El proceso continua de la misma forma con la siguiente temperatura hasta alcanzar la temperatura final.

La etapa de potencia se activa al enviar un "1" lógico por el bit más bajo (Bo) del puerto B del MC68705P3 y habilitar el buffer correspondiente (74LS77) cuya salida se encuentra conectada a un optoacoplador MOC3030.

Cuando el fotodiodo que constituye el MOC3030 se excita, el Triac se satura y cierra el circuito que impulsa la compuerta de un segundo triac (SC257) el cual cierra el circuito que permite la alimentación al horno. Debido a los transitorios de la línea de ca, se cuenta con un circuito RC amortiguador, con el fin de evitar un cierre en falso así como el daño del triac y del optoacoplador. En la **Fig. 1.39** puede verse el esquema de los optoacopladores tipo MOC3030. El detector está conectado en ellos de modo que operan como interruptores bidireccionales (TRIAC) con detección de paso por cero. Una aplicación típica sería, como en nuestro caso, controlar un TRIAC de potencia externo conectado en

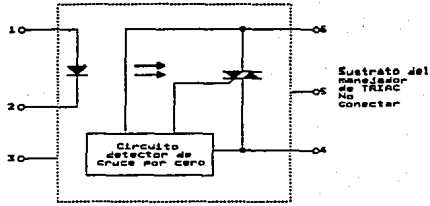


Fig. 1.39.- Esquema del optoacoplador MOC3030.

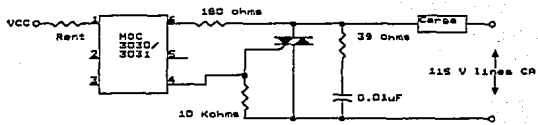


Fig. 1.40.- Uso de un optoacoplador para conmutar un triac de potencia externo.

serie con una carga de ca, a partir de una señal lógica de manejo, como se ve en la **Fig. 1.40.** El resistor de 39 Ohms en serie con el capacitor de 0.01 uF forma una red de filtro destinada a reducir la dV/dt sobre el TRIAC de potencia, para evitar en lo posible disparos en falso como se mencionaba anteriormente.

1.7. ALARMA.

La alarma es un dispositivo que genera señales audibles con el fin de llamar la atención de el o los usuarios en situaciones que requieran de la intervención humana.

Por tal razón, en este diseño se procuró agregar un circuito de Alarma audible, con dos tipos de sonido, uno para aviso de fallas y el otro como una interfaz amigable de acceso de datos o fin normal de proceso.

Observando la **Fig. 1.41**, tenemos que el circuito de alarma esta formado por un monoestable y dos multiestables (Maestro - Esclavo). El monoestable tiene como función arrancar los multiestables por un tiempo corto, a fin de generar solamente dos sonidos (BEEP's). Mientras que los multiestables efectúan el efecto de intermitencia de sonido (BEEP).

Efectuando un análisis del circuito, se tiene para el monostable:

$$Talta\_m = 1.1\ (680K \cdot 2.2\mu F) = 1.65 \text{ segundos.}$$

Para los dos multiestables, observamos que el esclavo tiene una frecuencia de oscilación de:

$$fesclavo = 1.44/(100K + 2(56K)\ 0.1\ \mu F) = 0.6793 \text{ Hz,}$$

mientras que el maestro tiene un tiempo de alta igual a:

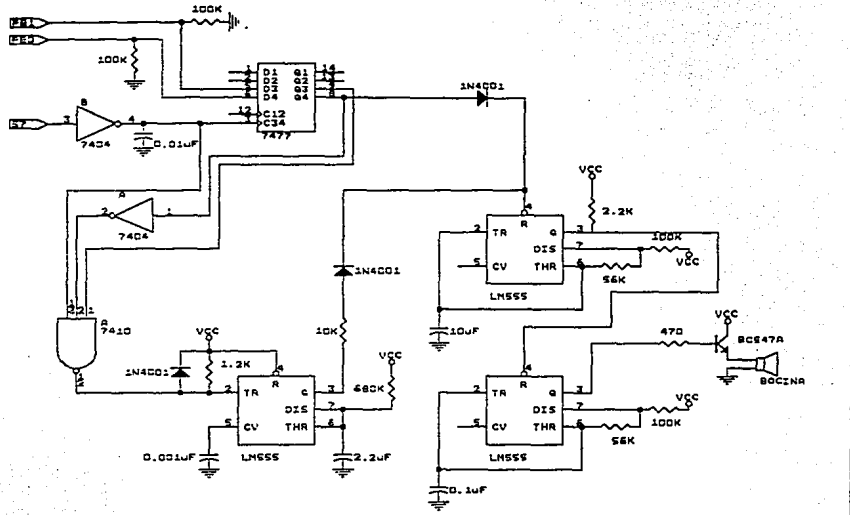
$$Talta = 0.695\ (100K + 56K)\ 10\ \mu F = 1.1 \text{ segundos}$$

lo cual fue elegido para escuchar una intermitencia poco molesta.

En la **Fig. 1.42** se observa el circuito implementado para ser manejado por el µControlador.

Fig. 1.41.- Conexión de Multiestables.

Fig. 1.42.- Circuito para Conexión de la Alarma.



El control de los dos posibles sonidos se efectúa mediante el manejo de la entrada de reestablecimiento del maestro (pin 4). Así, una señal fija produce una intermitencia constante, mientras que una señal variable genera trenes de sonido (grupos de BEEP's).

Se eligieron del puerto de salida dos líneas que manejarán la alarma. La línea b0 se encarga de arrancar el monoestable para generar sólo dos BEEP's y la línea b1 se encarga de mantener activa la intermitencia hasta que se oprima RESET.

CAPITULO 2
SOFTWARE

CAPITULO 2. SOFTWARE.

2.1. CONCEPTOS BASICOS

En un microcontrolador, al igual que en una computadora, la unidad más elemental de información es el dígito binario, conocido como bit, el cual sólo puede tomar dos posibles valores, "1" ó "0". Sin embargo, el empleo de un bit como una entidad aislada no proporciona la información suficiente para el microcontrolador, por ello se hace uso de un grupo de bits conocido con el nombre de palabra. El tamaño de palabra varía ampliamente de un microcontrolador a otro y está ampliamente relacionado con el tamaño de palabra de la unidad de memoria.

El tamaño de palabra generalmente se expresa en bytes, cuyo valor es igual a ocho bits. Así, un microcontrolador con un tamaño de palabra de ocho bits se dice que tiene un tamaño de palabra de 1 byte.

El valor de una palabra se puede interpretar de varias formas dependiendo de la arquitectura del microcontrolador así como del programador mismo. Los valores que una palabra puede tomar pueden ser algún dato numérico binario, algún dato codificado o bien, alguna instrucción.

2.1.1. SISTEMAS NUMERICOS

Los principales sistemas numéricos se describen en los párrafos siguientes sin profundizar en la definición de los mismos, tomando en cuenta que el estudio de éstos no forman parte de los fines para los cuales fue realizado el presente texto.

2.1.1.1. Sistema numérico binario.

El sistema numérico binario es un sistema posicional donde cada dígito binario tiene un cierto valor basado en su posición relativa con respecto al bit menos significativo (LSB).

En el sistema binario sólo existen dos posibles valores digitales, el 0 y el 1. Cada dígito tiene un valor propio

expresado como potencia de 2. El bit más significativo (MSB) es aquél más a la izquierda y el bit menos significativo (LSB) es aquel que esta más a la derecha.

2.1.1.2. Sistema numérico octal

El sistema numérico octal tiene una base de ocho, lo cual significa que tiene ocho posibles valores: 0, 1, 2, 3, 4, 5, 6 y 7. El valor de cada dígito en una cantidad octal se puede expresar como una potencia de ocho.

La ventaja principal de este tipo de sistema radica en la facilidad con que se puede realizar la conversión entre números binarios y octales. La conversión de octal a binario se lleva a cabo convirtiendo cada dígito octal en su equivalente binario de 3 bits. La conversión de binario a octal es simplemente la operación inversa, los bits del número binario se conjuntan en grupos de tres comenzando desde el LSB. Posteriormente cada grupo se convierte en su equivalente octal.

El sistema numérico octal se presenta como una forma alternativa, más compacta, de representar una cantidad binaria haciendo más fácil su lectura.

2.1.1.3. Sistema numérico hexadecimal

El sistema hexadecimal emplea la base 16, lo cual significa que cuenta con 16 posibles valores. Utiliza los dígitos del 0 al 9 más las letras A, B, C, D, E y F como los 16 símbolos digitales. Los dígitos hexadecimales del A al F son equivalentes a los valores decimales del 10 al 15.

Al igual que el sistema numérico octal, el sistema hexadecimal se usa principalmente como un método abreviado para representación de números binarios. Para llevar a cabo la conversión de un número hexadecimal a binario, cada dígito hexadecimal se convierte en su equivalente binario de 4 bits.

La conversión de binario a hexadecimal es exactamente la operación contraria a la anterior. El número binario se agrupa en conjuntos de cuatro bits y cada grupo se convierte en su dígito hexadecimal correspondiente.

2.1.1.4. Código BCD

Una computadora sólo puede trabajar con cantidades numéricas de tipo binario, sin embargo, cualquier número decimal se puede representar por medio de un número binario equivalente. Cuando un número decimal se representa con su número binario equivalente, a esto se le denomina codificación binaria directa.

Al código que se produce cuando cada dígito de un número decimal se representa con su equivalente binario se le denomina "decimal codificado en binario" (BCD). Para poder representar un dígito decimal (del 0 al 9) en binario sólo bastan cuatro bits, por lo tanto se necesitan cuatro bits por cada dígito existente en un número decimal.

El código BCD no hace uso de los números binarios 1010, 1011, 1100, 1101, 1110 y 11111, lo cual quiere decir que sólo se emplean 10 de los 16 posibles grupos de código binario de 4 bits.

Es importante entender que el BCD no es otro sistema numérico como el binario, el octal, el decimal y el hexadecimal. Es, de hecho, el sistema decimal con cada dígito codificado en su equivalente binario. El código BCD requiere más bits que el binario directo para representar números decimales de más de un dígito.

El código BCD es muy útil en procesos en los cuales la información decimal se aplica como entrada o bien se exhibe como salida.

2.1.2. INTRUCCIONES DE PROGRAMA

Las instrucciones de un microcontrolador constituyen palabras de instrucciones las cuales contienen información necesaria para que el microcontrolador ejecute sus diversas operaciones y varían ampliamente de un fabricante a otro.

La palabra de instrucción se forma de dos partes: el código de operación y la dirección del operando sobre el cual se llevará a cabo la operación.

El número de bits que forman el código de operación indica el total de operaciones disponibles para un microcontrolador, de esta forma un código de operación de cuatro bits implica un total de 16 instrucciones disponibles por parte del usuario. Cada instrucción tiene un código de operación específico que el microcontrolador debe interpretar.

El número de bits para la parte de la dirección del operando establece el número de direcciones que pueden referenciarse para la obtención del mismo.

Las palabras de instrucciones se denominan así mismo, instrucciones en lenguaje máquina pues se representan por unos y ceros, siendo el único lenguaje reconocido por los microcontroladores.

Cada instrucción implementada en un microcontrolador emplea una serie de símbolos, denominados "Mnemotécnicos" que pueden traducirse en una instrucción codificada en binario. Esta traducción es hecha por un programa especial denominado ensamblador.

Un ensamblador es un programa que acepta un programa de lenguaje simbólico y produce su equivalente binario de lenguaje máquina. El programa simbólico de entrada se denomina programa fuente y el programa binario resultante se denomina programa objeto.

En el momento de corrida, la unidad de control del microcontrolador busca y trae el código de operación y la dirección del operando de la memoria y realiza la operación indicada mediante la decodificación del código de operación.

2.2. CONCEPTO GLOBAL DEL PROGRAMA.

La circuitería del controlador es el primer paso, pero solamente representa la mitad del proyecto. La segunda parte esta constituida por la programación de el µControlador, la cual incluye rútinas de Entrada/Salida de información al usuario, Adquisición de Temperatura, Control de Proceso, Temporización y Detección de Errores.

La idea fundamental de la programación es aprovechar todos los recursos provistos al controlador, y a su vez, la circuitería se diseño para optimizar la programación. Es decir, que el controlador no fue concebido individualmente, sino que contempla ambos factores mencionados; aún cuando aquí lo separamos para ganar claridad.

Con el fin de efectuar la visualización global, es necesario retomar el diagrama a bloques del controlador, **Fig. 2.1a**. Podemos observar que los puertos del MCU están dedicados a una tarea específica. Así, el puerto A es el bus de Entrada del MCU, por el cual ingresan datos el convertidor A/D y el teclado. El puerto B es el bus de Salida, por el cual el MCU envia información al usuario a traves de los display de estado, los displays numéricos y la alarma; además acciona la etapa de potencia y los muestreos de temperatura. Finalmente el puerto C está dedicado a la habilitación de los módulos externos, orquestando la sincronización de todo el controlador.

De esta manera, el MCU observa el proceso como Entrada - Cálculos - Salida, que también es nuestro punto de vista. Dentro de la Entrada se tiene la captura de parámetros necesarios para el proceso, como son: Temperatura Final (Set Point), Variaciones de Temperatura y Tiempo. También se incluye como entrada la medición de Temperatura.

Dentro de los Cálculos se tiene la determinación del control a ejecutar, el punto de alcance con el Set Point y la temporización.

Por último, la Salida contempla la visualización en los Displays y Leds, Alarma, Etapa de Potencia y habilitación de cada módulo.

Fig. 2.1a. - Controlador de Temperatura.

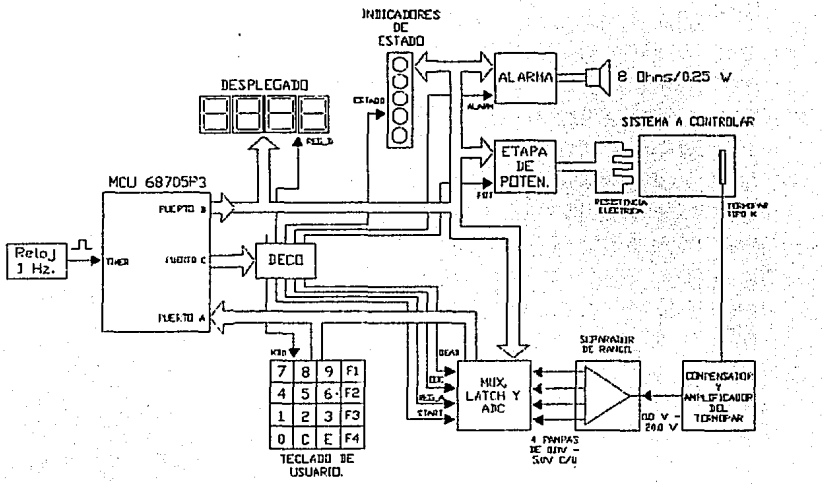
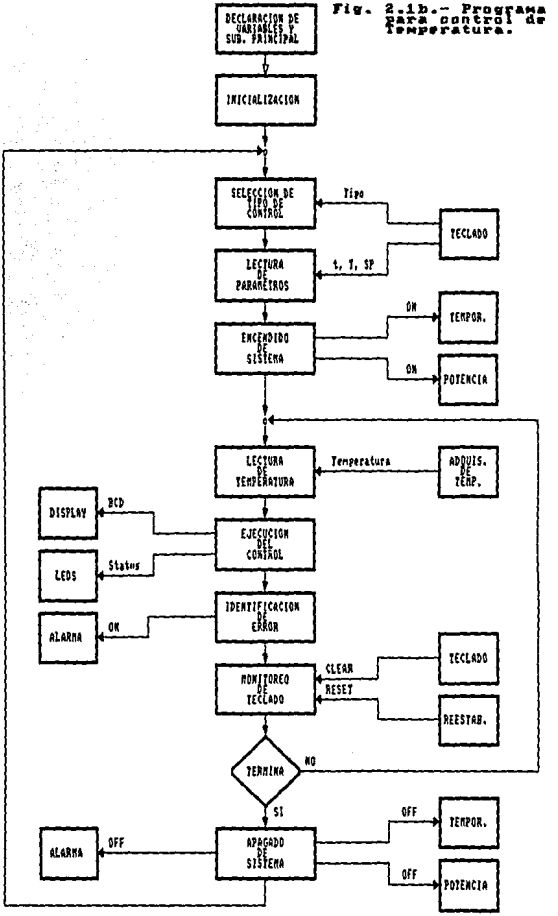


Fig. 2.1b.- Programa para control de temperatura.



En la **Fig. 2.1b** se muestra una carta de flujo global del programa, donde puede observarse que está constituido por los siguientes procesos:

a) DECLARACION DE VARIABLES Y CONSTANTES.
Se declaran las direcciones que tendrán las variables y constantes usados por el programa.

b) INICIALIZACION.
Asigna valor a las constantes y blanquea la RAM desde 10H hasta 6FH para su uso posterior.

c) SELECCION DEL TIPO DE CONTROL.
En este punto el controlador espera la elección del usuario, una vez dada determina que parámetros necesita leer.

d) LECTURA DE PARÁMETROS.
Los parámetros a leer dependen del tipo de control elegido, asi, el Termométro sólo requiere el intervalo de tiempo, el control ON/OFF y el Proporcional sólo requieren la temperatura final (Set Point) y el PAT (Proporcional Automático Temporizado) requiere de la temperatura final, el intervalo de tiempo y de un incremento de temperatura.

e) ENCENDIDO DEL SISTEMA.
Efectúa el encendido de la etapa de potencia y del temporizador.

f) LECTURA DE TEMPERATURA.
Efectúa la lectura de temperatura, con todos los pasos que esto implica.

g) EJECUCION DEL CONTROL.
Lleva a cabo las tareas correspondientes del control que el usuario halla elegido, los cuales son: Termométro Temporizado, Contro ON/OFF, Control Proporcional o Control Proporcional por Pasos Automáticos Temporizados.
Para el Termometro, contabiliza el tiempo y lo compara contra el intervalo capturado para efectos de monitoreo de temperatura. Además despliega la temperatura y el tiempo.
En el control ON/OFF, sólo monitorea la temperatura y la despliega.
El control Proporcional, monitorea la temperatura, la despliega y la mantiene constante.
Finalmente, el control PAT monitorea la temperatura, despliega el dato y la mantiene constante por lapsos de tiempo especificados. Una vez terminado el tiempo incrementa automáticamente el Set Point en un valor determinado por el usuario.

h) IDENTIFICACION DE ERROR.
Aquí, el programa determina si la temperatura ha rebasado los límites establecidos por error del circuito de adquisicón o ganancia excesiva del sistema controlado. En cuyo caso acciona la Alarma y fuerza la terminación del flujo normal, apagando la etapa de potencia.

i) MONITOREO DEL TECLADO.
En el caso de un flujo normal o anormal, se requiere por diversas su interrumpción. Por lo cual, el programa monitorea constantemente la señal de CLEAR (C) del Teclado o RESET de botón, a fín de interrumpir el proceso normal de control.

j) APAGADO DEL SISTEMA.
Efectúa el apagado del Temporizador, Etapa de Potencia y Alarma para reiniciar un nuevo flujo.

Adicionalmente, pueden identificarse las subrútinas que llevan a cabo cada proceso. Estas subrútinas son:

1) PRINCIPAL.
La cual se ubica dentro de los procesos de Inicialización, Selección de Tipo de Control y Apagado de Sistema.

2) DECIDE.
Ubicada en la Lectura de Parámetros.

3) TERMOMETRO, ON_OFF, PROPORC y PROGRAM.
Las cuatro corresponden a los tipos de control implementados y están contempladas dentro de los procesos de Encendido de Sistema Ejecución de Control.

4) LECTER.
Esta subrútina se encarga de efectuar la lectura de temperatura y checar el desborde de la misma. Por lo tanto, se ubica dentro de los procesos de Lectura de Temperatura e Identificación de Error.

5) LEE_KBD.
Su tarea es checar el suceso de CLR en el teclado, por lo cual está dentro del procesos de Monitoreo de Teclado.

Cada subrútina anterior no efectúa por si sóla el proceso completo, se apoya en otras para llevarlo a cabo. En las siguientes secciones se detalla cada una.

2.3. MNEMOTECNICOS

El MC68705P3 cuenta con un juego muy completo de instrucciones o mnemotécnicos que facilitan la elaboración de programas por parte del usuario. El programa puede ser capturado en cualquier procesador de textos que genere código ASCII para posteriormente, traducirlo a sus correspondientes códigos en hexadecimal mediante el empleo del ensamblador de motorola. Este programa calcula automáticamente las direcciones de los operandos y saltos a subrutinas por lo que el programador no tiene que preocuparse de hacerlo.

Las instrucciones de programación para el MC68705P3 pueden clasificarse en diversas categorías:

1.- Instrucciones sobre registros y memoria.
2.- Instrucciones de lectura, escritura y modificación de registros y memoria.
3.- Instrucciones de salto.
4.- Instrucciones de manipulación de bits y
5.- Instrucciones de control.

Los registros empleados por estas instrucciones incluyen el acumulador (A), el registro índice (X), el registro de datos del temporizador (TDR), el registro de control del temporizador (TCR), y los que corresponden a los puertos de entrada/salida, aunque el operando también puede ser algún dato almacenado en memoria.

En la categoría de instrucciones sobre registros y memoria, se cuenta con instrucciones de transferencia, aritméticas (suma y resta), lógicas, de comparación, de almacenamiento y brinco (incondicional y a subrutina).

Las instrucciones de lectura/escritura/modificación incluyen las instrucciones de desplazamiento y rotación, incremento y decremento así como de complementación a uno y dos.

Las instrucciones de brinco ejecutan un chequeo sobre algún bit del registro de condiciones y de acuerdo al estado del mismo, realizan un salto a la dirección de memoria especificada o bien, continuan con la siguiente instrucción en secuencia.

Dentro de las instrucciones de manipulación de bits, se cuenta con sólo cuatro instrucciones, en primer lugar BRSET y BRCLR, las cuales checan el estado del bit especificado de algún registro u operando en memoria, y de acuerdo al estado del mismo realizan o no el salto hacia la dirección especificada; en segundo lugar, se encuentran las instrucciones BSET y BCLR, cuyo propósito es cambiar el bit especificado de un determinado registro a "1" (BSET) o bien a "0" (BCLR) con algún propósito definido.

Las instrucciones de control se basan en el modo de direccionamiento inherente en el cual toda la información necesaria para ejecutar la instrucción se encuentra implícita en el código de operación, e incluyen la transferencia entre el acumulador y el registro índice, en una u otra dirección, el retorno de subrutina o interrupción, la interrupción por software y el restablecimiento del apuntador de pila (SP).

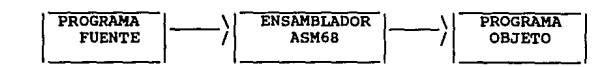
En las tablas 2 a la 6 se muestra un listado de todos los mnemotécnicos del MC68705P3 según su clasificación, indicando los tipos de direccionamiento empleados, así como los números de ciclos de máquina y bytes requeridos.

En la tabla 7 se proporciona un listado en orden alfabético de todas las instrucciones que conforman el MC68705P3 junto con el registro de condiciones con el fin de establecer como son afectadas las banderas de estado.

2.3.1. EL ENSAMBLADOR DEL MICROCONTROLADOR MC68705

El programa ensamblador del microcontrolador MC68705 en sus diferentes versiones P3, R3 y U3, lo constituye el archivo denominado ASM68.EXE. Este programa es el encargado de transformar un programa objeto, construido mediante mnemónicos, a el código de máquina del microcontrolador.

El siguiente diagrama esquematiza el proceso:



El ensamblador ASM68 arranca a partir de un archivo objeto, el cual ha sido creado utilizando algún procesador de textos convencional tipo ASCII.

Una vez terminado el ensamblaje del programa, el ensamblador se encarga de generar dos tipos de archivos:

1.- <nombre_archivo>.LST
Contiene el código máquina correspondiente a cada instrucción dentro del programa.

2.- <nombre_archivo>.HEX
Contiene el programa en código hexadecimal para la programación de la EPROM. Para copiar este archivo a la EPROM se puede emplear, por ejemplo, el programador de memorias PROMPROG.

El ensamblaje del programa fuente puede ser realizado de dos formas distintas. La primera forma se lleva a cabo tecleando las instrucciones correspondientes desde el sistema operativo como se muestra a continuación:

METODO 1:

A:\> asm68 <RETURN>

Después de oprimir RETORNO, el ensamblador pregunta por el nombre del archivo que contiene el programa fuente en el lenguaje ensamblador del microcontrolador, extensión .ACM.

Enter [.acm] file name: MIPROG.ACM <RETURN>

Ensegida pregunta por un nombre de archivo en el cual será almacenado el código máquina correspondiente a cada una de las instrucciones del programa. La extensión por defecto para este archivo es .LST .

Enter [.lst] file name: MIPROG.LST <RETURN>

Finalmente el programa pide un nombre de archivo, extensión .HEX por defecto, en el cual almacenará la información del programa en código hexadecimal necesaria para la grabación de la memoria EPROM.

Enter [.hex] file name: MIPROG.HEX <RETURN>

Este último archivo puede ser empleado para realizar la simulación del funcionamiento del programa, utilizando el simulador para el microcontrolador MC68705, conocido como TSIM05.

Para realizar la simulación del programa basta con teclear, desde el sistema operativo, la siguiente sentencia:

A> TSIM05 <nombre_archivo>.HEX

El simulador se encargará de ejecutar cada una de las instrucciones del programa, mostrando en pantalla el resultado de de las mismas mediante el desplegado del valor de los registros que integran el microcontrolador MC68705.

El segundo método es más rápido y sencillo que el anterior pero requiere que el archivo fuente contenga la extención .ACM. Por tal motivo al introducir el nombre del archivo no es necesario especificar la extención .ACM ya que el programa la toma así por omisión.

METODO 2:

A:\> asm68 MIPROG

Una vez que se introducen cada uno de los nombres de archivo en los cuales se almacenarán los distintos tipos de información o bien que se da el nombre del archivo fuente, en el caso del segundo método, el ensamblador empieza su tarea de transformar el código fuente al código máquina correspondiente. Si se detecta algún error en la sintaxis de alguna instrucción, el ensamblador detiene el proceso y envía el numero de errores encontrados así como la línea en la cual se detectaron.

Si el ensamblador no encuentra errores en el programa, el ensamblaje se realiza en forma exitosa generándose los archivos correspondientes.

2.3.2. METODOLOGIA DE PROGRAMACION

El código para el ensamblador ASM68 presenta, para la mayoría de las instrucciones, la siguiente sintaxis:

<INSTRUCCION> [<OPERANDO O DIRECCION DEL OPERANDO>]

Para cada una de estas instrucciones hay que tomar en cuenta las siguientes consideraciones:

a).- Por cada línea de código se acepta solamente una instrucción y alternativamente el comentario correspondiente.

Ejemplo:

```
STA  $001      * Este es un comentario
```

Los comentarios no son tomados por el ensamblador y solo sirven como fines informativos y de organización para el programador. Para realizar un comentario, basta con anteponer al texto el símbolo asterisco (*).

b).- Un renglón puede contener únicamente un comentario sin existir previamente una instrucción.

Ejemplo:

```
LDA  #$034
* Comentario en línea sin instrucción
STA  $000
```

c).- Las instrucciones deben de estar separadas del operando o de la dirección del operando por lo menos de un espacio en blanco. No se acepta que el operando o dirección se escriba en un renglón separado al de la instrucción:

Ejemplos:

```
LDA #b01001001          * Instrucción válida

STA     $001            * Instrucción válida

CMP#$01                 * Instrucción no válida
                        * No existe separación

AND                     * Instrucción no valida
#b01101011              * Ocupa dos reglones
```

d).- Las instrucciones pueden teclearse en mayúsculas o minúsculas.

Ejemplo:

```
LDA #$00F
sta $67
Cmp $0F
```

2.3.3. SINTAXIS DE LAS INSTRUCCIONES

2.3.3.1. Comentarios

Los comentarios se definen a través de un asterisco (*).

Ejemplo:

```
LDA  $000 * Carga en el acumulador la información
          * del puerto cero.
```

2.3.3.2. Dirección de programa

La instrucción ORG indica la dirección en memoria en la cual se localiza el inicio del programa. Esta instrucción es útil para definir los vectores de interrupción, área de datos, área de programa, etc.

Ejemplo:

```
ORG $080

LDA $00F
CMP #$03
AND $01F
STA $002
```

2.3.3.3. Tipos de constantes

Existen tres tipos válidos:

a).- DECIMAL

Ejemplo: 1234

b).- HEXADECIMAL

Ejemplo: $0AB2 (H,h: símbolos opcionales)

c).- BINARIO

Ejemplo: %01010100 [B,b: símbolos opcionales]

Si el valor equivalente de la constante es superior a los dos bytes, es decir mayor que $FFFF, entonces el ensamblador sólo toma los 2 bytes menos significativos de la constante.

2.3.3.4. Definición de expresiones

Una expresión es una operación de adición o substracción. El ensamblador únicamente soporta estas dos operaciones.

Ejemplos:

```
LDA  buffer + 2
STA  buffer - 1
LDA  buffer + ptr - 2
```

El ensamblador no permite el uso de paréntesis al escribir alguna expresión por lo que no es apropiado el empleo de los mismos.

2.3.3.5. Definición de constantes

La definición de constantes se lleva a cabo mediante el empleo de la directiva EQU la cual permite asociar una etiqueta con una constante. Este hecho le da al programa cierta flexibilidad en el momento de realizar cambios, permitiendo utilizar la etiqueta en sustitución de la constante y haciendo más legible su lectura para el usuario.

Ejemplo:

```
PUERTOA    EQU   $000
PUERTOB    EQU   $001
ENTRADA    EQU   $00
SALIDA     EQU   $FF
MASK       EQU   b01010011
DDRA       EQU   $003
DDRB       EQU   $004

LDA   #ENTRADA
STA   DDRA          * programa el DDRA
LDA   #SALIDA
STA   DDRB          * programa el DDRB
LDA   PUERTOA
AND   MASK
STA   PUERTOB
```

Una etiqueta puede estar definida hasta con 9 caracteres. En este caso el ensamblador sí distigue entre una letra en mayúscula y una en minúscula.

Ejemplo:

Para el ensamblador cada una de las siguientes etiquetas son distintas:

```
PUERTOA
puertoa
Puertoa
pUERTOA
```

Se recomienda utilizar etiquetas con caracteres en mayúsculas para la definición de etiquetas que son constantes y minúsculas para las etiquetas que se utilizan en las instrucciones de salto de control.

2.3.3.6. Etiquetas de flujo de control

Estas etiquetas se utilizan con la finalidad de identificar una dirección de memoria hacia la cual se transfiere el flujo del programa después de cumplirse una determinada condición.

Ejemplo:

```
LOOP LDA  PUERTOA
     CMP  #$01
     BEQ  LOOP
```

La mayoría de los textos para ensambladores de la serie 68000 indican como calcular el desplazamiento en bytes (offset) al realizar un salto. Sin embargo, con la definición de etiquetas, el cálculo lo realiza automáticamente el programa ensamblador, evitando este tipo de cálculos al usuario.

Como restricción en la definición de etiquetas, se recomienda no utilizar etiquetas que comiencen con las letras B, b, H y h, empleadas para especificar números binarios y hexadecimales, lo cual ocasiona un error al utilizar la etiqueta como operando de alguna instrucción.

Ejemplo:

```
BEGIN   LDA #b01010011
        JMP BEGIN          * Se genera un error

_BEGIN  LDA $23FE
        JMP _BEGIN         * Instrucción válida
```

2.3.3.7. Apartado y definición de memoria

El apartado de memoria se puede realizar mediante el empleo de dos tipos de instrucciones: FCB y RMB.

La primera instrucción se encarga de apartar un byte de memoria, inicializando el byte a la constante especificada enseguida de la instrucción.

La segunda instrucción aparta dos bytes de memoria y se inicializa de igual forma que la instrucción anterior.

Ejemplo:

```
        ORG  $100

TABLE   RMB  $5678
        RMB  $3421
        RMB  $AD37
        RMB  $34

VARS    FCB  45
        FCB  56
        FCB  67
```

Los bytes se pueden referenciar a través de una etiqueta como se vió en los ejemplos anteriores.

También se pueden apartar bytes e inicializarlos con el valor correspondiente a una etiqueta.

Ejemplo:

```
ORG $7FE

RMB TABLE
```

En este caso, el byte $7FE se ha inicializado con 01 y el byte $7FF con 00 ya que la etiqueta TABLE tiene asignado un OFFSET de $100 el cual es un número de dos bytes.

Finalmente, se pueden inicializar varios bytes mediante la definición de una cadena de caracteres entre comillas y números, ambos separados por comas.

Ejemplo:

```
        fcb  "cadena de caracteres", "otra cadena"
        fcb  "ejemplo dos", 34, $45, %10110001, "fin"
        rmb  "otro tipo de ejemplo", $3423, 3215
```

Cada byte apartado contendrá el número ASCII de los caracteres contenidos dentro de la cadena. En el caso de los números, se almacenarán los mismos en el byte correspondiente.

A continuación se muestra el código de un programa fuente para realizar la suma y resta de dos números. El fin de este programa no es otro más que mostrar al usuario la forma en que se estructura.

```
*********************************************************
* Comentarios:                                          *
* Programa en lenguaje ensamblador del MC68705P3        *
* que realiza una suma y resta entre dos números        *
* de un dígito.                                         *
*********************************************************

Declaración ->    PTOA EQU $000
de puertos        PTOB EQU $001
                  PTOC EQU $002
                  DDRA EQU $004
                  DDRB EQU $005
                  DDRC EQU $006

Declaración ->    DIGITO1 EQU $010
de constantes     DIGITO2 EQU $011
                  SUMADI  EQU $012
                  RESTADI EQU $013

                * PROGRAMA PRINCIPAL
Comienzo del ->
programa                  ORG $080
                  INICIO  CLR DDRA      * ESTABLECE TODOS LOS
```

```
                                    * PUERTOS DE
                    CLR DDRB        * ENTRADA
                    CLR DDRC
                    LDA #$05        * PRIMER DIGITO = 5
Cuerpo del          STA DIGITO1
programa            LDA #$04        * SEGUNDO DIGITO = 4
                    STA DIGITO2
                    JSR SUMA
                    JSR DIF
                    BRA INICIO
                SUMA LDA DIGITO1
                    ADD DIGITO2     * SUMA = DIGITO1+DIGITO2
                    STA SUMADI
                    RTS
                DIF  LDA DIGITO1
                    SUB DIGITO2     * RESTA= DIGITO1-DIGITO2
                    STA RESTADI
                    RTS

        * DEFINICION  DE  VECTORES

        ORG $7F8
        RMB INICIO    * VECTOR DE INTERRUPCION DEL TIMER
        RMB INICIO    * VECTOR DE INTERRUPCION EXTERNA
        RMB INICIO    * VECTOR DE INTERRUPCION POR PROGRAMA
        RMB INICIO    * VECTOR DE INTERRUPCION POR RESET
        ORG $784
        FCB  $00      * VALOR DEL MOR
```

Como vemos, el programa se conforma de cuatro partes fundamentales: declaración de puertos, declaración de constantes, cuerpo del programa y declaración de vectores.

En la declaración de constantes se definen una serie de localidades de memoria fijas referenciadas mediante un nombre específico, haciendo uso de la pseudo-operación EQU y el símbolo de dolar ($) que indica una dirección de memoria especificada por los dos bytes que siguen al símbolo.

Al inicio del programa la pseudo-operación ORG $nn, establece el valor inicial del contador de programa PC al valor dado por nn el cual marca la dirección de comienzo del programa.

Aunque el programa no hace uso de los puertos, su inicialización es importante con el fin de evitar problemas en el hardware por lo que se han establecido todos de entrada.

En el cuerpo del programa se hace uso de dos subrutinas referenciadas mediante una etiqueta la cual no es más que una cadena de caracteres no mayor a diez, cuyo caracter inicial no puede ser B,b,H o h.

Finalmente se tiene la declaración de vectores donde se establece la dirección de empiezo de cada una de las rutinas de interrupción de acuerdo al tipo de interrupción que se genere, en nuestro caso todas están referenciadas al comienzo del programa principal.

Mediante la directiva rmb se pueden reservar dos bytes para las direcciones de tales vectores.

El valor del MOR también se establece en la dirección $784 y se ha puesto a un valor de 00 para un modo de control del TCR por programa.

2.4. TERMOMETRO TEMPORIZADO.

El objetivo de tener en el controlador una función que indiqué la temperatura existente en tiempos determinados es basicamente la de un muestreador. Inicialmente se implementó para efectuar pruebas del equipo, pero a la postre se observó que ésta cumplía con un requerimiento propio del analísis de sustancias compuestas.

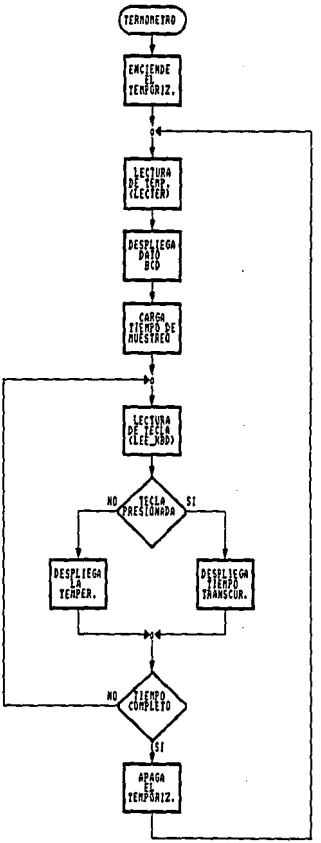
Al calentar una sustancia compuesta, la temperatura aumenta de una manera no lineal, dependiendo de los elementos disueltos o agregados en ella. Por lo tanto, cada sustancia presenta una curva de respuesta distinta, pero el Cromatógrafo no lo presenta, lo que hace necesario muestrar la temperatura ganada por la sustancia en tiempos fijos y justifica la existencia de esta función.

En la **Fig. 2.2**, se observa una carta de flujo para esta función y a continuación se lista el código que la compone.

```
434    TERMOMETRO    LDA #$01
435                  JSR ACTIVAE
436                  JSR RELOJ_OFF
437        RE_TERMO  JSR LECTER
438                  JSR DESP_BCD
439                  JSR CARGATPO
440         LOOPTER  JSR LEE_KBD
441                  LDA KBD
442                  STA PTOC
443                  BRSET 4,PTOA,DESP_TPO
444                  JSR DESP_BCD
445         VER_FIN  LDA TESP
446                  BEQ FIN_TPO
447                  BRA LOOPTER
448        DESP_TPO  JSR DESP_TIME
449                  BRA VER_FIN
450         FIN_TPO  JSR RELOJ_OFF
451                  BRA RE_TERMO
452                  RTS
```

Principia indicando en los leds de estado la función termometro (líneas 434 y 435) y apaga el temporizador, para iniciar de cero (línea 436). En seguida, lee la temperatura del sistema a muestrear, despliega el dato obtenido y carga el tiempo

Fig. 2.2.- Termómetro Temporizado.



a esperar para el siguiente muestreo en el temporizador (líneas 437 a 439).

Mientras se cumple el tiempo de espera, monitorea el teclado para determinar si el usuario desea terminar el proceso, presionando CLEAR (línea 440); o bien visualizar el tiempo transcurrido, presionando cualquier otra tecla (líneas 441 a 443 y 448). Si no se presiona ninguna tecla, se despliega la temperatura leida (línea 444).

En cuanto el tiempo de espera termina (líneas 445 y 446), se apaga el temporizador (línea 450) y se repite la lectura de temperatura y espera de tiempo (línea 451).

2.5. CONTROL ON/OFF

El control ON/OFF o control de dos posiciones es básico en el "Controlador Lineal Programable de Temperatura" ya que los restantes tipos de control, Proporcional y Proporcional por Pasos Automáticos Temporizados, requieren de este tipo de control como complemento indispensable para mantener una temperatura.

En el control ON/OFF se establece una temperatura (SET POINT) como único parámetro, la cual una vez alcanzada no se mantiene, es decir, este tipo de control, sólo tiene como propósito mantener el horno encendido o apagado, de acuerdo a si la temperatura actual del horno se encuentra abajo o arriba del SET POINT respectivamente, tratando de establecer una línea recta del tiempo contra temperatura, y una vez alcanzada, terminar el proceso, apagando el horno si es que éste se encuentra encendido.

Una vez establecida la forma de operación del control ON/OFF, ésta puede ser resumida de la siguiente forma:

Como primer punto, es necesario establecer la temperatura deseada o SET POINT, dato que será proporcionado a través del teclado y almacenado en localidades contiguas de memoria en formato BCD.

Una vez fijada la temperatura a alcanzar, si ésta es mayor a la temperatura actual, se activa la etapa de potencia, encendiendo el horno y manteniendolo así hasta alcanzar la temperatura deseada. Este proceso implica llevar a cabo un monitoreo constante sobre la temperatura, proceso que se lleva a cabo a través del convertidor analógico digital (ADC) para lo cual se han desarrollado las rutinas correspondientes.

Alcanzada la temperatura requerida, la etapa de potencia deja de actuar sobre el horno apagándolo.

El valor máximo tolerado por el controlador para temperaturas es de 1020° C, por lo que una vez alcanzada la temperatura fijada, el horno será apagado, activándose a su vez la alarma.

Fig. 2.3.- CONTROL ON - OFF

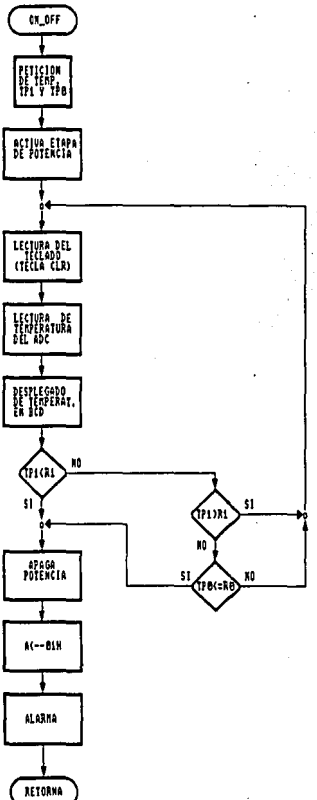
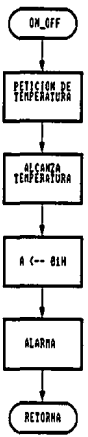


Fig. 2.4.- Control ON - OFF.



El diagrama de flujo correspondiente se muestra en la **Fig. 2.3.** Posteriormente serán desarrolladas las rutinas relacionadas a procesos específicos como lo es "Alcanzar Temperatura" y "Alarma".

Debido a que el proceso necesario para alcanzar una temperatura también será empleado por "Control Proporcional" y "Control Proporcional por Pasos Automáticos Temporizados", lo ubicaremos como una rutina aparte, de tal forma que el proceso CONTROL ON/OFF se reduce al llamado de las rutinas correspondientes como se muestra en la **Fig. 2.4.**

En las siguientes líneas se muestra el código equivalente en ensamblador del proceso ON-OFF:

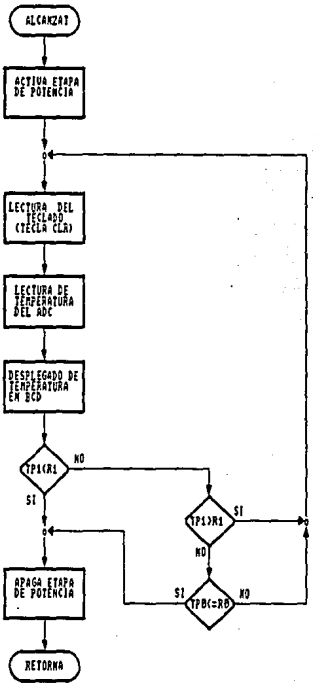
```
469     ON_OFF      LDA #$02
470                 JSR ACTIVAE
471                 JSR ALCANZAT
472                 LDA #$01
473                 JSR ALARMA
474                 RTS
```

La **Fig. 2.5** muestra el diagrama de flujo correspondiente al proceso "Alcanza Temperatura" (ALCANZAT).

Esta rutina será posteriormente modificada para aceptar los tipos de control restantes. Como podemos apreciar, se hace uso de algunas rutinas ya empleadas en "Termómetro Temporizado", como son "Lectura del Teclado" (LEE_KBD), "Lectura de temperatura" (LECTER) y "Despliega temperatura" (DESP_BCD), así como las correspondientes a la habilitación del horno (POTENCIA_ON y POTENCIA_OFF).

Recordando lo que hacen cada una de estas rutinas tenemos que "LEE_KBD" checa la ocurrencia de la tecla "clear" cuando se está llevando a cabo algún proceso; ésto indica que tal proceso se desea abortar, volviendo a la rutina "Menú" para la elección de un nuevo proceso, haciendo para ello un reestablecimiento del apuntador de pila (SP). "DESP_BCD", por su parte, es una rutina que se encarga de hacer el desplegado de las temperaturas medidas reportadas por el ADC y transformadas a un formato BCD. Finalmente el proceso "LECTER" se encarga de hacer las habilitaciones correspondientes en el ADC para llevar a cabo la conversión de la temperatura analógica a digital, apoyada por la rutina "CONVER" que realiza la conversión de hexadecimal a BCD.

Fig. 2.5.- Subrutina ALCANZAT.



"POTENCIA_ON" y "POTENCIA_OFF" son dos rutinas que se emplean para el encendido y apagado del horno respectivamente.

Para encender el horno simplemente necesitamos mandar un 01H por el puerto B y habilitar el buffer correspondiente al horno (palabra de control 08H) a través del puerto C. Para apagarlo, basta con mandar un 00H al puerto B y realizar las mismas habilitaciones. Por lo tanto estas dos rutinas pueden ser implementadas de la siguiente forma:

```
647    POTENCIA_ON  LDA #$01
648                 BRA SPOT
649    POTENCIA_OFF LDA #$00
650           SPOT  STA PTOB
651                 LDA POT
652                 STA PTOC
653                 LDA #$0F
654          RETPOT DECA
655                 BNE RETPOT
656                 LDA #$FF
657                 STA PTOC
658                 RTS
```

Finalmente nos encontramos con una nueva rutina, "ALARMA", la cual se requiere cuando la temperatura deseada ha sido alcanzada y solo produce una señal audible como medio indicador. Esta rutina será posteriormente implementada.

2.6. SUBRUTINA CONTROL PROPORCIONAL.

Teoricamente se define al control proporcional como un amplificador con ganancia ajustable, debido a que su función de transferencia es:

$$M(s) = Kp \cdot E(s),$$

donde M(s) es la señal de salida,
Kp es la sensibilidad proporcional o ganancia y
E(s) es la señal de error.

Bajo esta premisa, se tiene que el control proporcional acerca la señal de salida al punto deseado de una forma más suave que el control On/Off, modificando el ajuste de la ganancia Kp.

En nuestro diseño implicaba contar con una etapa de potencia ajustable, pero económicamente no se justificó. Por lo cual se optó por aprovechar un efecto térmico muy similar a la inercia mecánica, él cual consiste en el pequeño aumento de temperatura que sufre toda sustancia calentada, aun cuando se ha retirado la fuente de calor, debido a energía cinética residual.

Con lo anterior, tenemos que durante el calentamiento la ganancia Kp se mantiene constante y al acercarnos al punto final se apaga la etapa de potencia, dentro de un margen establecido para el horno de 5 °C, quedando la sustancia con la suficiente energía cinética en sus moléculas para alcanzar suavemente dicho punto por la modificación gradual de la ganancia Kp.

Además, se añadió a esta función la característica de mantener la temperatura constante en el punto final. Véase la **Fig. 2.6.**

```
476     PROPORC     LDA #$03
477                 JSR ACTIVAE
478                 JSR MANTENT
479                 RTS
480
481     MANTENT     JSR LEE_KBD
482                 LDA #$03
483                 CMP TIPO
484                 BNE ALOTRO
```

```
485                 LDA TESP
486                 BEQ SALMAN
487          ALOTRO JSR LECTER
488                 LDA KBD
489                 STA PTOC
490                 BRSET 4,PTOA,VER_TPO
491         VER_BCD JSR DESP_BCD
492                 BRA VER_SIG
493         VER_TPO LDA TIPO
494                 CMP #$03
495                 BNE VER_BCD
496                 JSR DESP_TIME
497         VER_SIG LDA TP1
498                 CMP R1
499                 BMI MANTENT
500                 BHI CALENTAR
501                 LDA TP0
502                 CMP R0
503                 BLS MANTENT
504                 SUB R0
505                 CMP #$05
506                 BHI CALENTAR
507                 BRA MANTENT
508          SALMAN JSR RELOJ_OFF
509                 RTS
510       CALENTAR JSR ALCANZAT
511                 BRA MANTENT
```

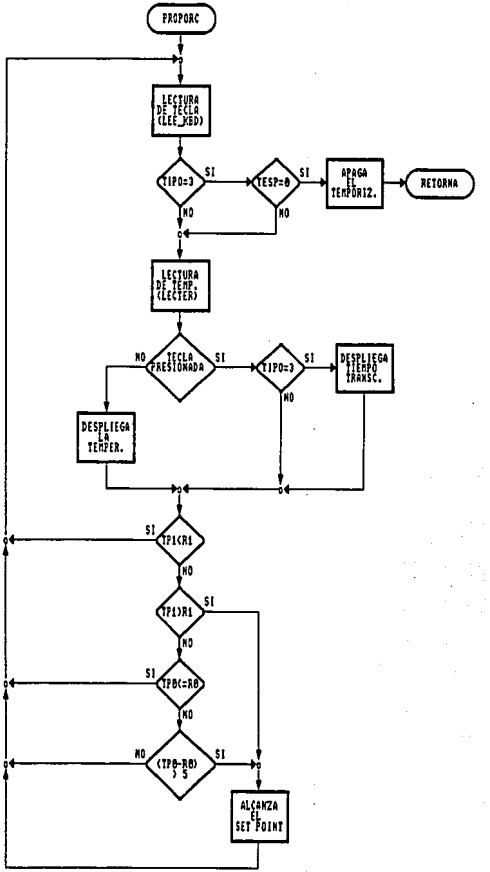
Al igual que los demás tipos de control, el Proporcional también indica en los leds de estado que se encuentra activo (líneas 476 y 477).

Seguidamente, ejecuta el código de la subrutina MANTENT que corresponde con el control proporcional, pero que fue separado para poderse usar en el control Programado (línea 478).

Así, lee la ocurrencia de CLEAR del teclado para terminar el proceso (línea 481), en caso de no existir, determina si el tipo de control que esta ejecutando es el Programado (líneas 482 a 484) para verificar que el tiempo de espera haya terminado (líneas 485 y 486), apagando el temporizador y regresar al punto de llamada (líneas 508 y 509).

En el caso de ambos controles, pero particularmente el proporcional, efectúa la lectura de temperatura (línea 487) y monitorea el teclado para checar si está oprimida alguna tecla

Fig. 2.6.- Subrutina Control Proporcional.



(líneas 488 a 490) a fin de desplegar el tiempo transcurrido (líneas 493 a 496) o de lo contrario desplegar la temperatura (líneas 491 y 492).

A continuación, compara la temperatura actual [almacenada en las variables R1 y R0] con la temperatura que debe alcanzar [SET POINT almacenado en TP1 y TP0] para determinar el momento de alcance del margen o superación del SET POINT (líneas 497 a 507). Si no alcanza todavía el margen, sigue encendida la etapa de potencia (línea 510 y 511).

2.7. CONTROL PROPORCIONAL POR PASOS AUTOMATICOS TEMPORIZADOS

Este tipo de control sigue el mismo criterio que el establecido para control proporcional, es decir, mantener una temperatura específica una vez que ésta es alcanzada. Sin embargo este tipo de control trabaja a partir de la temperatura inicial a la que se encuentra el horno. En base a un incremento de temperatura establecido, la temperatura se va incrementando y una vez alcanzada cada nueva temperatura, ésta es mantenida por un tiempo fijo, para después pasar hacia la siguiente temperatura y así sucesivamente. La **Fig. 2.7** muestra en forma gráfica el proceso.

Como observamos en la gráfica, el proceso antes descrito se lleva a cabo en forma continua hasta que se llega a una temperatura tope o se rebasa, por lo que el programa tendrá que checar la nueva temperatura incrementada antes de seguir con el procéso. Si la nueva temperatura está por debajo o es igual a la temperatura tope, el control se sigue ejecutando de lo contrario el proceso se suspende.

Por tanto este tipo de control necesita de tres parámetros iniciales:

1.- La temperatura final o temperatura tope.
2.- Un incremento de temperatura y
3.- Un intervalo de tiempo.

Para alcanzar cada temperatura establecida y para mantenerla durante el tiempo fijado, se hace uso de las rutinas ya implementadas, ALCANZAT y MANTENT a las que se les ha agregado las líneas necesarias para considerar el tiempo, lo cual involucra una rutina ya antes empleada en termómetro temporizado, la rutina TIMER, la cual a través del TCR y un reloj de tiempo real hace un decremento del tiempo hasta cero, siendo en este momento cuando se hace el incremento hacia la nueva temperatura.

Si la nueva temperatura a alcanzar rebasa la temperatura tope, se realiza una nueva asignación que establece dicha temperatura como la siguiente a alcanzar y mantenerse, haciendo uso de la alarma para indicar tal ocurrencia.

Fig. 2.7.- Grafica correspondiente a Control PAT.

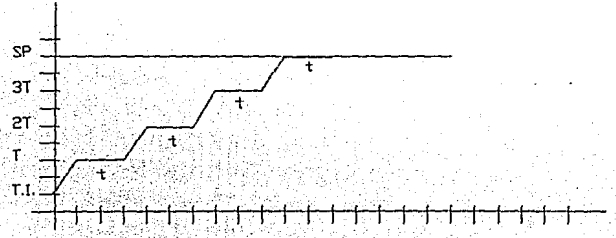
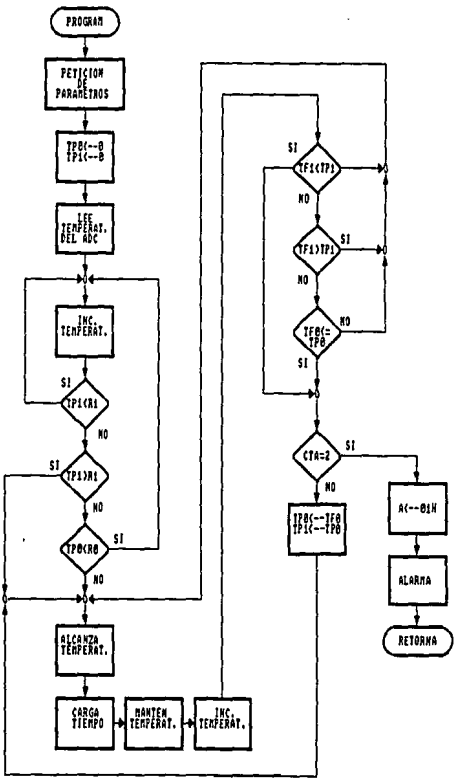


Fig. 2.8.- Subrutina Control PAT.



Las rutinas correspondientes para el incremento de temperatura serán implementadas posteriormente.

La **Fig. 2.8** muestra el diagrama de flujo correspondiente a control proporcional por pasos automáticos temporizados.

La rutina equivalente en ensamblador del MC68705P3 se muestra en las siguientes líneas.

```
513     PROGRAM       LDA #$04
514                   JSR ACTIVAE
515                   LDA #$00
516                   STA TP0
517                   STA TP1
518                   JSR LECTER
519           INCREMT JSR INCTEM
520                   LDA TP1
521                   CMP R1
522                   BMI INCREMT
523                   BHI LOOPPROG
524                   LDA TP0
525                   CMP R0
526                   BMI INCREMT
527         LOOPPROG  JSR ALCANZAT
528                   JSR CARGATPO
529                   JSR MANTENT
530                   JSR INCTEM
531                   LDA TF1
532                   CMP TP1
533                   BMI ASIGNAN
534                   BHI LOOPPROG
535                   LDA TF0
536                   CMP TP0
537                   BLS ASIGNAN
538                   BRA LOOPPROG
539          ASIGNAN  INC CTA
540                   LDA CTA
541                   CMP #$02
542                   BEQ PRIMER_AL
543          ASIGNAC  LDA TF0
544                   STA TP0
545                   LDA TF1
546                   STA TP1
547                   BRA LOOPPROG
548       PRIMER_AL   LDA #$01
549                   JSR ALARMA
550                   BRA ASIGNAC
```

2.8. SUBRUTINAS DE APOYO.

2.8.1. SUBRUTINA PRINCIPAL.

La subrutina principal es el primer código que el controlador ejecuta, por lo tanto en ella se lleva a cabo la inicialización de variables, apagado de módulos y la entrada en operación del sistema.

A continuación se muestra el código fuente correspondiente a esta subrutina (**Fig. 2.9**)

```
                    ORG $80
PRINCIPAL           CLR DDRA
                    LDX #$FF
                    STX DDRB
                    STX DDRC
                    LDX #$010
                    LDA #$00
LOOP1               STA X
                    INCX
                    CPX #$6F
                    BNE LOOP1
                    LDX #$030
OTRA                STA X
                    INCX
                    INCA
                    CMP #$0A
                    BNE OTRA
                    LDA #$04
                    STA CLEAR
MENU                JSR RELOJ_OFF
                    JSR POTENCIA_OFF
                    JSR CLDIS
                    LDA #$00
                    STA CTA
                    JSR ACTIVAE
                    LDX KBD
                    STX PTOC
TEC1                BRCLR 4,PTOA,TEC1
TEC2                BRSET 4,PTOA,TEC2
                    LDA PTOA
                    LDX #$0F
                    STX PTOC
                    AND #$0F
                    SUB #$0C
```

```
102                     STA TIPO
103                     JSR DECIDE
104                     JSR CONTROLES
105                     BRA MENU
```

En el observamos que se indica la dirección a partir de la cual se desea inicie el código en la EPROM (ORG $80, línea 68), se efectúa la definición del puerto A como entrada y los puertos B y C como salidas (líneas 69 a 72), se blanquea la RAM desde la dirección $010 a $06F con la intención de eliminar cualquier basura que pudiera afectar el correcto inicio del proceso (líneas 73 a 78) y se inicializan las constantes con los valores correspondiente (líneas 79 a 86).

Ya preparado el ambiente de trabajo se apaga el temporizador y la etapa de potencia como prevención (líneas 87 y 88). En seguida se muestran ceros en el display (línea 89), se enciende el led indicador de "MENU" (líneas 90 a 92, led 0) para indicar al usuario que el controlador se encuentra listo y esperando un tipo de control a ejecutar. Activa el teclado y espera se presione alguna de las teclas que corresponden al tipo de control deseado, F1 - F4 (líneas 93 a 97); una vez presionada, se carga al Acumulador el código de la tecla y se deshabilita el teclado (líneas 97 a 99). Debido a que el código de la tecla es más sencillo de manejar como un valor entre 0 y 9, se convierte antes de guardarse en la variable "tipo" (líneas 100 a 102); se lleva a cabo la lectura de parámetros de acuerdo al tipo de control requerido (línea 103) y se ejecuta dicho control (línea 104).

Finalmente, al regresar de la ejecución vuelve a iterar en espera de otro control a ejecutar (línea 105).

2.8.2. SUBRUTINA PARA LECTURA DE PARAMETROS (DECIDE).

La lectura de los parámetros requeridos por cada tipo de control lo efectúa la subrutina DECIDE. En ella se capturan el SET POINT (Punto de Temperatura por Alcanzar), el incremento de temperatura deseado (Control Programado) y el tiempo de muestreo/espera (Termómetro y Control Programado). Véase la **Fig. 2.10.**

Fig. 2.9.- Subrutina Principal.

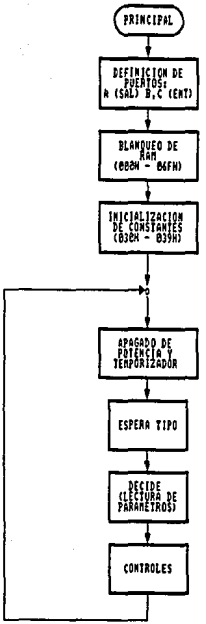
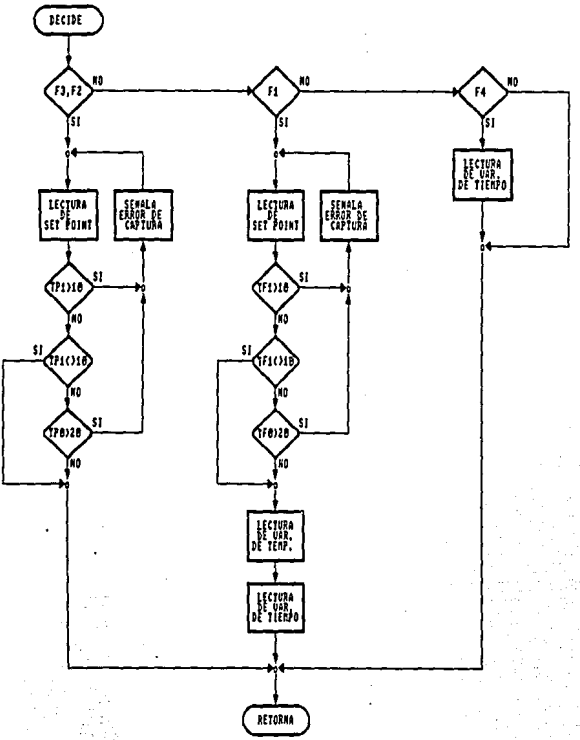


Fig. 2.18.- Subrutina para lectura de parámetros.



```
107     * SUBRUTINA DECIDE *
108
109     DECIDE        LDA TIPO
110                   CMP #$01
111                   BEQ DECI0
112                   CMP #$02
113                   BEQ DECI0
114                   CMP #$03
115                   BEQ DECI1
116                   CMP #$00
117                   BEQ NPJM2
118                   RSP
119                   JMP MENU
120           DECI0   LDA #$05
121                   JSR ACTIVAE
122                   JSR READ_KEY
123                   LDA $063
124                   STA TP1
125                   LDA $064
126                   STA TP0
127                   LDA TP1
128                   CMP #$10
129                   BHI PJMS
130                   BNE NPJMS
131                   LDA TP0
132                   CMP #$20
133                   BHI PJMS
134           NPJMS   RTS
135            PJMS   JSR SEQUIV
136                   JMP DECI0
137
138           DECI1   LDA #$05
139                   JSR ACTIVAE
140                   JSR READ_KEY
141                   LDA $063
142                   STA TF1
143                   LDA $064
144                   STA TF0
145                   LDA TF1
146                   CMP #$10
147                   BHI PJM1
148                   BNE NPJM1
149                   LDA TF0
150                   CMP #$20
151                   BHI PJM1
152                   JMP NPJM1
153            PJM1   JSR SEQUIV
154                   JMP DECI1
155           NPJM1   LDA #$06
156                   JSR ACTIVAE
157                   JSR READ_KEY
```

```
158                 LDA $041
159                 STA VAR3
160                 LDA $042
161                 STA VAR2
162                 LDA $043
163                 STA VAR1
164                 LDA $044
165                 STA VAR0
166                 LDA $63
167                 CMP #$03
168                 BHI PJM2
169                 BNE NPJM2
170                 LDA $64
171                 CMP #$00
172                 BHI PJM2
173                 JMP NPJM2
174           PJM2  JSR SEQUIV
175                 JMP NPJM1
176         NPJM2   LDA #$07
177                 JSR ACTIVAE
178                 JSR READ_KEY
179                 LDA $041
180                 STA VTIME3
181                 LDA $042
182                 STA VTIME2
183                 LDA $043
184                 STA VTIME1
185                 LDA $044
186                 STA VTIME0
187                 RTS
```

Inicialmente carga en el Acumulador el tipo de control que desea ejecutarse (línea 109), se identifica de cual se trata y se procede a capturar los parámetros que necesita (líneas 110 a 117) y en el caso que no sea ningún tipo válido se procede a regresar a la subrutina principal (líneas 118 y 119).

Para los controles ON_OFF (On/Off) y PROPORC (Proporcional) solamente se captura el SET POINT, checando que no sea mayor a 1020 °C (máximo del controlador) y si lo fuera, se le indica al usuario que es erróneo (líneas 120 a 136).

Para el control PROGRAM (Programado), se capturan el SET POINT, validando que no superé los 1020 °C, como en ON_OFF y PROPORC (líneas 138 a 154); el incremento de temperatura que desea darse a cada paso del controlador (líneas 155 a 175) y el tiempo que se requiere mantenga constante la temperatura en cada paso, una vez alcanzado (líneas 176 a 187).

Finalmente, el TERMOMETRO sólo necesita se le capturé el tiempo de muestreo de temperatura (líneas 176 a 187).

2.8.3. SUBRUTINA PARA ACTIVAR LEDS DE ESTADO (ACTIVAE).

Básicamente la función de esta subrutina es encender el led de estado correspondiente a cada control o condición del sistema. Los distintos estados están numerados de la siguiente manera:

Numero	Estado
0	Listo (MENU)
1	Termómetro Temp.
2	Control On/Off
3	Control Proporcional
4	Control PAT
5	Captura de SET POINT
6	Captura Incremento de Temp.
7	Captura Incremento de Tpo
8	Dato Erróneo
9	Mal Funcionamiento

El numero del estado que se desea encender es pasado a través del Acumulador. Véase la **Fig. 2.11.**

```
189     ACTIVAE       STA PTOB
190                   LDA ESTADO
191                   STA PTOC
192                   LDA #$0F
193         RETARDO1  DECA
194                   BNE RETARDO1
195                   LDA #$0F
196                   STA PTOC
197                   RTS
```

Como punto inicial pone el estado deseado en el puerto B (línea 189), en seguida activa el latch de los leds de estado para que tome el dato del puerto B (líneas 190 y 191). Debido a la alta velocidad del µControlador deben efectuarse ciclos de retardo para sincronizarlo con la circuitería externa (líneas 192 a 194).

Fig. 2.11.- Subrutina para activar leds de estado.

Finalmente, deshabilita el latch de los Leds, dejando encendido el led deseado (líneas 195 a 197).

2.8.4. SUBRUTINA PARA INDICACION DE ERROR Y PAUSA (SEQUIV).

Con esta subrutina se busca agregar a la captura de parámetros una indicación audible de error, acompañada de una espera de tecla para repetir la captura. Véase la Fig. 2.12.

```
199     SEQUIV      LDA #$01
200                 STA PTOB
201                 LDA ALARM
202                 STA PTOC
203                 LDA #$0F
204         REPEAT  DECA
205                 BNE REPEAT
206                 LDA #$09
207                 JSR ACTIVAE
208                 LDA KBD
209                 STA PTOC
210            SEQ1 BRCLR 4,PTOA,SEQ1
211            SEQ2 BRSET 4,PTOA,SEQ2
212                 LDA #$0F
213                 STA PTOC
214                 RTS
```

Primeramente, activa la alarma en modo temporal (sólo 2 beeps), poniendo en el puerto B un 01H (líneas 199 y 200) y habilitando el latch de la alarma (líneas 201 y 202). Enseguida efectúa una ciclo de retardo para que el latch alcance a tomar el dato (líneas 203 a 205) y comience a sonar la alarma.

Habilita el teclado para esperar que el usuario observe su error (líneas 208 a 211) y repetir la captura. Por último deshabilita el teclado (líneas 212 y 213).

2.8.5. SUBRUTINA PARA LECTURA DE DATOS (READ_KEY).

Como hemos estado mencionando, en cada tipo de control se requiere antes de hecharlo a funcionar, una serie de parámetros en base a los cuales se llevará a cabo el control. Estos datos deberán ser introducidos vía teclado y almacenados en memoria

Fig. 2.12.- Subrutina para indicación de error y pausa.

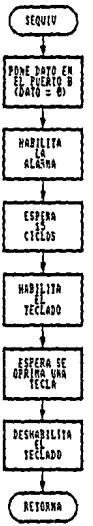
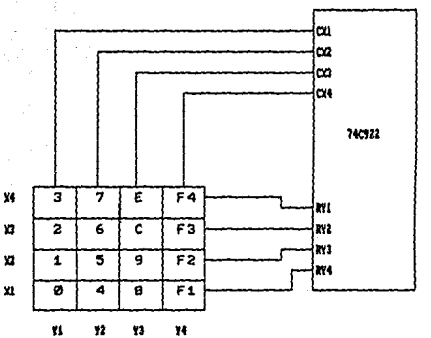
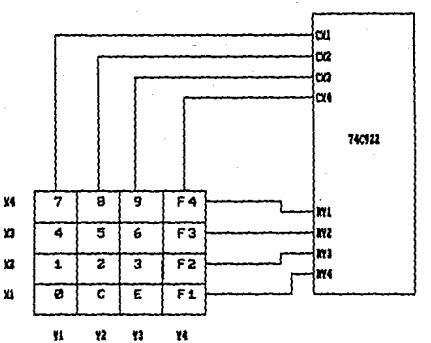


Fig. 2.13.- Conexión de teclados al 74C922.





dentro del microcontrolador para su manipulación a lo largo del proceso. Al introducir alguna cantidad a través del teclado, se debe tener la posibilidad de poder corregirla en dado caso que se halla cometido algún error, o bien aceptarla si no ha ocurrido tal hecho. De lo anterior se desprende la necesidad de contar con una tecla de borrado (clear) y otra que indique la aceptación de un dato (enter). El programa, por tanto, debe detectar en cualquier momento la ocurrencia de tales teclas y tomar diferentes vías de acción para cada caso.

Sólo un valor numérico puede ser aceptado como válido (números del 0 al 9), de tal forma que si se llega a oprimir alguna tecla de función, ésta no repercutirá en forma alguna en la cantidad. Además solo se aceptan como máximo 4 dígitos numéricos, si en dado caso se llegan a teclear más de 4 números antes de teclear enter (E), esos números no serán tomados en cuenta.

La visualización de los datos introducidos mediante el teclado, constituye un hecho de valiosa ayuda ya que el usuario puede detectar cualquier error de digitación en el momento mismo en que ocurre y proceder a su corrección, por lo que se hará uso de dos rutinas de manipulación de displays "Despliega Tecla" y "Borra Tecla" que serán estudiadas posteriormente.

Cabe aclarar antes de proceder a la realización de la rutina "Lee Teclado" la no correspondencia del teclado con el codificador empleado (C.I. 74C922). Efectivamente, el teclado empleado es de tipo matricial de 16 teclas, lo que si corresponde al C.I. 74C922, codificador para teclado matricial de 16 teclas. Sin embargo, la posición de las teclas no corresponde al valor obtenido a la salida del codificador por lo que sólo quedan dos caminos: diseñar un nuevo teclado, o bien, realizar un programa que lleve a cabo la transportación de una tecla a su valor real. Si nos vamos por el criterio monetario, la segunda opción es la más indicada.

La **Fig. 2.13** muestra la conexión de dos teclados al 74C922, siendo el de la derecha el empleado por el controlador.

Los valores obtenidos en el decodificador de la derecha serán los mismos que los obtenidos en el de la izquierda, solo

tomaremos estos valores y los transformaremos a los mostrados en la siguiente tabla:

```
0 ----- 0          9 ----- 3
1 ----- 1         10 ----- 6
----------        11 ----- 9
2 ----- 4         ----------
3 ----- 7         12 ----- 12
4 ----- A         13 ----- 13
----------        14 ----- 14
5 ----- 2         15 ----- 15
6 ----- 5
7 ----- 8
8 ----- B
```

Observamos de esta tabla que los valores 0, 1, 12, 13, 14 y 15 no necesitan ser transformados ya que si corresponden al valor deseado.

Podemos dividir, esta tabla en tres secciones a partir del número 2 hasta el 11 como se muestra a continuación:

```
2 ____ 4      5 ___ 2      9 ___ 3
3 ____ 7      6 ___ 5     10 ___ 6
4 ____ A      7 ___ 8     11 ___ 9
              9 ___ B
```

Si tomamos como base el primer número podemos conocer los otros, sumando 3 al número base de acuerdo a la posición que ocupa en memoria. Supongamos que almacenamos el valor de la tecla ya corregido en posiciones consecutivas de memoria, entonces sólo necesitamos una dirección base a la cual sumaremos el valor dado por el codificador, y obtendremos el valor correcto de la tecla.

Por ejemplo, si tecleamos la tecla ubicada en el renglón 2 columna 3, tendremos a la salida del codificador el número 9d = 1001b, si ubicamos la lista de números corregidos a partir de la dirección $052, sólo basta sumar 9h a 050h, que será la dirección base, para encontrar el valor correcto de la tecla, esto se muestra gráficamente en las siguientes líneas:

Dirección	Dato
052	04H
053	07H

054	0AH
055	02H
056	05H
057	08H
058	0BH
059	03H
05A	06H
05B	09H

$050H + 09H = $059 H

$059H contiene el número 03H que constituye la tecla tres en el teclado empleado en el controlador de tal forma que el usuario no tendrá ningún problema en el manejo del teclado.

Si se opta por diseñar un teclado acorde al C.I. 74C922, se tiene la ventaja de eliminar esta parte del programa de la EPROM y utilizarlo para implementar otro tipo de rutinas de mejoramiento.

Se puede implementar la sección corrección de tecla como se muestra en el diagrama de flujo de la **Fig. 2.14.**

Sintetizando, debemos destacar los siguientes puntos para el desarrollo de la rutina "Lee teclado":

1.- Transformar la tecla oprimida a su valor numérico correcto, excluyendo las teclas 0, 1 y de funciones que no necesitan ser transformadas.
2.- Solamente se aceptan valores numéricos del 0 al 9 y las teclas Enter (E), para aceptar un dato, y Clear (C), para borrar algún dígito de alguna cantidad.
3.- Pueden introducirse hasta un total de 4 números para cualquier parámetro.
4.- El valor de cada una de las teclas oprimidas será almacenada en posiciones consecutivas de memoria ocupando para ello un total de 4 bytes como máximo, empezando desde la dirección 041h hasta la 044h. Debido a la necesidad de otras rutinas (para realizar los diferentes tipos de control), se tendrá la necesidad de comprimir cada uno de los datos a tan sólo 2 bytes.
5.- Para llevar a cabo dicha comprensión, tomaremos la siguiente norma, el total de teclas oprimidas tendrá que estar ordenado, en las cuatro direcciones de memoria destinadas para ello, de tal forma que el dígito más significativo se encuentre en la dirección de memoria

Fig. 2.14.- Subrutina para corrección de tecla.

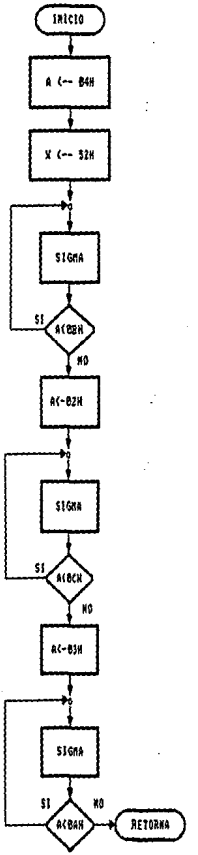


Fig. 2.15.- Subrutina para lectura de datos.

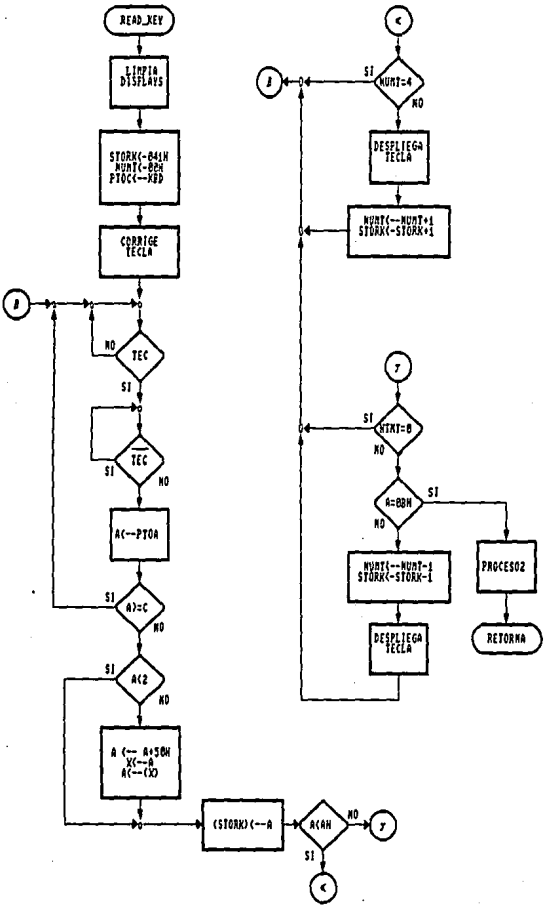
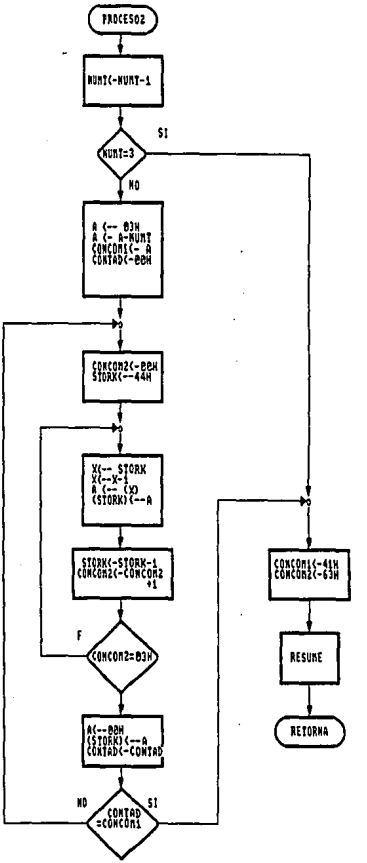


Fig. 2.16.- Función Proceso2.



más alta. Tomando en cuenta esto, sólo cuando se oprimen un total de cuatro números, estos quedarán en el orden convenido, teniendo que realizarse un reacomodo hacia arriba en la memoria, cuando el número de dígitos es menor de 4, introduciendo ceros en las posiciones de los dígitos más significativos.

6.- Cuando se oprima la tecla Enter, el total de dígitos tecleado se toma como válido y se acepta. Este valor solo podrá ser alterado, empezando nuevamente el proceso.

7.- En contraparte a la tecla enter (E), cuando sea detectada la tecla clear (C), el último dígito introducido no será tomado en cuenta, haciéndose un llamado a la rutina "Borra Tecla". Esta tecla puede ser oprimida en forma sucesiva hasta borrar totalmente los todos los dígitos introducidos.

8.- Finalmente se hará un paso de parámetros a la función de "Compresión de datos" que incluirá, la dirección inicial en la cual se almacenarán los datos, dirección $041 y la dirección de destino en la que será almacenado el dato en su forma de 2 bytes.

Para implementar el resto del programa , consistente en el reacomodo de teclas en memoria, necesitaremos el apoyo de una rutina adicional.

Los diagramas de flujo correspondientes se muestran en la **Fig. 2.15** y **Fig. 2.16** respectivamente.

El programa correspondiente en ensamblador se muestra en las siguientes líneas:

```
218     READ_KEY      JSR CLDIS
219                   LDA #$41
220                   STA STORK
221                   LDA #$00
222                   STA NUMT
223                   LDA KBD
224                   STA PTOC
225                   LDA #$04
226                   LDX #$52
227             PASO1 JSR SIGMA
228                   CMP #$0B
229                   BMI PASO1
230                   LDA #$02
231              PAS2 JSR SIGMA
232                   CMP #$0C
```

```
                BMI PAS2
                LDA #$03
        PAS3    JSR SIGMA
                CMP #$0A
                BMI PAS3
RE_READ         BRCLR 4,PTOA,RE_READ
CHECK_RE        BRSET 4,PTOA,CHECK_RE
                LDA PTOA
                AND #$0F
                CMP #$02
                BMI NEWKEY
                CMP #$0C
                BPL RE_READ
                ADD #$50
                TAX
                LDA X
NEWKEY          LDX STORK
                STA X
                CMP #$0A
                BPL FUNKEY
                LDA NUMT
                CMP #$04
                BEQ RE_READ
                JSR DISPLAY
                INC NUMT
                INC STORK
                BRA RE_READ
FUNKEY          LDX NUMT
                CPX #$00
                BEQ RE_READ
                CMP #$0B
                BEQ UNKEY
                DEC NUMT
                DEC STORK
                JSR RIGHT_DIS
                BRA RE_READ
UNKEY           DEC NUMT
                LDA NUMT
COMPAR          CMP #$03
                BEQ FINKEY
                LDA #$03
                SUB NUMT
                STA CONCOM1
                LDA #$00
                STA CONTAD
INIC            STA CONCOM2
                LDA #$44
                STA STORK
RECDER          LDX STORK
                DECX
                LDA X
```

```
284                 LDX STORK
285                 STA X
286                 DEC STORK
287                 INC CONCOM2
288                 LDX CONCOM2
289                 CPX #$03
290                 BNE RECDER
291                 LDA #$00
292                 LDX STORK
293                 STA X
294                 INC CONTAD
295                 LDX CONTAD
296                 CPX CONCOM1
297                 BNE INIC
298           FINKEY LDX #$41
299                 STX CONCOM1
300                 LDX #$63
301                 STX CONCOM2
302                 JSR RESUME
303                 RTS
```

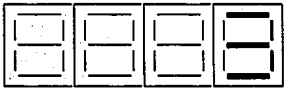
2.8.6. SUBRUTINA PARA DESPLEGADO DE DATOS (DISPLAY).

Esta parte del programa constituye un elemento de gran importancia ya que permite, en el momento de introducir los datos, visualizar en pantalla cada una de las teclas pulsadas, realizando un desplazamiento a la izquierda del dígito anterior para dar lugar a la última tecla pulsada, siempre y cuando ésta sea un valor numérico. Al introducir un parámetro dado, ya sea la temperatura final, el tiempo o bien el incremento de temperatura, se tiene la posibilidad de corroborar cada dígito que se teclea y en caso de ser incorrecto, borrarlo y teclear el número deseado. Cabe aclarar que esta rutina no borra un número incorrecto, esto es realizado por otra rutina que desplaza los dígitos una posición a la derecha en los displays cada vez que se borra un número pulsando la tecla clear (C).

La rutina para introducción de datos o parámetros hace uso del desplegado de dichos parámetros en los displays.

Para implementar esta rutina, hagamos de cuenta que elegimos control ON/OFF y que necesitamos introducir la temperatura a alcanzar o SET POINT. Esta temperatura es de 350° C.

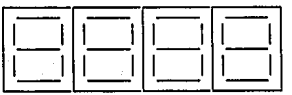
Fig. 2.17.- Desplegado de datos.



(A)

(B)

(C)



(D)

Tabla que muestra el procedimiento.

Contador de teclas	Tecla pulsada	Datos en memoria 041H	042H	043H	044H	Codigo a enviar por puerto B Binario	Hexadecimal
0	3	* 03H	00H	00H	00H	00001100	0CH
1	5	03H	* 05H	00H	00H	00010100 00001101	14H 0DH
2	8	03H	05H	* 08H	00H	00000000 00010101 00001110	00H 15H 0EH
3	E	03H	05H	08H	* 0EH	FIN	

* Apuntador de memoria

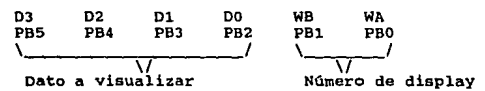
El procedimiento que se sigue es el siguiente (véase Fig. **2.17a, 2.17b** y **2.17c**):

1.- Tecleamos el número 3 y en los displays visualizamos dicho número a la derecha.
2.- Como paso siguiente tecleamos el número 5, y el dígito anterior se desplaza una posición hacia la izquierda.
3.- Finalmente introducimos el número 0 seguido de la tecla Enter (E).

Este proceso implica manipular cada uno de los dígitos almacenados en memoria (posiciones $041 hasta la $044) por el programa "Lee teclado". Por otro lado, para exhibir un número en los displays, necesitamos mandar el valor numérico así como la dirección del display en el cual será visualizado.

Ya que el puerto B se emplea como puerto de salida, se necesitarán 6 bits para mandar la información necesaria hacia el buffer de displays (C.I. 74LS670). Los dos bits menos significativos indicarán el número de display (del 0 al 3) y los 4 bits superiores, el valor numérico decimal a exhibir (del 0 al 9). Consideraremos el orden de los displays de derecha a izquierda, tal como se muestra en la **Fig. 2.17d**.

En un principio los displays se encuentran apagados, es decir, sin ningún dato en exhibición.



En nuestro ejemplo para visualizar el número 3 en el display 0, tendremos que enviar por tanto el código 001100 por el puerto B con el correspondiente código de habilitación de displays a través del puerto C, que no hace más que habilitar la escritura hacia el buffer de displays.

Dentro del programa "Lee teclado" se lleva a cabo el siguiente procedimiento, tomando en cuenta el ejemplo establecido.

Como podemos observar, el programa "Lee teclado" almacena cada una de las teclas oprimidas en posiciones consecutivas de memoria, en el orden en que dichas teclas se suceden por lo que el procedimiento despliega tecla resulta muy sencillo.

Debemos aclarar que cada vez que se pulsa una nueva tecla dentro del programa "Lee tecla", se hace un llamado a la rutina "Despliega tecla".

El diagrama de flujo correspondiente, nos permite un mejor entendimiento de como trabaja el procedimiento "Despliega tecla". Véase **Fig. 2.18.**

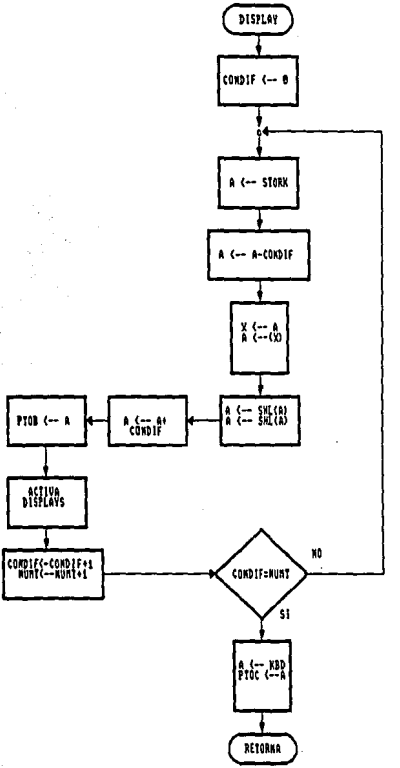
En las siguientes líneas se muestra el programa despliega tecla en lenguaje ensamblador del MC68705P3.

```
307         DISPLAY  LDA #$00
308                  STA CONDIF
309           NUMDIS LDA STORK
310                  SUB CONDIF
311                  TAX
312                  LDA X
313                  LSLA
314                  LSLA
315                  ADD CONDIF
316                  STA PTOB
317                  JSR ACTIVAD
318                  INC CONDIF
319                  LDA NUMT
320                  INCA
321                  CMP CONDIF
322                  BNE NUMDIS
323                  LDA KBD
324                  STA PTOC
325                  RTS
```

2.8.7. SUBRUTINA PARA ACTIVACION DE DISPLAYS (ACTIVAD).

Toda rutina que hace uso de los displays, requiere habilitar el buffer de displays para escritura del dato, lo cual implica mandar por el puerto C la palabra de control requerida, que para este caso se trata del número 02H. Debido a que la velocidad a la que trabaja el microcontrolador no es suficiente para que el dato enviado por el puerto B sea grabado en el buffer de displays

Fig. 2.18.- Subrutina para desplegado de datos.



(dispositivo más lento), se hace necesario generar un retardo; por lo que el algoritmo de trabajo consiste en enviar la palabra de control por el puerto C, una vez que el dato está presente en el puerto B, y generar un retardo adecuado.

El diagrama de flujo correspondiente se muestra en la **Fig. 2.19.**

El programa en ensamblador se ha implementado de la siguiente forma:

```
327    ACTIVAD      LDX REG_D
328                 STX PTOC
329                 LDX #$0F
330            WAIT2 DECX
331                 BNE WAIT2
332                 LDX #$0F
333                 STX PTOC
334                 RTS
```

2.8.8. SUBRUTINA PARA BORRAR DATO DEL DISPLAY (RIGTH_DIS).

Esta rutina viene a formar el complemento, junto con la rutina "Despliega tecla", para la manipulación de los displays en la introducción de los datos. Se genera un llamado a la misma, cuando la tecla clear es detectada, lo cual implica que se desea volver a introducir el último dígito y se procede a eliminarlo de los displays mediante un corrimiento a la derecha. El proceso es similar al que se empleó en la rutina "Despliega Tecla", y en la **Fig. 2.20** se presenta una tabla equivalente.

Para esta caso, supongamos que hemos tecleado el dato 350, para la temperatura final a alcanzar en control ON/OFF. En seguida nos damos cuenta, antes de teclear Enter (E), que la temperatura que deseamos es de 358° C y no 350° C, el proceso que se desarrolla por el programa "Lee Teclado", el cual hace uso del programa "Borra Display", entonces será:

A partir de que se detecta la tecla clear, vemos que el cero se ha eliminado de los displays y los números 3 y 5 se han recorrido una posición hacia la derecha, además de que el apuntador de memoria no se ha recorrido, lugar donde se almacenará el próximo número correcto.

Fig. 2.19.- Subrutina para activación de displays.

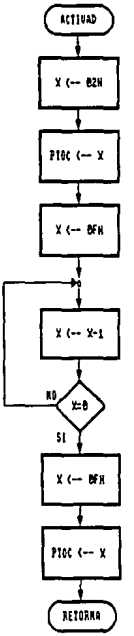
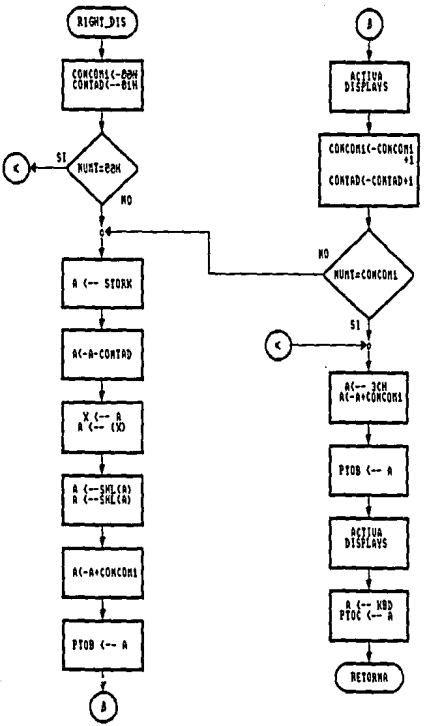


Fig. 2.20.- Tabla que muestra proceso borra tecla.

Contador de teclas	Tecla pulsada	Datos en memoria 841H	842H	843H	844H	Código a enviar por puerto B Binario	Hexadecimal
0	3	* 83H	86H *	88H	88H	00001100	0CH
1	5	83H	85H	88H	88H	00010100 00001101	14H 0DH
2	8	83H	85H	88H	88H	00000000 00010101 00001110	00H 15H 0EH
	C (Clear) = 84H	83H	85H	* 82H	84H	00010100 00001101	14H 0DH

* Apuntador de memoria

Fig. 2.21.- Subrutina para borrar datos de los displays.



Otro punto que cabe mencionar es que al recorrer una cantidad una posición a la derecha, es necesario apagar todos los segmentos del display iluminado más a la izquierda.

Una vez que se borra el número no deseado, el proceso continua normalmente dentro del programa "Lee Teclado" hasta que se detecta un Enter (E), pudiendo sucederse una serie de clear's y números tecleados antes de que se presente tal hecho.

Para una mejor visualización del proceso seguido en esta rutina se muestra el diagrama de flujo correspondiente en la **Fig. 2.21.**

Programa en ensamblador:

```
338     RIGHT_DIS     LDA #$00
339                   STA CONCOM1
340                   LDA #$01
341                   STA CONTAD
342                   LDX NUMT
343                   CPX #$00
344                   BEQ ZEROS
345           REWIND  LDA STORK
346                   SUB CONTAD
347                   TAX
348                   LDA X
349                   LSLA
350                   LSLA
351                   ADD CONCOM1
352                   STA PTOB
353                   JSR ACTIVAD
354                   INC CONCOM1
355                   LDA NUMT
356                   INC CONTAD
357                   CMP CONCOM1
358                   BNE REWIND
359             ZEROS LDA #$3C
360                   ADD CONCOM1
361                   STA PTOB
362                   JSR ACTIVAD
363                   LDA KBD
364                   STA PTOC
365                   RTS
```

2.8.9. SUBRUTINA PARA BLANQUEADO DE DISPLAYS (CLDIS).

Como su nombre lo indica, esta rutina tiene como propósito específico limpiar los displays, es decir, apagar todos los segmentos de cada uno de ellos, con el fin de exhibir alguna cantidad. El código de dato para el 74LS248 que logra el propósito de apagar todos los segmentos del display de 7 segmentos es el número 15 decimal, 1111 en binario. De esta forma, solo necesitamos enviar unos en los bits b2 hasta b5, que constituyen los bits del dato, y generar un contador desde 00H hasta 03H para apagar todos los displays. Esto se muestra en el diagrama de flujo de la **Fig. 2.22.**

Como observamos, el acumulador es inicializado a 3BH pero inmediatamente se incrementa al entrar al ciclo, por lo que su valor será 3CH=00111100 lo cual implica apagar todos los segmentos del display 00.

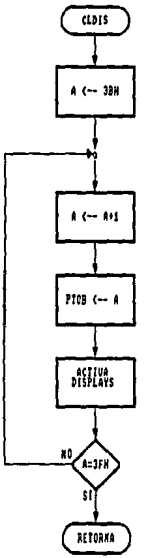
Programa en ensamblador:

```
369     CLDIS       LDA #$3B
370           WHITE INCA
371                 STA PTOB
372                 JSR ACTIVAD
373                 CMP #$3F
374                 BNE WHITE
375                 RTS
```

2.8.10. SUBRUTINA PARA COMPRESION DE DATOS (RESUME).

Debido al hardware utilizado, el máximo valor tolerado por el microcontrolador como se ha visto es de 9999 para el parámetro tiempo y 1020 para la temperatura. Estos valores son acomodados en localidades consecutivas, dígito por dígito, en orden de izquierda a derecha, es decir, en el orden natural en que son tecleados. Sin embargo, algunas rutinas requieren que estos datos se encuentren en forma comprimida, es decir, ocupando 2 bytes en lugar de 4. Por lo tanto se requiere de una rutina adicional que logre tal propósito. Dicha rutina requiere como parámetros, la dirección inicial en que se encuentra localizada la cantidad en su formato de 4 bytes, y la nueva dirección a partir de la cual se ubicará la cantidad en su forma reducida de dos bits.

Fig. 2.22.- Subrutina para blanqueado de displays.



El proceso a seguir se muestra en forma gráfica a continuación, suponiendo que se ha tecleado el número 3982 para el parámetro tiempo:

Mapa de memoria:
3982 en formato de 4 bytes

Dirección	Dato
041	03H
042	09H
043	08H
044	02H

Mapa de memoria:
3982 en formato de 2 bytes

Dirección	Dato
063	39H
064	82H

Como puede observarse, los datos se encuentran ubicados de tal forma que los dígitos más significativos se encuentran en las posiciones más bajas de memoria y los menos significativos en las más superiores.

El proceso a seguir para comprimir una cantidad en memoria, tan sólo consiste en desplazar el número en la posición $041 cuatro posiciones a la izquierda, así como también el que se encuentra ubicado en la dirección $043 y sumarles posteriormente los dígitos en las direcciones $042 y $044 respectivamente.

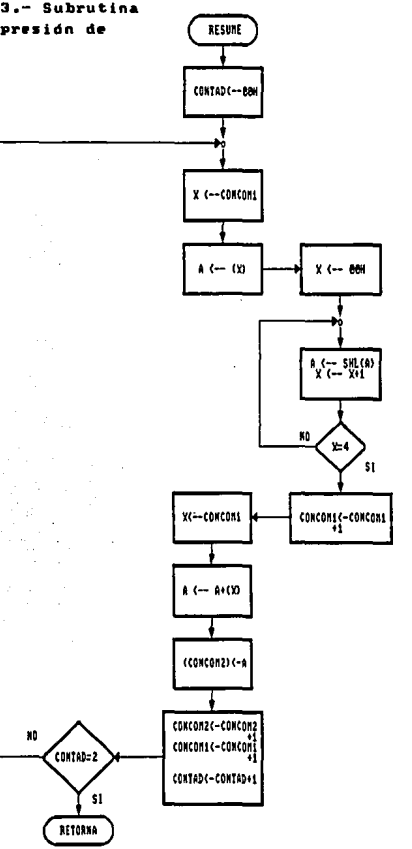
En nuestro ejemplo:

```
03h  --->  30h          08h  --->  80h
          +09h                    +02h
          ----                    -----
           39h                     82h
```

Esto implica además, que aunque los números son manejados como números binarios por el microcontrolador, el programa se encarga de manipularlos como si fueran números decimales.

La **Fig.2.23** muestra el diagrama de flujo de la rutina de compresión de datos.

Fig. 2.23.- Subrutina para compresión de datos.



Programa en ensamblador:

```
386     RESUME      LDA #$00
387                 STA CONTAD
388           ROTA  LDX CONCOM1
389                 LDA X
390                 LDX #$00
391                 CMP #$00
392                 BEQ SZER
393         DESPLZ  LSLA
394                 INCX
395                 CPX #$04
396                 BNE DESPLZ
397           SZER  INC CONCOM1
398                 LDX CONCOM1
399                 ADD X
400                 LDX CONCOM2
401                 STA X
402                 INC CONCOM2
403                 INC CONCOM1
404                 INC CONTAD
405                 LDX CONTAD
406                 CPX #$02
407                 BNE ROTA
408                 RTS
```

2.8.11. SUBRUTINA PARA EJECUCION DEL TIPO DE CONTROL (CONTROLES).

Esta subrutina realiza el trabajo de identificación de control a ejecutar y llama a la subrutina correspondiente. Vease la **Fig. 2.24.**

```
412     CONTROLES   LDA #$00
413                 STA PARAM
414                 CLI
415                 LDA TIPO
416                 CMP #$00
417                 BEQ CONTROL1
418                 CMP #$01
419                 BEQ CONTROL2
420                 CMP #$02
421                 BEQ CONTROL3
422                 CMP #$03
423                 BEQ CONTROL4
424                 RTS
425       CONTROL1  JSR TERMOMETRO
```

```
426                 RTS
427         CONTROL2 JSR ON_OFF
428                 RTS
429         CONTROL3 JSR PROPORC
430                 RTS
431         CONTROL4 JSR PROGRAM
432                 RTS
```

Primeramente, pone a 0 la variable PARAM y borra el suceso de Interrupción (líneas 412 a 414). Compara la variable TIPO con 0, 1, 2 y 3 [Termometro, On/Off, Proporc y Program respectivamente], ejecutando el control que le corresponde (líneas 415 a 432). Solo en el caso que no fuera un tipo valido retorna al principal [MENU] (línea 424).

2.8.12. SUBRUTINA PARA CARGA DEL TIEMPO DE RETARDO (CARGATPO).

El objetivo fundamental de la subrutina es refrescar el dato de tiempo en el temporizador, a fin de repetir una espera temporal. Vease la **Fig. 2.25.**

```
455     CARGATPO    LDA VTIME0
456                 ADD #$01
457                 STA TDR
458                 LDA VTIME1
459                 STA TIEMPO1
460                 LDA VTIME2
461                 STA TIEMPO2
462                 LDA VTIME3
463                 STA TIEMPO3
464                 LDA #$01
465                 STA TESP
466                 JSR RELOJ_ON
467                 RTS
```

De inicio, carga el TDR [Timer Data Register] con el valor capturado en la variable VTIME0 + 1 (líneas 455 a 457), debido a que la contabilización de tiempo no considera 0 como un instante individual y por ejemplo al poner 9 en el TDR, contará 8 segundos en lugar de 9.

Se asigna a las variables BCD de tiempo [TIEMPO1, TIEMPO2 y TIEMPO3] los valores capturados en las variables VTIME1, VTIME2 y VTIME3 (líneas 258 a 263).

Fig. 2.24.- Subrutina para ejecución del tipo de control.

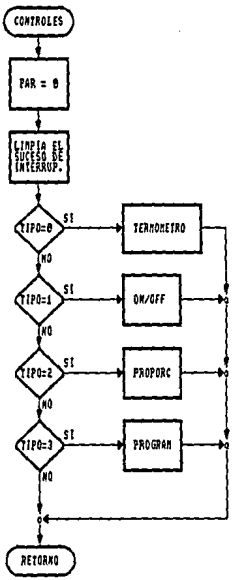
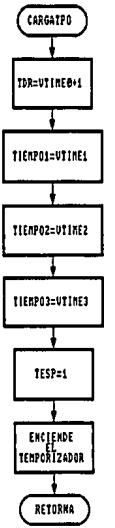


Fig. 2.25.- Subrutina para carga del tiempo de retardo.



Por último, se inicializa la variable TESP a 1 (líneas 464 y 465) y se enciende el temporizador (línea 466).

2.8.13. SUBRUTINA PARA MANTENER TEMPERATURA (MANTENT).

La subrutina para mantener la temperatura constante es básica en el Controlador Lineal Programable de Temperatura en lo que respecta al Control Proporcional y al Control Proporcional por Pasos Automáticos Temporizados. Para estos dos tipos de procesos se requiere que la temperatura permanezca constante hasta que el usuario así lo desee o bien, hasta un tiempo fijo establecido desde el inicio del proceso. Vease la **Fig. 2.26.**

El proceso se efectua tal como se muestra en las líneas 481 a la 511 del programa en ensamblador las cuales se explican con mayor detalle en los párrafos posteriores.

```
481    MANTENT      JSR LEE_KBD
482                 LDA #$03
483                 CMP TIPO
484                 BNE ALOTRO
485                 LDA TESP
486                 BEQ SALMAN
487          ALOTRO JSR LECTER
488                 LDA KBD
489                 STA PTOC
490                 BRSET 4,PTOA,VER_TPO
491          VER_BCD JSR DESP_BCD
492                 BRA VER_SIG
493          VER_TPO LDA TIPO
494                 CMP #$03
495                 BNE VER_BCD
496                 JSR DESP_TIME
497          VER_SIG LDA TP1
498                 CMP R1
499                 BMI MANTENT
500                 BHI CALENTAR
501                 LDA TP0
502                 CMP R0
503                 BLS MANTENT
504                 SUB R0
505                 CMP #$05
506                 BHI CALENTAR
507                 BRA MANTENT
```

```
508          SALMAN JSR RELOJ_OFF
509                 RTS
510        CALENTAR JSR ALCANZAT
511                 BRA MANTENT
```

En la línea 481 se hace un llamado a la subrutina LEE_KBD con el fin de detectar la sucesión de la tecla CLEAR en el bus de datos, lo cual significa que el usuario desea abortar el proceso e iniciar uno nuevo.

Como esta subrutina se emplea tanto para Control Proporcional como para Control Proporcional por Pasos Automáticos Temporizados es necesario distinguir entre una y otra para poder establecer el flujo a seguir (línea 482 y 493).

Si el proceso que se está llevando a cabo corresponde a Control Proporcional por Pasos Automáticos Temporizados, una vez alcanzada la temperatura requerida, esta se mantiene por un lapso de tiempo fijo.

En caso de que el usuario presione una tecla cualquiera, exceptuando la tecla CLEAR (línea 490), éste podrá ver en la pantalla de visualización de datos el tiempo transcurrido correspondiente, el cual una vez que llega a cero marca el fin de esta rutina.

Debido a que la temperatura tiende a seguir subiendo una vez que la etapa de potencia se apaga, se tomo como margen de seguridad +5, -5 °C el SET POINT. Si el sistema rebasa este margen, la etapa de potencia se enciende o se apaga con el fin de que la temperatura establecida se siga manteniendo.

En Control Proporcional la temperatura o SET POINT se mantiene hasta que el usuario así lo deseé, pues se puede abortar el proceso oprimiendo la tecla CLEAR.

2.8.14. SUBRUTINA INCREMENTO DE TEMPERATURA (INCTEM).

Al tener dentro del diseño una función capaz de incrementar su temperatura escalonadamente, se necesitaba de una rutina que pudiera sumar a la temperatura actual el incremento capturado. Véase la **Fig. 2.27.**

Fig. 2.26.- Subrutina para mantener temperatura.

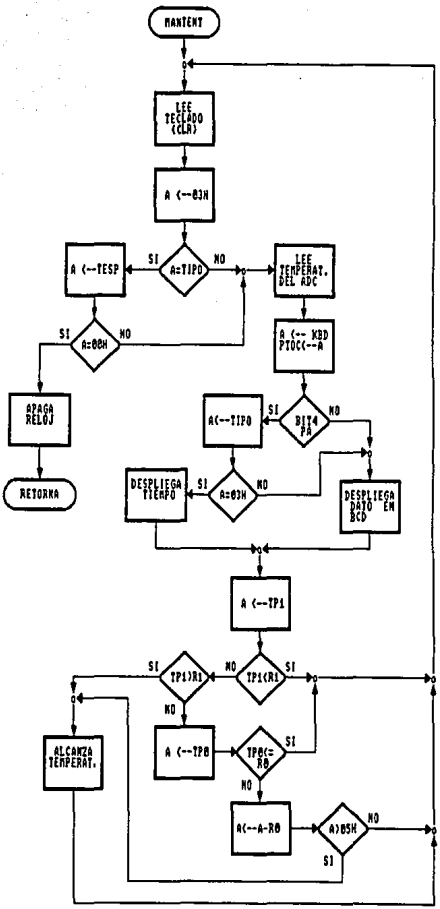
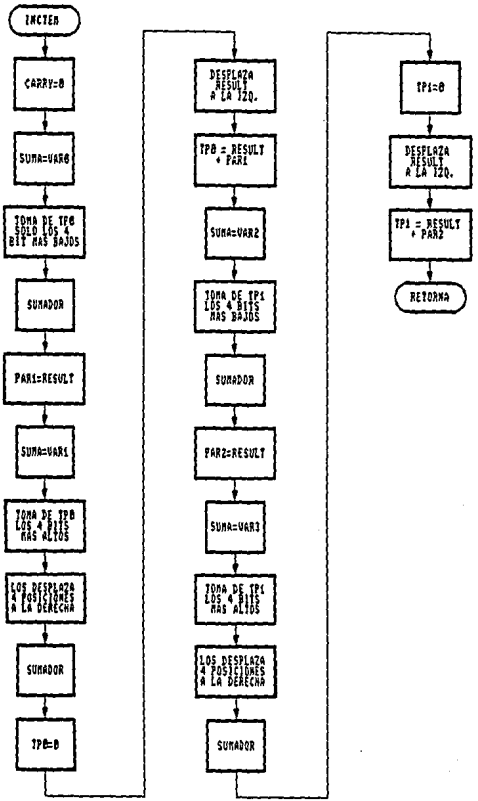


Fig. 2.27.- Subrutina para incremento de temperatura.



Si tomamos en cuenta que la temperatura actual esta almacenada como un BCD comprimido, es decir, 4 dígitos BCD en dos bytes, y los incrementos almacenados como BCD normales, entonces dicha rutina debe separar el BCD comprimido en sus partes correspondientes para poder efectuar la suma.

```
553     INCTEM      LDA #$00
554                 STA CARRY
555                 LDA VAR0
556                 STA SUMA
557                 LDA TP0
558                 AND #$0F
559                 JSR SUMADOR
560                 STA PAR1
561                 LDA VAR1
562                 STA SUMA
563                 LDA TP0
564                 AND #$F0
565                 JSR DESPLAZD
566                 JSR SUMADOR
567                 CLR TP0
568                 JSR DESPLAZI
569                 ADD PAR1
570                 STA TP0
571                 LDA VAR2
572                 STA SUMA
573                 LDA TP1
574                 AND #$0F
575                 JSR SUMADOR
576                 STA PAR1
577                 LDA VAR3
578                 STA SUMA
579                 LDA TP1
580                 AND #$F0
581                 JSR DESPLAZD
582                 JSR SUMADOR
583                 CLR TP1
584                 JSR DESPLAZI
585                 ADD PAR1
586                 STA TP1
587                 RTS
```

El incremento de temperatura requiere el manejo de un acarreo, el cual se almacena en la variable CARRY e inicialmente es puesto a 0 (líneas 553 y 554). Después, se asigna a la variable SUMA el valor capturado en la variable VAR0 (líneas 555 y 556) y a través de un filtro toma de TP0 los 4 bits menos

significativos [BCD 0] y los suma con el dato almacenado en SUMA (líneas 557 a 559).

El resultado, almacenado en el Acumulador, se guarda temporalmente en la variable PAR1 (línea 560). Almacena a continuación VAR1 en SUMA y toma de TP0 los 4 bits mas significativos [BCD 1] (líneas 561 a 564), desplaza 4 veces a la derecha el BCD 1 para volverlo unidad (línea 565) y poder sumarlo al valor de SUMA (línea 566). Se pone a cero la variable TP0 (línea 567) y el resultado obtenido de la suma se desplaza 4 bits a la izquierda (línea 568), a su posición original. Luego, se suman directamente el contenido del Acumulador y la variable PAR1 (línea 569) almacenándose el resultado en TP0 (línea 570).

Para los otros BCD's el proceso es similar, excepto por que se toma TP1, VAR2 y VAR3 (líneas 571 a 586).

2.8.15. SUBRUTINA PARA SUMA HEXADECIMAL (SUMADOR).

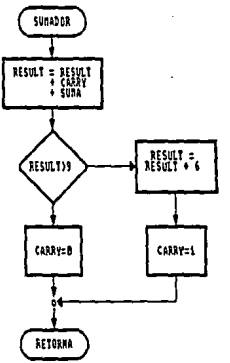
Esta subrutina efectúa la suma de los BCD's para el incremento de temperatura. Véase la **Fig. 2.28.**

```
589    SUMADOR      ADD CARRY
590                 ADD SUMA
591                 CMP #$09
592                 BHI SEIS
593                 LDX #$00
594                 STX CARRY
595                 BRA FINSUMA
596            SEIS ADD #$06
597                 LDX #$01
598                 STX CARRY
599        FINSUMA  AND #$0F
600                 RTS
```

Todo el proceso puede explicarse de la siguiente forma (líneas 589 a 590):

```
  BCDx              07h
+ SUMA            + 04h
+ CARRY           + 00h
-------           -----
  RESUL             0Bh (11d)
```

Fig. 2.28.- Subrutina para suma hexadecimal.



Como el resultado es mayor que nueve [requerimos un BCD] (línea 591), añadimos 6 y hacemos el nuevo CARRY = 1 (líneas 596 a 598), quedando

0Bh + 06H = 11h,

donde BCD = 01h y CARRY = 1, lo cual corresponde con el resultado esperado.

En el caso opuesto, CARRY = 0 y el resultado no se modifica (líneas 593 a 595). Finalmente, se filtran solamente los 4 bits menos significativos del resultado (líneas 599 y 600).

2.8.16. SUBRUTINA PARA DESPLAZAMIENTO BINARIO (DESPLAZAD/DESPLAZAI)

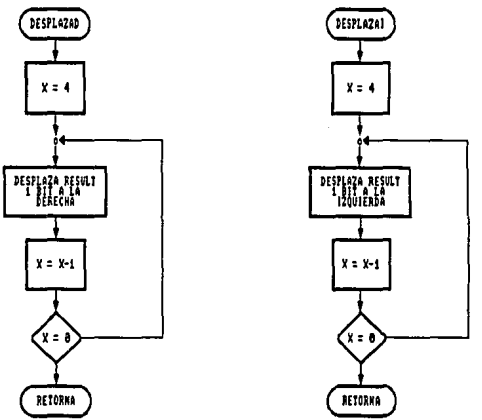
El desplazamiento consiste en mover los bits de un registro hacia la derecha o la izquierda dentro de sus posiciones. Así, por ejemplo:

el dato 00011000b (18h), al ser desplazado en ambas direcciones origina:

Desplaza 1 bit Izq.	00110000b (30h)
Desplaza 1 bit Izq.	01100000b (60h)
Desplaza 2 bits Der.	00011000b (18h)
Desplaza 3 bits Der.	00000011b (03h)

En nuestro caso particular, debido a que la temperatura parcial esta en formato BCD comprimido, cada byte contiene dos dígitos BCD. Por ello, los BCD almacenados en los bits mas significativos del byte deben desplazarse 5 posiciones a la derecha para quitarles peso y poderlos manejar como dígito BCD. Una vez operado, debe volverse a guardar en los bits mas significativos del byte, para lo cual se desplaza 5 posiciones a la izquierda. Por ejemplo, tenemos el BCD comprimido 70h e iniciemos el proceso de separarlo.

Fig. 2.29.- Subrutinas para desplazamiento binario.



Como primer punto, lo desplazamos 4 bits a la derecha

```
Desplaza 1 bit der.        01110000b (70h)
Desplaza 1 bit der.        00111000b (38h)
Desplaza 1 bit der.        00011100b (1Ch)
Desplaza 1 bit der.        00001110b (0Eh)
Desplaza 1 bit der.        00000111b (07h)
```

Lo cual por simple inspección, se nota que es cierto. Véase la **Fig. 2.29.**

```
602     DESPLAZD    LDX #$04
603           CICLOD LSRA
604                  DECX
605                  BNE CICLOD
606                  RTS
607
608     DESPLAZI    LDX #$04
609           CICLOI LSLA
610                  DECX
611                  BNE CICLOI
612                  RTS
```

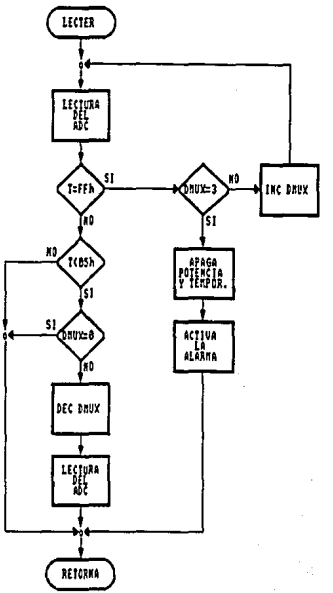
Ambas rutinas son similares, excepto por la dirección del desplazamiento. Así que solamente se explicará la subrutina DESPLAZAD.

Inicia poniendo el contador a 4 [5 ciclos] (línea 602), en seguida desplaza 1 bit a la derecha el dato (línea 603), decrementa el contador y checa si es cero (líneas 604 y 605), si lo es, termina la rutina. De lo contrario itera.

2.8.17. SUBRUTINA PARA LECTURA DE TEMPERATURA (LECTER).

La lectura de Temperatura implica leer el ADC y determinar el rango dentro del cual se ubica la temperatura correcta. Es decir, si tuviéramos una temperatura de 100 °C, el ADC regresa 064H por la entrada 0 y 000H por las entradas 1, 2 y 3. Entonces, el dato de temperatura correcto se obtiene de la entrada 0 (por lo cual la variable DMUX inicia siendo 0). Pero, si la temperatura fuera de 400 °C, entonces el ADC regresa 0FFH por la entrada 0, 090H por la entrada 1 y 000H por la restantes; lo cual nos indica que el dato correcto se obtiene de la entrada 1.

Fig. 2.30.- Subrutina para lectura de temperatura.



Partiendo de los dos ejemplos anteriores y recordando que existe una relación 1 : 1 entre temperatura analógica (decimal) y temperatura digitalizada (Hexadecimal), se tiene:

Temp. Correcta = [DMUX] [Dato de ADC]

es decir, para el caso de los 100 °C tenemos que

100d = 064h,

entonces,

Temp. Correcta = [0] [64],

lo cual nos indica que la temperatura correcta se encuentra en la tomando la entrada 0 (DMUX = 0). Si lo efectuamos con una temperatura de 967 °C tenemos,

967d = 3C7h,

de donde,

Temp. Correcta = [3] [C7],

en otras palabras, la temperatura correcta se encuentra en la entrada 3 del ADC (DMUX = 3). Véase la **Fig. 2.30.**

```
616    LECTER      JSR LEE_ADC
617                LDA T
618                CMP #$FF
619                BEQ POR_AR
620                CMP #$00
621                BEQ POR_AB
622          SAL_TER RTS
623           POR_AR LDA DMUX
624                  CMP #$03
625                  BEQ ERROR_L
626                  INC DMUX
627                  BRA LECTER
628           POR_AB LDA DMUX
629                  BEQ SAL_TER
630                  DEC DMUX
631                  JSR LEE_ADC
```

```
632                 BRA SAL_TER
633         ERROR_L JSR RELOJ_OFF
634                 JSR POTENCIA_OFF
635                 LDA #$02
636                 JSR ALARMA
637                 RSP
638                 JMP MENU
```

En un principio, la subrutina LECTER lee la temperatura del ADC, guardada en la variable "T", de la entrada que indiqué DMUX y la cual es la última accesada (línea 616); compara el dato de temperatura con OFFh para determinar si la temperatura ya superó el rango actual y es necesario tomarla del rango inmediato superior (líneas 617 a 619) o si por el contrario es igual a cero, entonces debe tomarse del rango inmediato inferior (líneas 620 y 621).

Para el caso de tomar el rango inmediato superior se verifica que la variable DMUX no sea igual a 3 antes de incrementarla (líneas 623 a 625), ya que en ese caso la temperatura salió por completo del control del equipo, debiéndose apagar la etapa de potencia, el temporizador y activar la alarma en modo continuo (líneas 633 a 638). Si DMUX es correcto se vuelve a iterar (líneas 626 a 627).

En el caso de tomar el rango inmediato inferior, se compara la variable DMUX con 0 antes de decrementarla (líneas 628 y 629), ya que de ser así, la temperatura es correcta y DMUX es igual a 0. En caso contrario efectúa el decremento de DMUX y vuelve a leer el ADC, pero sin iterar (líneas 630 a 632).

2.8.18. SUBRUTINA PARA ENCENDIDO Y APAGADO DE CRONOMETRO (RELOJ_ON/RELOJ_OFF).

Estas subrutinas se encargan de encender y apagar el temporizador del MCU. El temporizador es un circuito independiente diseñado para poder ser impulsado por una fuente distinta a la del MCU. En el diseño se implementó una fuente de 1 Hz, generando un temporizador de tiempo real, capaz de cronometrar tiempos hasta de 9999 segundos. Véase la **Fig. 2.31.**

```
640     RELOJ_ON    BSET 4,TCR
641                 BCLR 6,TCR
642                 RTS
```

Fig. 2.31.- Subrutinas para encendido apagado de cronometro.

```
643     RELOJ_OFF     BCLR 4,TCR
644                   BSET 6,TCR
645                   RTS
```

Para el encendido del temporizador tenemos, que primero habilita el TIE [Timer Input Enable, bit 4 del TCR] (línea 640) lo cual permite que entre al temporizador la señal del reloj. En seguida, apaga el TIR [Timer Interrupt Request, bit 6 del TCR] (línea 641) con lo que el temporizador comienza a decrementar el contenido del TDR hasta llegar a 0.

Para el apagado del temporizador la situación se presenta de manera inversa, pues se deshabilita el TIE (línea 643) y se habilita el TIR (línea 644), con lo cual el temporizador se detiene.

2.8.19. SUBRUTINA PARA ENCENDIDO Y APAGADO DE POTENCIA (POTENCIA_ON/POTENCIA_OFF).

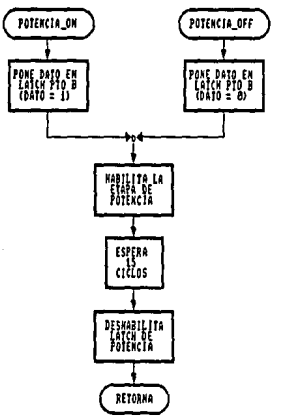
El objetivo de estas subrutinas es encender o apagar la etapa de potencia. Véase la **Fig. 2.32.**

```
647     POTENCIA_ON   LDA #$01
648                   BRA SPOT
649     POTENCIA_OFF  LDA #$00
650           SPOT    STA PTOB
651                   LDA POT
652                   STA PTOC
653                   LDA #$0F
654         RETPOT    DECA
655                   BNE RETPOT
656                   LDA #$FF
657                   STA PTOC
658                   RTS
```

Para el caso del encendido de la etapa de potencia, se pone en el Puerto B un valor 01h, mientras que para apagarla se pone en dicho puerto un valor 00h (líneas 647 a 650).

Indistintamente, de si se apaga o enciende, se habilita el latch de la etapa de potencia para que tome el dato del puerto B (líneas 651 y 652). A continuación, se efectúa un ciclo de espera

Fig. 2.32.- Subrutina para encendido apagado de potencia.



para sincronizar el MCU con el latch (líneas 653 a 655) y al terminar se deshabilita el latch (líneas 656 y 657).

2.8.20. SUBRUTINA PARA ALCANCE DE TEMPERATURA (ALCANZAT).

Esta subrutina se encarga de alcanzar la temperatura indicada por el usuario (SET POINT) en las rutinas ON/OFF, PROPORCIONAL Y PAT, véase la **Fig. 2.33.** Su código se muestra a continuación:

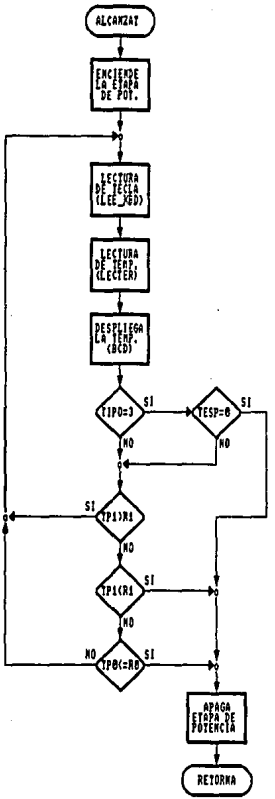
```
661    ALCANZAT     JSR POTENCIA_ON
662         ALCANCE JSR LEE_KBD
663                 JSR LECTER
664                 JSR DESP_BCD
665                 LDA #$03
666                 CMP TIPO
667                 BNE NOEST
668                 LDA TESP
669                 BEQ SALTO
670           NOEST LDA TP1
671                 CMP R1
672                 BMI SALTO
673                 BHI ALCANCE
674                 LDA TP0
675                 CMP R0
676                 BLS SALTO
677                 BHI ALCANCE
678           SALTO JSR POTENCIA_OFF
679                 RTS
```

Inicialmente se enciende la etapa de potencia (línea 661) y sensa el teclado, para determinar una interrupción de usuario, (línea 662). A continuación se lee la temperatura actual del sistema y se despliega el dato (líneas 663 y 664).

En seguida, se determina si el tipo de control es PAT para evaluar el fin de tiempo de espera y continuar con otra rampa (líneas 665, 666, 668, 669, 678 y 679). En caso contrario, compara la temperatura leida con el SET POINT para determinar si debe continuar el calentamiento o apagar la potencia y regresar al punto de llamada (líneas 670 a 679).

Restaría señalar que la temperatura se almacena en un BCD comprimido, a fin de evitar sobrecálculos y eliminar el numero de

Fig. 2.33.- Subrutina para alcance de temperatura.



comparaciones necesarias para determinar si se alcanzo el SET POINT.

2.8.21. SUBRUTINA PARA ACCIONAMIENTO DE ALARMA (ALARMA).

Esta subrutina se encarga de manejar la Alarma del sistema tomando del acumulador el tipo que se desee accionar, véase la Fig. 2.34. El código correspondiente se muestra a continuación:

```
682    ALARMA      STA PTOB
683                LDX ALARM
684                STX PTOC
685                LDX #$0F
686        ESPERA1 DECX
687                BNE ESPERA1
688                LDX #$0F
689                STX PTOC
690                CMP #$01
691                BEQ CONTFL
692                JSR ESPTEC
693         CONTFL LDA #$00
694                STA PTOB
695                LDA ALARM
696                STA PTOC
697                LDA #$0F
698        ESPERA2 DECA
699                BNE ESPERA2
700                LDA #$0F
701                STA PTOC
702                RTS
```

Básicamente, toma del acumulador el tipo de alarma y lo pone en el puerto B, activa el latch de datos de la alarma para que tome el tipo, efectúa una espera y lo desactiva (líneas 682 a 689).

A continuación apaga la alarma, dependiendo del tipo elegido, directamente o después de oprimir una tecla (líneas 690 a 693). Nuevamente se repite el proceso de poner en el puerto B el dato de apagado (0), activar el latch, efectuar la espera y desactivar el latch (líneas 693 a 702), mediante los cuales se completa el proceso de apagado.

Fig. 2.34.- Subrutina para accionamiento de la alarma.

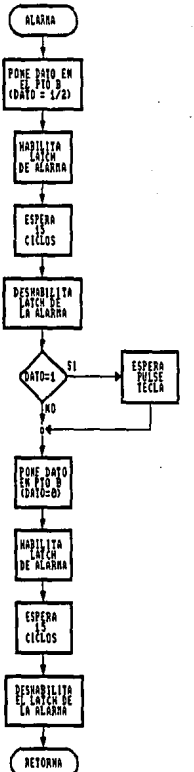


Fig. 2.35.- Subrutina para espera de tecla.

2.8.22. SUBRUTINA PARA ESPERA DE TECLA (ESPTEC).

Esta subrutina es usada únicamente por la subrutina ALARMA y su tarea es detectar la opresión de una tecla por el usuario, véase la **Fig. 2.35.**

```
704    ESPTEC       LDA KBD
705                 STA PTOC
706           ESPT1 BRCLR 4,PTOA,ESPT1
707           ESPT2 BRSET 4,PTOA,ESPT2
708                 LDA PTOA
709                 AND #$0F
710                 LDX #$0F
711                 STX PTOC
712                 RTS
```

Habilita el teclado y espera hasta que sea presionada una tecla (líneas 704 a 707). Toma el dato del puerto A y deshabilita el teclado (líneas 708 a 712), regresando al punto de llamada en la subrutina ALARMA.

2.8.23. SUBRUTINA PARA SENSADO DE CANCELACION (LEE_KBD).

El objeto de tener una subrutina que detecta o espera la pulsación de una tecla, está relacionada con la interface Usuario - Controlador. Ya que de esta manera se brinda al usuario un medio para romper el flujo normal del Controlador o su aceptación para continuar otro proceso. Véase la **Fig. 2.36.**

```
715    LEE_KBD      LDA PARAM
716                 CMP #$00
717                 BNE PORESP
718                 LDA KBD
719                 STA PTOC
720                 LDA #$04
721       DELAYKBD  DECA
722                 BNE DELAYKBD
723                 LDA PTOA
724                 AND #$0F
725                 LDX #$0F
726                 STX PTOC
727                 BRA OTROL
728         PORESP  JSR ESPTEC
729          OTROL  CMP #$04
```

```
730                 BEQ ABORTA
731                 RTS
732           ABORTA JSR RELOJ_OFF
733                 JSR POTENCIA_OFF
734                 LDA #$00
735                 STA PARAM
736                 RSP
737                 JMP MENU
```

A fin de identificar alguno de los dos modos de operación (Detección de Tecla o Espera de Tecla), se guarda en PARAM el tipo deseado antes de entrar a la subrutina LEE_KBD. En ella se compara esta variable con 0 (líneas 715 y 716), si se cumple, entonces se ejecuta la detección de la tecla pulsada (líneas 718 a 727) o de lo contrario, se espera a que el usuario pulse una tecla (línea 728).

Una vez que se tiene el código de la tecla pulsada, se compara con CLEAR [Abortado de Proceso, CLEAR = 4] (líneas 729 y 730) para establecer si el usuario desea terminar el proceso actual, en cuyo caso, se apaga el temporizador, la etapa de potencia y se regresa directamente a la petición de tipo de control [MENU] (líneas 732 a 737).

En el caso que la tecla fuera distinta de CLEAR, se regresa normalmente al punto de llamada (línea 731).

2.8.24. SUBRUTINA PARA LECTURA DE CONVERTIDOR A/D (LEE_ADC).

La presente subrutina efectúa la lectura del ADC, lo cual implica elección de entrada, arranque de conversión, espera de fin de conversión y toma de dato, véase la **Fig. 2.37.** A continuación se muestra el código correspondiente:

```
740     LEE_ADC       LDA DMUX
741                   STA PTOB
742                   LDA MUX_A
743                   STA PTOC
744                   LDA #$0F
745           DELAY1  DECA
746                   BNE DELAY1
747                   LDA #$00
748                   STA PTOC
749                   LDA #$1E
```

```
750          DELAY2 DECA
751                 BNE DELAY2
752                 LDX #$0F
753                 STX PTOC
754                 LDA #$1E
755          DELAY3 DECA
756                 BNE DELAY3
757                 LDA #$04
758                 STA PTOC
759           FINCV BRCLR 0,PTOA,FINCV
760                 LDA OEAD
761                 STA PTOC
762                 LDA #$27
763          DELAY4 DECA
764                 BNE DELAY4
765                 LDA PTOA
766                 STA T
767                 LDA #$0F
768                 STA PTOC
769                 LDA DMUX
770                 STA R1
771                 LDA T
772                 STA R0
773                 JSR CONVER
774                 RTS
```

Dentro de la programación, la elección de entrada se efectúa tomando el valor de la variable DMUX, la cual es incrementada o decrementada de acuerdo al nivel de temperatura presente (líneas 740 y 741). Seguidamente, se habilita el latch del MUX y se efectúa un retardo para permitirle que tome el dato (líneas 742 a 746).

En seguida, se envía la señal de arranque de conversión (START) al ADC, se efectúa otro retardo para dar la amplitud necesaria a esta señal, siendo desconectada al final del retardo (líneas 747 a 756).

Para este momento, mientras se lleva a cabo el proceso de conversión de temperatura, se activa el sensado de la señal de fin de conversión (EOC) y efectúa un retardo para que el latch tome el valor correspondiente (líneas 757 a 765).

Después del paso anterior se tiene ya en el acumulador el valor de temperatura monitoreado. Por lo cual, el dato es

Fig. 2.36.- Subrutina para sensar cancelación de proceso.

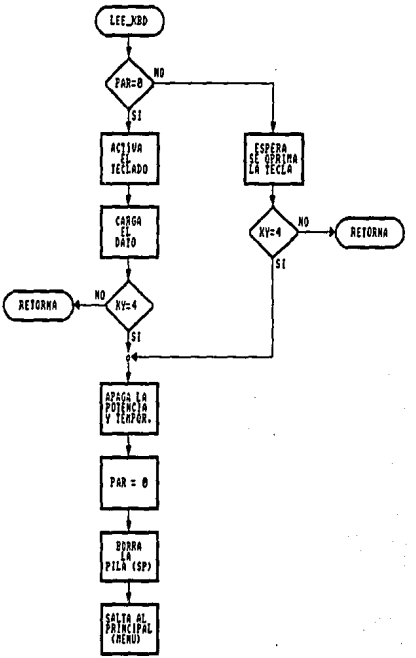
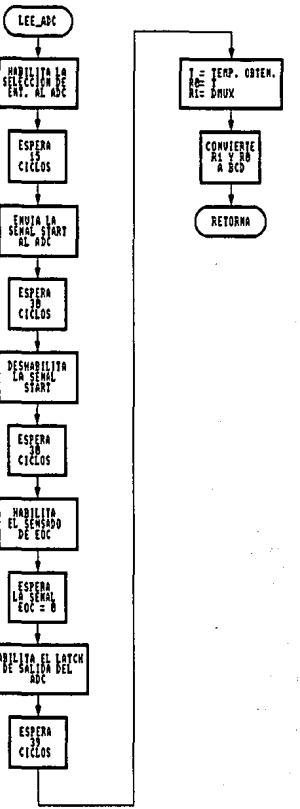


Fig. 2.37.- Subrutina para lectura de temperatura del ADC.



asignado a las variables que le corresponden:

T --> R0 y DMUX --> R1

para ser pasadas como parámetros a la función CONVER, ya explicada (líneas 766 a 774).

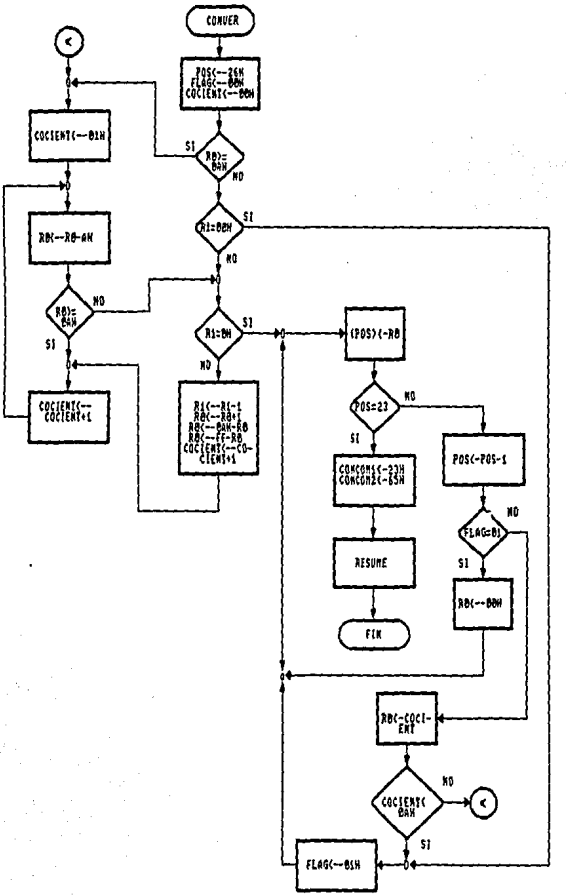
2.8.25. SUBRUTINA PARA CONVERSION HEX/BCD (CONVER).

Es muy importante para todo tipo de proceso que realicemos, establecer el sistema de numeración a emplear con el fin de evitar errores en los datos y tratar de utilizar eficiéntemente la memoria optimizándola al máximo. Haciendo un análisis del sistema numérico empleado por cada uno de los periféricos podemos llegar a una conclusión definitiva. Sabemos que de hecho, el microcontrolador trabaja con valores binarios, pero que sin embargo, nosotros como diseñadores o usuarios, estamos en la posibilidad de trabajar diferentes sistemas numéricos mediante el diseño de programas adecuados.

Por el lado de los dispositivos de entrada, se cuenta con dos elementos: el teclado y el ADC. El teclado junto con el codificador, al igual que el ADC, reportan a la entrada del microcontrolador valores hexadecimales. Sin embargo, los valores de los diferentes parámetros válidos introducidos a través del teclado, se limitan a los dígitos del 0 al 9 para las diferentes posiciones relativas de una cantidad. Por otra parte, del lado de la salida, los dígitos expuestos en los displays también involucran un sistema BCD. Por este hecho se ve la ventaja de emplear un sistema de numeración BCD, haciéndose necesaria una rutina que transforme un número hexadecimal a BCD para los valores reportados por el ADC.

Emplearemos el método de la división por restas sucesivas, el cual consiste en restar a la cantidad hexadecimal el valor Ah en forma sucesiva, llevando un conteo de las restas lo cual constituye el cociente de la división. Cuando se tiene una diferencia menor a Ah, este residuo constituye el dígito BCD menos significativo. Posteriormente se toma el cociente y se le empieza a aplicar el método anterior, es decir, se le restará sucesivamente Ah llevando una nueva cuenta hasta que la diferencia sea menor a Ah, y esta diferencia constituirá el siguiente dígito BCD.

Fig. 2.38.- Subrutina para conversión HEX-BCD.



El proceso continua de la misma forma hasta que se obtiene un cociente menor a Ah el que pasará a ser el dígito más significativo del valor BCD.

Por ejemplo, transformemos el número 67h, que en decimal es el número 103d, a BCD:

```
67h-Ah = 5Dh      COCIENTE= 1
5Dh-Ah = 53h      COCIENTE= 2
53h-Ah = 49h      COCIENTE= 3
49h-Ah = 3Fh      COCIENTE= 4
3Fh-Ah = 35h      COCIENTE= 5
35h-Ah = 2Bh      COCIENTE= 6
2Bh-Ah = 21h      COCIENTE= 7
21h-Ah = 17h      COCIENTE= 8
17h-Ah = 0Dh      COCIENTE= 9
0Dh-Ah = 03h      COCIENTE= A    03h < Ah ---> DLS = 3 BCD
0Ah-Ah = 00h      COCIENTE= 1    00h < Ah
                                 Siguiente dígito = 00h
                          COCIENTE < Ah  ----> DMS = 1 BCD
```

y el número BCD resultante es 103, el cual corresponde al número decimal calculado.

El ADC reporta una cantidad hexadecimal de 1 byte, es decir, 2 dígitos hexadecimales, sin embargo, el hardware para lectura de temperatura está implementado para trabajar por rangos, lo cual requiere a lo sumo dos bytes, así que la rutina de transformación deberá tomar en cuenta esta característica.

Debido a que la máxima temperatura tolerada por el controlador es de 1020 °C, se necesitarán 4 bytes para ubicar cada dígito BCD generado, haciendose un llamado a la rutina de "Compresión de Datos", para reducir la cantidad a 2 bytes.

El diagrama de flujo se muestra en la **Fig. 2.38** y el programa en ensamblador en las líneas siguientes:

```
777     CONVER      LDA #$26
778                 STA POS
779                 LDA #$00
780                 STA FLAG
781                 STA COCIENT
782         CONV_BCD LDA R0
```

```
783                 BMI CICLO
784                 CMP #$0A
785                 BPL CICLO
786                 LDX R1
787                 CPX #$00
788                 BEQ ALMAC
789                 CMP #$0A
790                 BMI SMSB
791           CICLO LDX #$01
792                 STX COCIENT
793           RESTA SUB #$0A
794                 BMI IGNORE
795                 CMP #$0A
796                 BPL IGNORE
797            SMSB LDX R1
798                 CPX #$00
799                 BEQ CHECA_C
800                 DEC R1
801                 INCA
802                 STA REGTEM
803                 LDA #$0A
804                 SUB REGTEM
805                 STA REGTEM
806                 INC COCIENT
807                 LDA #$FF
808                 SUB REGTEM
809          IGNORE INC COCIENT
810                 BRA RESTA
811        CHECA_C BRA STORE
812             P2 LDA COCIENT
813                 CMP #$0A
814                 BMI ALMAC
815                 BRA CICLO
816          ALMAC LDX #$01
817                 STX FLAG
818          STORE LDX POS
819                 STA X
820                 CPX #$23
821                 BEQ FIN
822                 DEC POS
823                 LDX FLAG
824                 CPX #$01
825                 BEQ SIG
826                 BRA P2
827            FIN LDX #$23
828                 STX CONCOM1
829                 LDX #$65
830                 STX CONCOM2
831                 JSR RESUME
832                 RTS
833            SIG LDA #$00
```

```
834                     BRA STORE
```

2.8.26. SUBRUTINA PARA DESPLEGADO DE BCD'S (DESP_BCD).

Esta subrutina se encarga de desplegar algún dato BCD en cada uno de los displays según le corresponda de acuerdo a su posición absoluta. Véase la **Fig. 2.39.** El código correspondiente en lenguaje ensamblador del MC68705P3 se muestra en las siguientes líneas:

```
837     DESP_BCD    LDA #$00
838                 STA CAUX
839                 LDA R0
840                 JSR PROCD1
841                 LDA R0
842                 JSR PROCD2
843                 LDA R1
844                 JSR PROCD1
845                 LDA R1
846                 JSR PROCD2
847                 RTS
```

La subrutina inicia poniendo a cero la variable correspondiente al número del display en el cual será desplegado el dato. Se cargan los primeros dos dígitos BCD, comprendidos en un solo byte y posteriormente a través de las rutinas PROCD1 y PROCD2 se hace la separación de los mismos y se les agrega el número de display en el cual serán visualizados. El mismo procedimiento se aplica para los dos dígitos BCD más significativos

2.8.27. SUBRUTINA PARA DESPLEGADO DE TIEMPO (DESP_TIME).

Esta subrutina efectúa el desplegado del cronómetro del sistema (4 dígitos), véase la **Fig. 2.40.** Su código se muestra a continuación:

```
849     DESP_TIME   LDA #$00
850                 STA CAUX
851                 LDA TDR
852                 SUB #$01
853                 JSR PROCD1
```

```
854                 LDA TIEMPO1
855                 JSR PROCD1
856                 LDA TIEMPO2
857                 JSR PROCD1
858                 LDA TIEMPO3
859                 JSR PROCD1
860                 RTS
```

La subrutina inicia poniendo a cero el contador de dígito a desplegar (CAUX) y toma del TDR el dígito menos significativo de tiempo, al cual se le resta 1 para representar el tiempo real (líneas 849 a 852). Ya con el dato en el acumulador se llama a la función PROCD1, encargada de desplegar el numero indicado en el display que le corresponde (línea 853).

Para los restantes 3 dígitos (TIEMPO1, TIEMPO2 y TIEMPO3) el proceso consiste únicamente en cargarlos al acumulador y llamar a la subrutina PROCD1 (líneas 854 a 860).

2.8.28. SUBRUTINA PARA MANEJO DE CRONOMETRO (TIMER).

La temporización (cronómetro) del sistema se maneja a través de esta subrutina. La temporización implementada en el µControlador se limita a manejar un byte, por lo cual para usar un valor de tiempo máximo de 9999 segundos se aprovechó la interrupción de temporizador. Véase la **Fig. 2.41.**

La idea básica para el manejo fue la siguiente:

tenemos un tiempo de 234 segundos, por ejemplo, pero el temporizador únicamente decrementará el dígito menos significativo (4) hasta llegar a 0. Cuando el TDR llega a cero se desvia el flujo a esta subrutina, donde se checa si el tiempo final es cero y en caso contrario seguir decrementándolo. Así tenemos que después de cada interrupción el tiempo pasa de 234 inicial a 229, 219, 209, 199 ... 000. En este momento, se detiene el temporizador y debe señalizarse que el tiempo fue completado.

Fig. 2.39.- Subrutina para desplegado de BCD's.

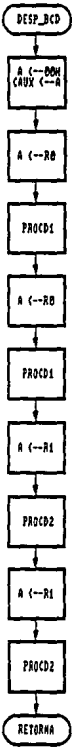


Fig. 2.48.- Subrutina para desplegar el tiempo.

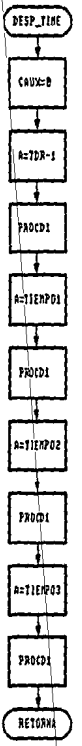
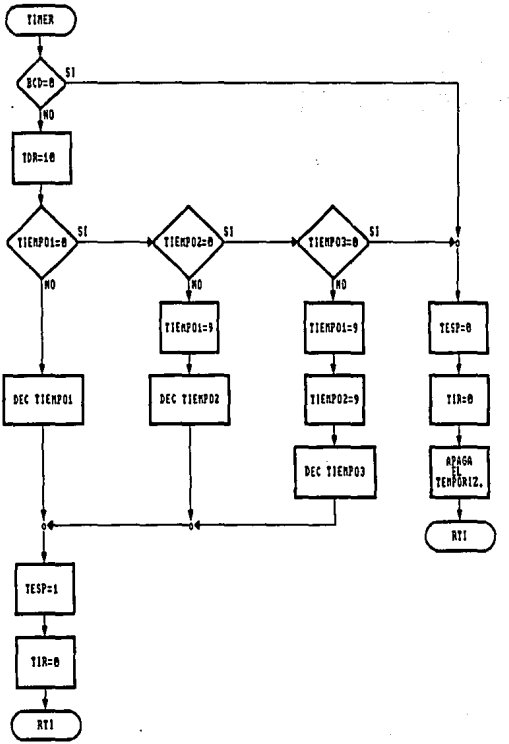


Fig. 2.41.- Subrutina para manejo de cronómetro.



A continuación se muestra el código correspondiente a esta subrutina:

```
863    TIMER        LDA TIEMPO1
864                 ORA TIEMPO2
865                 ORA TIEMPO3
866                 BEQ FINTEMP
867                 LDA #$0A
868                 STA TDR
869                 LDA TIEMPO1
870                 BEQ VERS
871                 DEC TIEMPO1
872         CONTEMP LDA #$01
873                 STA TESP
874                 BCLR 7,TCR
875                 RTI
876            VERS LDA TIEMPO2
877                 BEQ VERSS
878                 LDA #$09
879                 STA TIEMPO1
880                 DEC TIEMPO2
881                 BRA CONTEMP
882           VERSS LDA TIEMPO3
883                 BEQ FINTEMP
884                 LDA #$09
885                 STA TIEMPO1
886                 STA TIEMPO2
887                 DEC TIEMPO3
888                 BRA CONTEMP
889         FINTEMP LDA #$00
890                 STA TESP
891                 BCLR 7,TCR
892                 JSR RELOJ_OFF
893                 RTI
```

Inicialmente se checa si el tiempo total es cero, para deshabilitar el temporizador y señalizar fin de tiempo (líneas 863 a 866 y 889 a 893).

En caso contrario, se decrementa el tiempo total (3 dígitos) de la siguiente manera:

1) Si la variable TIEMPO1 = 0, entonces se decrementa la variable siguiente, de lo contrario, se decrementa TIEMPO1.

2) Si la variable TIEMPO2 = 0, entonces se decrementa la variable TIEMPO3, de lo contrario decrementamos la variable TIEMPO2 y hacemos TIEMPO1 = 9.

3) Finalmente, si la variable TIEMPO3 es cero (caso no usual, pero que pudiera suceder) se deshabilita el temporizador y se activa la bandera de fin de tiempo. En caso contrario, se decrementa TIEMPO3 y las variables TIEMPO2 y TIEMPO1 se igualan a 9.

Este proceso puede observarse en las líneas 867 a 888, ademas debe señalarse que el TDR para estos casos es siempre igual a 10 (9+1).

2.8.29. SUBRUTINAS PARA PREPARAR DATO A VISUALIZAR (PROCD1 Y PROCD2).

PROC1 y PROC2 son rutinas elaboradas para exhibición de los datos en los displays cuyo código en ensamblador se muestra en las siguientes líneas:

```
897    PROCD1      LSLA
898                LSLA
899                AND #$3C
900                ADD CAUX
901                STA PTOB
902                JSR ACTIVAD
903                INC CAUX
904                RTS
905
906    PROCD2      LSRA
907                LSRA
908                AND #$3C
909                ADD CAUX
910                STA PTOB
911                JSR ACTIVAD
912                INC CAUX
913                RTS
```

Como puede verse ambos procedimientos hacen uso de la subrutina ACTIVAD la cual se encarga de activar el buffer de displays haciendo un retardo para una correcta grabación de los datos. Véase la **Fig. 2.42.**

Fig. 2.42.- Subrutinas para preparar dato a visualizar.

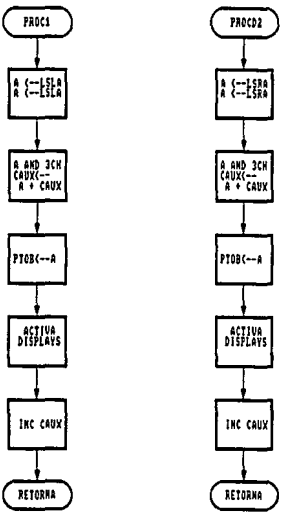
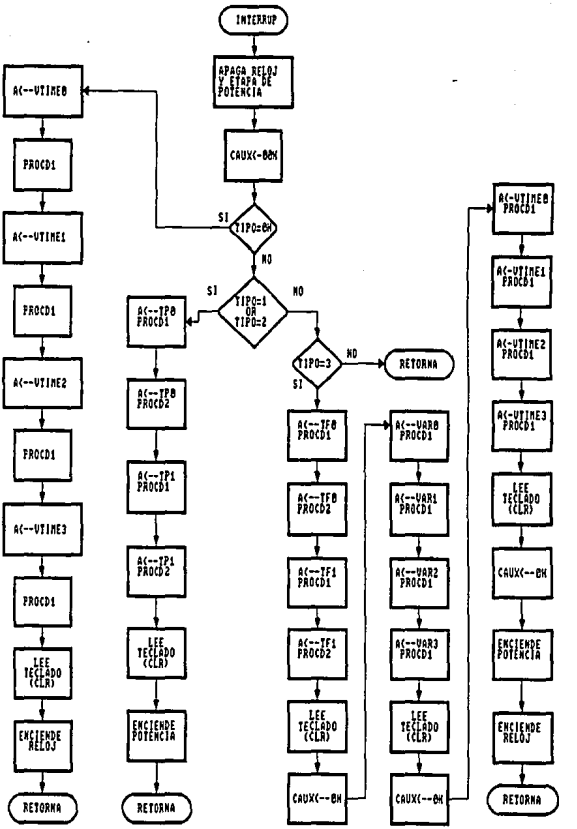


Fig. 2.43.- Subrutina para manejo de interrupción.



Estos procesos son similares como puede verse, a los procedimientos utilizados para desplegar datos en los displays.

2.8.30. SUBRUTINA PARA MANEJO DE INTERRUPCION (INTERRUP).

Cuando se genera un pulso bajo a través del pin INT del MC68705P3, se produce una interrupción externa. Esta interrupción hace que la ejecución normal del programa se interrumpa, guardando el estado actual del PC en la pila, y se ejecute la rutina correspondiente cuya dirección se encuentra almacenada en el vector de interrupción ubicado en las direcciones $7FA y $7FB. En nuestro caso, este tipo de interrupción, provoca la visualización de los parámetros iniciales establecidos de acuerdo al tipo de control escogido.

En Termómetro Temporizado sólo se visualizará el tiempo, en Control Proporcional y en Control ON/OFF, la temperatura final (SET POINT), y en Control Proporcional por Pasos Automáticos Temporizados, la temperatura final, el incremento de temperatura y el tiempo. Esto se hace con el fin de que el usuario tenga la posibilidad de revisar si los datos introducidos fueron correctos o bien como simple recordatorio. En caso de error en los datos, el usuario tiene la posibilidad de abortar el proceso.

De acuerdo a lo anterior, lo único que tenemos que hacer es llamar a aquella serie de datos almacenados en memoria, y exhibirlos en los displays. El diagrama de flujo correspondiente se muestra en la **Fig. 2.43**.

Las rutinas RELOJ_OFF y POTENCIA_OFF así como RELOJ_ON Y POTENCIA_ON son rutinas simples que se encargan de apagar o encender el reloj y la etapa de potencia respectivamente.

El listado en ensamblador es el que se muestra en las siguientes líneas:

```
916     INTERRUP    JSR RELOJ_OFF
917                 JSR POTENCIA_OFF
918                 LDA #$02
919                 STA PARAM
920                 LDA #$00
921                 STA CAUX
922                 LDA TIPO
923                 ADD #$01
```

```
924                 JSR ACTIVAE
925                 LDA TIPO
926                 CMP #$01
927                 BEQ PRONOF
928                 CMP #$02
929                 BEQ PRONOF
930                 CMP #$03
931                 BEQ PRG
932                 CMP #$00
933                 BEQ DESTIM
934                 RTI
935           PRONOF LDA #$05
936                 JSR ACTIVAE
937                 LDA TP0
938                 JSR PROCD1
939                 LDA TP0
940                 JSR PROCD2
941                 LDA TP1
942                 JSR PROCD1
943                 LDA TP1
944                 JSR PROCD2
945                 JSR LEE_KBD
946                 JSR POTENCIA_ON
947                 JMP SALIDAI
948             PRG LDA #$05
949                 JSR ACTIVAE
950                 LDA TF0
951                 JSR PROCD1
952                 LDA TF0
953                 JSR PROCD2
954                 LDA TF1
955                 JSR PROCD1
956                 LDA TF1
957                 JSR PROCD2
958                 JSR LEE_KBD
959                 LDA #$06
960                 JSR ACTIVAE
961                 LDA #$00
962                 STA CAUX
963                 LDA VAR0
964                 JSR PROCD1
965                 LDA VAR1
966                 JSR PROCD1
967                 LDA VAR2
968                 JSR PROCD1
969                 LDA VAR3
970                 JSR PROCD1
971                 JSR LEE_KBD
972           DESTIM LDA #$07
973                 JSR ACTIVAE
974                 LDA #$00
```

```
975                 STA CAUX
976                 LDA VTIME0
977                 JSR PROCD1
978                 LDA VTIME1
979                 JSR PROCD1
980                 LDA VTIME2
981                 JSR PROCD1
982                 LDA VTIME3
983                 JSR PROCD1
984                 JSR LEE_KBD
985                 LDX TIPO
986                 CPX #$00
987                 BEQ NOPOTEN
988                 JSR POTENCIA_ON
989         NOPOTEN JSR RELOJ_ON
990         SALIDAI LDA TIPO
991                 ADD #$01
992                 JSR ACTIVAE
993                 LDA #$00
994                 STA PARAM
995          FININ  RTI
```

CAPITULO 3
EL SISTEMA INTEGRADO: PRUEBAS Y RESULTADOS

CAPITULO 3. EL SISTEMA INTEGRADO: PRUEBAS Y RESULTADOS

En los capítulos anteriores se llevó a cabo el análisis del "Controlador Lineal Programable de Temperatura" separándolo en dos apartados: el Hardware y el Software. En el capítulo dedicado al Hardware se dio un panorama global de todo el circuito y se dedicó una sección especial para el estudio de cada uno de los módulos que lo integran: el microcontrolador MC68705P3, el teclado, la pantalla de visualización de datos, los leds indicadores de estado, la etapa de potencia, la alarma y la etapa concerniente a la lectura de temperatura. En el capítulo relativo al software, se llevó en si la elaboración de las rutinas necesarias para el control de cada uno de los módulos tratados en la sección del hardware, así como los programas de integración de todos los módulos para la ejecución de las funciones Termómetro Temporizado, Control ON-OFF, Control Proporcional y Control Proporcional Automático por Pasos Temporizados.

La integración del Hardware y el Software implica finalmente hechar a andar todo el sistema, estableciendo las condiciones necesarias para cada uno de los controles y realizando las pruebas pertinentes. El presente capítulo trata cada uno de los controles por separado estableciendo las pruebas realizadas y reportando los resultados obtenidos.

3.1. FUNCION TERMOMETRO TEMPORIZADO

La función termómetro temporizado no representa ningún tipo de control sobre la temperatura, simplemente trabaja como un termómetro normal reportando la temperatura leída, cada determinado lapso de tiempo.

Para efectos de prueba se cuenta con una parrilla de calentamiento y un recipiente con agua, aproximadamente 4 litros, a temperatura ambiente (18 °C) y presión atmosférica igual a la del aire, considerando que el recipiente se encuentra destapado.

La alimentación de la parrilla se ha conectado directamente a la salida de la etapa de potencia y el termocople se encuentra, junto con un termómetro de mercurio, en contacto con la sustancia.

En la espera del controlador por algún tipo de función, se teclea F1 para elegir Termómetro Temporizado. Una vez establecida la función a ejecutar el sistema pide el incremento de tiempo en base al cual se dará la lectura de la temperatura por parte del controlador, iniciándose el proceso una vez capturado.

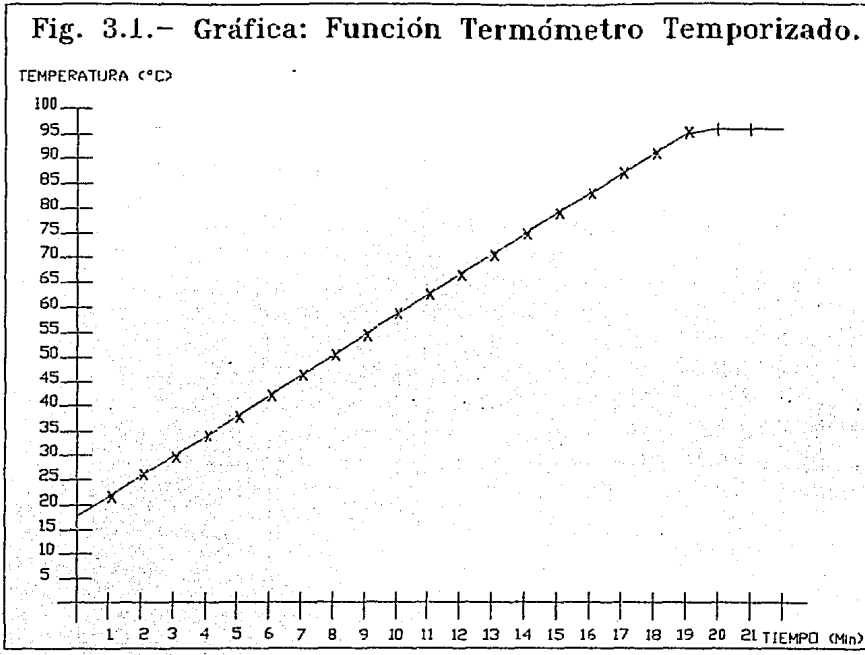
El tiempo se ha fijado igual a 60 segundos, es decir, 1 minuto. El sistema reportará la temperatura del agua cada minuto la cual será comparada con la temperatura marcada por el termómetro.

La tabla siguiente muestra una serie de lecturas realizadas por el controlador y el termómetro:

Tiempo (minutos)	Temperatura Termómetro Temporizado (° C)	Temperatura Termómetro de prueba (° C)
0	18	18
1	22	22.4
2	26	25.8
3	30	30.1
4	34	33.9
5	38	38.1
6	42	41.9
7	46	46.2
8	50	50.2
9	54	54.4
10	59	58.6
11	63	62.8
12	67	66.7
13	71	70.9
14	75	75.1
15	79	79.1
16	83	83.1
17	87	86.8
18	91	90.5
19	94	94.2
20	96	96.1
21	96	96.2
22	96	96

En la **Fig. 3.1.** se muestra la gráfica correspondiente a la tabla anterior.

Fig. 3.1.– Gráfica: Función Termómetro Temporizado.



Como puede observarse, la temperatura leída por el controlador es similar a la obtenida a través del termómetro, presentando en algunos casos ligeras variaciones debidas a los posibles errores en cuanto a los decimales no considerados en el controlador y a errores de visión en las lecturas hechas en el termómetro de prueba.

Al presionar la tecla clear (C) se puede apreciar el transcurso del tiempo, dándose una nueva lectura de temperatura una vez que éste se ha decrementado hasta cero.

3.2. CONTROL ON/OFF.

Recordando, tenemos que el control ON/OFF requiere la captura del SET POINT por alcanzar, antes de iniciar el proceso. En base a este dato, determina si efectúa o no el calentamiento del sistema. Es decir, si la temperatura actual del sistema controlado es mayor o igual al SET POINT, entonces no se lleva a cabo el proceso.

Una vez capturado y validado el dato de temperatura, se inicia el proceso de alcance de temperatura. En el momento de ser alcanzado la etapa de potencia es apagada y se regresa al punto de petición de Tipo de Control.

La prueba aplicada consistió en calentar un volumen de agua para tres distintos SET POINT's, mientras era monitoreada alternativamente por otro termómetro.

Los tres SET POINTS establecidos fueron:

a) 10 °C, que corresponde a un valor inferior al ambiente,
b) 75 °C, un dato superior al ambiente y
c) 100 °C, un valor que supera por 4 °C el punto de ebullición a esta altitud.

Las condiciones iniciales del líquido antes de cada prueba fueron:

Temperatura Ambiente	= 18 °C,
Temp. Inicial del Líquido	= 18 °C,
Presión Aplicada	= 1 atm,
Volumen Inicial	= 4 Lts.

Los resultados obtenidos se muestran a continuación, para muestreos nominales de 60 segs (1 minuto):

a) SET POINT = 10 °C.

Temperatura Control ON/OFF (°C)	Temperatura Termómetro de Prueba (°C)
18	18.0

A partir del dato anterior, observamos que el controlador no procedió a efectuar el alcance de temperatura, ya que el SET POINT es menor a la temperatura corriente del sistema.

b) SET POINT = 75 °C

Temperatura Control ON/OFF (°C)	Temperatura Termómetro de Prueba (°C)
18	18.0
22	22.1
26	26.0
30	30.1
34	33.9
38	38.0
42	42.0
46	45.9
50	50.2
54	54.2
58	58.0
62	61.8
66	65.9
70	70.0
74	74.1
75	75.3

El valor un poco más alto al alcanzar la temperatura indicada se debió a la inercia cinética del líquido, puesto que se respetó el tiempo de monitoreo.

c) SET POINT = 100 °C.

En este caso usamos el sistema tal y como quedó después de alcanzar los 75 °C, obteniendo los siguientes datos:

Temperatura Control ON/OFF (°C)	Temperatura Termómetro de Prueba (°C)
75	75.1
79	79.1
83	83.0
87	87.0
91	90.9
95	94.9
96	96.1
96	96.6
96	96.6
96	96.6
<Se oprimió CLEAR>	

En este caso, el controlador mantuvo encendida la etapa de potencia debido a que no alcanzó la temperatura indicada y fue necesario efectuar la terminación manualmente. Lo anterior se debe al hecho que el agua a la altitud de la ciudad de México entra en ebullición a los 96.6 °C.

Los resultados gráficos se muestran en la **Fig. 3.2. y 3.3.**

Fig. 3.2.- Caso b) S.P. = 75

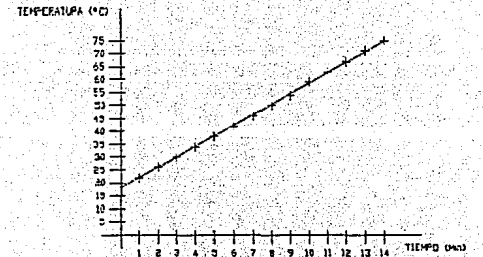
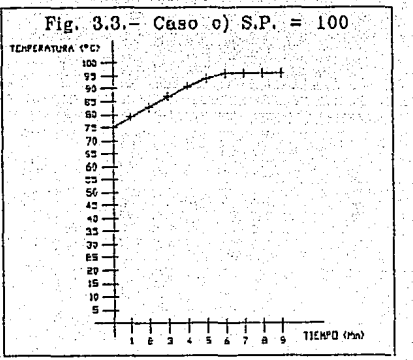


Fig. 3.3.- Caso c) S.P. = 100



3.3. CONTROL PROPORCIONAL.

El Control Proporcional sigue un proceso similar al Control ON-OFF, se establece una temperatura o SET POINT a alcanzar, si esta temperatura se encuentra por arriba de la temperatura inicial el proceso no se ejecuta, en caso contrario el proceso normal continua. A diferencia del Control ON-OFF, una vez alcanzada la temperatura, ésta se mantiene por un tiempo indeterminado hasta que el usuario decide abortar el proceso. Alcanzada la temperatura establecida, la etapa de potencia conmuta a encendido-apagado cuando la temperatura cae 5° C abajo del SET POINT o bien, cuando sube 5° C por arriba del mismo respectivamente.

Para arrancar el Controlador en modo de Control Proporcional presionamos la tecla F3, para lo cual el controlador responde pidiendo la temperatura especifica a alcanzar.

La temperatura máxima tolerada por el controlador se ha establecido igual a 1020 °C, por lo que capturar una temperatura superior a ésta resulta en un error, de tal forma que se tendrá que capturar un valor correcto.

Para efectos de prueba se tienen tres recipientes con agua a diferente temperatura: 18, 30 y 65 °C respectivamente a la presión atmosférica del aire. Al igual que en los procesos anteriores, se cuenta con un termómetro de prueba para corroborar los resultados. Para cada uno de estos recipientes se ejecutará un tipo de Control Proporcional estableciendo una temperatura diferente en cada uno de los casos, presentándose tres tipos de situaciones: cuando la temperatura a alcanzar está por abajo de la temperatura inicial del agua, cuando está por arriba de la temperatura del agua y cuando es igual a la temperatura actual del agua.

Seleccionamos un tipo de Control Proporcional e introducimos como temperatura tope 50 °C. Los resultados obtenidos fueron los siguientes (**Fig. 3.4.**):

Temperatura a alcanzar (SET POINT) : 50 °C.
Temperatura inicial: 18 °C.

Temperatura Control Proporcional (°C)	Temperatura Termómetro de prueba (°C)
18	18
22	22.3
26	25.7
30	29.9
34	34.3
38	37.6
42	42.2
46	45.8
50	50.4
50	50.3
50	49.6

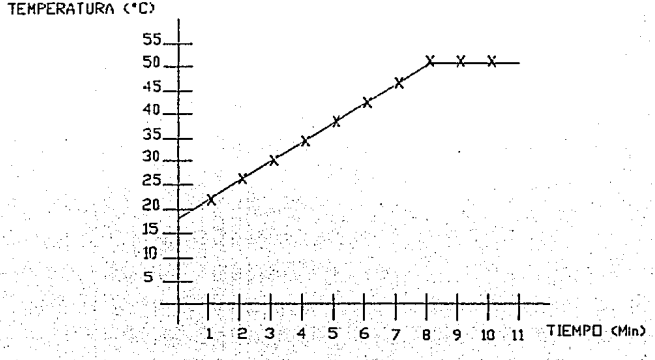
Al presionar la tecla clear el proceso se suspende y el controlador se encuentra en espera de un nuevo tipo de función.

Para el segundo recipiente, se seleccionó el Control Proporcional estableciéndose 24 °C como parámetro o temperatura a alcanzar. El proceso no se lleva a cabo debido a que la temperatura a alcanzar esta por debajo de la temperatura inicial de la sustancia.

Finalmente, en la sustancia cuya temperatura inicial es de 65 °C, se ha establecido el SET POINT igual a 65 °C. La temperatura se mantiene en un rango de +/- 5 °C la temperatura inicial del líquido, obteniéndose los siguientes resultados:

Temperatura Control Proporcional (° C)	Temperatura Termómetro de prueba (° C)
65	65
63	63.2
62	61.3
64	64.1
66	65.6
67	66.8
65	65.4

Fig. 3.4.- Gráfica: Control Proporcional.



Como puede observarse, la etapa de potencia se encuentra en constante conmutación encendido-apagado, tratando de mantener la temperatura en el rango establecido.

3.4. CONTROL PROPORCIONAL POR PASOS AUTOMATICOS TEMPORIZADOS.

Finalmente, para este tipo de control se efectuó solamente una prueba. Antes de iniciar debemos considerar las características de este control:

1) Presenta los mismos aspectos de alcance y mantenimiento de temperatura del control Proporcional,
2) Puede efectuar escalonamientos de temperatura y
3) Puede efectuar mantenimientos por tiempos definidos.

Por lo tanto, requiere de tres parámetros capturables importantes que son:

a) SET POINT,
b) Variación de Temperatura y
c) Tiempo de Mantenimiento

Por lo tanto, la prueba contempló los tres aspectos con los siguientes valores: SET POINT = 50 °C, Var. de Temperatura = 10 °C y Tiempo de Mantenimiento = 3 minutos (180 segundos).

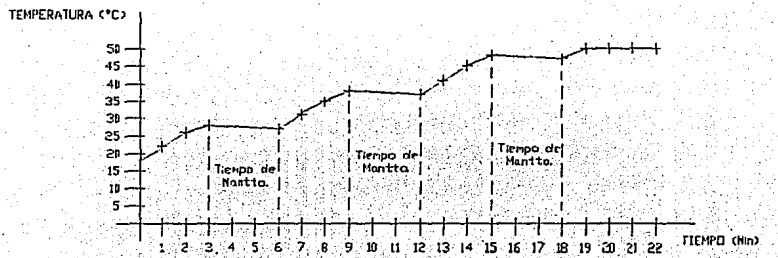
Bajo esta condiciones los resultados fueron los siguientes:

Temperatura Control PAT (°C)	Temperatura Termómetro de Prueba (°C)	Tiempo de Espera (min.)	Tiempo de Cronómetro (min.)
18	18.0		
22	22.1		
26	26.1		
28	28.0		
		3:00	3:05
27	27.3		
31	30.1		
35	35.0		
38	38.1		
		3:00	3:02
37	37.5		
41	41.2		

```
45                45.2
48                48.0
                                    3:00          3:03
47                47.8
50                50.3
50                50.1
50                49.9
50                49.7
 <Se oprimió CLEAR>
```

De acuerdo con los datos anteriores se tiene la gráfica mostrada en la **Fig. 3.5.**, la cual nos muestra que el desempeño del control es bastante estable.

Fig. 3.5.- Gráfica: Control PAT.



3.5. LECTURA DE PARAMETROS INICIALES

Una vez arrancada alguna de las funciones soportadas por el controlador, si deseamos conocer o bien corroborar los datos introducidos, podemos hacerlo presionando el botón de interrupción.

Para la función Termómetro Temporizado, el sistema despliega el incremento del tiempo, para Control ON-OFF, la temperatura final o SET POINT, para Control Proporcional la temperatura a alcanzar o SET POINT y para Control Proporcional Automático por Pasos Temporizados, la temperatura final, el incremento de temperatura y el incremento del tiempo. Si existe algún error en los datos, el proceso puede abortarse mediante la tecla clear, la cual provoca que la función que se está ejecutando actualmente por el controlador se suspenda. El controlador, en este momento, está listo para ejecutar otra función y aceptar nuevos parámetros en base a los cuales llevará a cabo el control de la temperatura.

En cada una de las pruebas realizadas en los apartados anteriores se checaron los datos introducidos por presionar el botón de interrupción, corroborando la veracidad de los mismos.

CONCLUSIONES

El presente trabajo, aún con la sencillez que presenta, sugiere una alternativa muy poderosa para el desarrollo de sistemas automáticos de control mediante el empleo del microcontrolador MC68705P3, y no sólo eso, las aplicaciones se pueden extender al empleo de otras familias de microcontroladores, similares en funcionamiento y con ventajas menores o superiores al empleado por el Controlador Lineal Programable de Temperatura, cuya elección está en función del grado de complejidad del diseño, del alcance del mismo y de que tan económico resulte.

El microcontrolador constituye el corazón del Controlador Lineal Programable de Temperatura ya que a través de un solo programa, residente en memoria, se lleva a cabo el control de cada una de las tareas requeridas para el control de temperatura, todo lo cual implica un menor espacio en componentes externas haciendo del Controlador un dispositivo fácil de manipular.

La grabación del programa en la EPROM resulta sencilla gracias al programa de fábrica elaborado para tal efecto, en todo caso, el único inconveniente es la construcción del circuito especial para grabación de la EPROM que se tiene que alambrar y cuyo diagrama lo proporciona el propio fabricante. No obstante, la construcción del mismo constituye una avance para posteriores aplicaciones.

Gracias a las ventajas inherentes proporcionadas por el microcontrolador MC68705P3, se concentró una mayor atención en los objetivos principales del proyecto: proporcionar al usuario una serie de tareas que le permitan llevar a cabo el control de la temperatura de una determinada sustancia, con el fin de observar el comportamiento de la misma de acuerdo a un fin específico. Para esto se hicieron una serie de pruebas simulando el proceso, pruebas que se mencionan en el capítulo titulado "El Sistema Integrado: Pruebas y Resultados" habiéndose obtenido, después de pequeños ajustes, los resultados esperados.

Debido a que el diseño se llevó a cabo en forma modular, se probó cada uno de los módulos por separado haciéndose mucho más sencilla la localización de fallas, siguiendo el flujo de las señales y a través del monitoreo de las entradas y las salidas.

En lo que concierne al programa elaborado, se hizo uso del simulador para los microcontroladores de Motorola serie MC68705 en sus diferentes modalidades P3, R3 y U3. El hecho de poder simular un programa implica que se puedan detectar fallas severas en la programación antes de que se produzcan en el entorno físico, fallas que pudieran provocar el daño de algún componente físico con la consecuente perdida de tiempo consumido en la detección del problema.

El presente diseño representa una propuesta viable para resolver el problema de control de temperatura y se encuentra abierto a mejoras. Para lograr esto se puede pensar en el uso del MC68705R3 el cual incluye un convertidor analógico digital así como mayor capacidad de memoria. Podría incluirse un teclado con un mayor número de funciones, una pantalla de cristal líquido y un banco de memoria de mayor capacidad, lo que daría la posibilidad de un Controlador de Temperatura Programable por el usuario a un nivel más avanzado y con un menor número de limitantes. Se puede incluir, por otro lado, una interface gráfica que permita al usuario visualizar el comportamiento de la sustancia. De antemano, todas estas mejoras implican un gasto mayor, gasto que por el momento no fue posible.

Aunque la situación actual del país no nos permite resolver los problemas cotidianos con la mejor tecnología, es importante conocer los alcances logrados diariamente y tomar toda esa serie de conocimientos para obtener cosas que nos puedan ser útiles, no nos quedemos con lo viejo, exploremos siempre que nos sea posible lo nuevo. La tecnología de los microcontroladores no es nueva y esta al alcance de todos los universitarios abriendo la puerta hacia otro tipo de tecnología más avanzada (como los PLCps) de la cual muchas veces no se tiene conocimiento dentro del aula.

BIBLIOGRAFIA

CANALES, Roberto Ruíz, Análisis de Sistemas Dinámicos y Control Automático, 1a. ed., México, Ed. Trillas, 1988.

COUGHLIN, Roberto F., Frederik Driscoll, Circuitos Lineales y Amplificadores Operacionales, 2a. ed., México, Ed. Prentice Hall, 1987.

CREUS, Antonio Sole, Instrumentación Industrial, 4a. ed., México, Ed. Marcombo, 1993.

DORF, Richard C., Sistemas Modernos de Control, Teoría y Práctica, 1a. ed., México, Ed. McGraw Hill, 1987.

GENE, Jim F. Franklin y J. David Powell Workman, 1a. ed., Ed. Prentice Hall, American Books.

HAYT, William H., Análisis de Circuitos en Ingeniería, 1a. ed., México, Ed. McGraw Hill, 1986.

INTEL, 8-Bit Embedded Controllers Handbook, Intel, U.S., 1990.

KUO, Benjamin C., Sistemas Autómaticos de Control, 1a. ed., México, Ed. CECSA, 1989.

KUTZ, Myer, Temperature Control, 1a. ed., Estados Unidos, Ed. Publishing Company Huntington, New York, 1975

MALVINO, Albert Paul, Principios de Electrónica, 2a. ed, México, Ed. McGRAW-HILL, 1990.

MCKELVEY, John P. y Howard Grotch, Física para Ciencias e Ingeniería, 1a. ed., México, Ed Harla, 1987.

MEJIA, Aurelio Meza, Diccionario Técnico Actualizado, 1a. ed., Colombia, Ed. Marín Vieco, 1990.

MORGAN, Cristopher L., Introducción al Microprocesador 8086/8088, 1a. ed., México, Ed. Mc Graw Hill, 1987.

MORRIS, Mano M., Arquitectura de Computadores, 1a. ed., México, Ed. Prentice Hall Hispanoamericana, 1983.

MOTOROLA, Microcontroladores, U.S., 1990.

MOTOROLA, Fast and LS TTL Data Book, 4a. ed., U.S., Motorola Inc, 1989.

NATIONAL SEMICONDUCTOR, Lineal Data Book National Semiconductor, National Semiconductor Co., 1988.

NORMAN, E. Lurch, Fundamentos de Electrónica, 2a. ed., México, Ed. CECSA, 1989.

PALLAS, Areny Ramón, Transductores y Acondicionadores de Señal, 1a. ed., México, Ed. Marcombo, pp. 1-15.

TOCCI, Ronald J., Sistemas Digitales, 3a. ed., México, Ed. Prentice Hall, 1987.

WILLIAMS, B. Arthur, Microprocesadores, Dispositivos Periféricos, Optoelectrónicos y de Interfaz, 1a. ed., México, Ed. McGRAW-HILL/INTERAMERICANA DE MEXICO, 1990.

APENDICE A

```
1   * PROGRAMA GENERAL PARA CONTROL DE TEMPERATURA *

2   * DECLARACION DE VARIABLES *
3
4   PTOA    EQU $000    * PUERTO A.
5   PTOB    EQU $001    * PUERTO B.
6   PTOC    EQU $002    * PUERTO C.
7   DDRA    EQU $004    * DDRA.
8   DDRB    EQU $005    * DDRB.
9   DDRC    EQU $006    * DDRC.
10  TDR     EQU $008    * REGISTRO DE DATOS DEL TEMPORIZADOR.
11  TCR     EQU $009    * REGISTRO DE CONTROL DEL TEMPORIZADOR.
12  T       EQU $010
13  PARAM   EQU $011
14  DMUX    EQU $013
15  CARRY   EQU $014
16  SUMA    EQU $015
17  TIPO    EQU $016    * TIPO DE FUNCION ELEGIDA POR EL USUARIO.
18  TESP    EQU $017    * TIEMPO DE ESPERA.
19  TF1     EQU $018    * TEMP. FINAL BYTE ALTO (CONTROL PAT).
20  TF0     EQU $019    * TEMP. FINAL BYTE BAJO.
21  TP1     EQU $01A    * TEMP. PARCIAL BYTE ALTO.
22  TP0     EQU $01B    * TEMP. PARCIAL BYTE BAJO.
23
24  REGTEM  EQU $01F
25  POS     EQU $020
26  FLAG    EQU $021    * BANDERA DE FIN DE CONVERSION.
27  COCIENT EQU $022    * $023 - $026 GUARDA BCD SIN COMPRIMIR.
28  CONCOM1 EQU $027    * REGISTRO RAM->ALMACENA NUMERO
DECIMAL.
29  CONCOM2 EQU $028    * REGISTRO RAM->ALMACENA NUM.
COMPRIMIDO.
30  CONTAD  EQU $029    * CONTADOR DE CONTROL DE COMPRIME.
31  VTIME0  EQU $02A
32  VTIME1  EQU $02B
33  VTIME2  EQU $02C
34  VTIME3  EQU $02D
35  VAR0    EQU $03A
36  VAR1    EQU $03B
37  VAR2    EQU $03C
38  VAR3    EQU $03D
39  PAR1    EQU $03E
40  CTA     EQU $03F
41  STORK   EQU $040
42  NUMT    EQU $046
43  TIEMPO0 EQU $04A
44  TIEMPO1 EQU $04B
45  TIEMPO2 EQU $04C
46  TIEMPO3 EQU $04D
47  CAUX    EQU $04F
48  CONDIF  EQU $051
49  R1      EQU $065
50  R0      EQU $066
51
```

```
52   * VARIABLES DE HABILITACION (CONSTANTES) *
53
54   CLEAR    EQU $01E      * PROCESO ABORTADO, CLEAR = 4.
55   START    EQU $030      * START = 0, INICIA CONVERSION EN EL ADC.
56   MUX_A    EQU $031      * MUX_A = 1, SELEC. DE ENTRADA AL ADC.
57   REG_D    EQU $032      * REG_D = 2, LATCH DEL DISPLAYS.
58   KBD      EQU $033      * KBD   = 3, TECLADO.
59   EOC      EQU $034      * EOC   = 4, SENSA FIN DE CONVERSION.
60   LIBRE    EQU $035      * LIBRE = 5.
61   ESTADO   EQU $036      * ESTADO= 6, ARREGLO DE LEDS.
62   ALARM    EQU $037      * ALARM = 7, HABILITACION DE LA ALARMA.
63   POT      EQU $038      * POT   = 8, HAB. HORNO (ETAPA DE POTENCIA).
64   OEAD     EQU $039      * OEAD  = 9, SIN USO FIJO.
65
66   * SUBRUTINA PRINCIPAL *
67
68             ORG $80
69   PRINCIPAL CLR DDRA      * PUERTO A NORMALMENTE ENTRADA
70             LDX #$FF
71             STX DDRB      * PUERTO B NORMALMENTE SALIDA
72             STX DDRC      * PUERTO C NORMALMENTE SALIDA
73             LDX #$010
74             LDA #$00      * BLANQUEO DE LA RAM
75   LOOP1     STA X         * DESDE $010 HASTA
76             INCX          * LA DIRECCION $06F
77             CPX #$6F
78             BNE LOOP1
79             LDX #$030
80   OTRA      STA X         * ASIGNACION DE VALORES
81             INCX          * DE CONTROL A CADA PALABRA
82             INCA          * INDICADA EN LA DECLARACION
83             CMP #$0A
84             BNE OTRA
85             LDA #$04
86             STA CLEAR
87   MENU      JSR RELOJ_OFF
88             JSR POTENCIA_OFF
89             JSR CLDIS
90             LDA #$00
91             STA CTA
92             JSR ACTIVAE
93             LDX KBD
94             STX PTOC
95   TEC1      BRCLR 4,PTOA,TEC1  *ESPERA EL TIPO DE CONTROL
96   TEC2      BRSET 4,PTOA,TEC2  * ELIMINA REBOTE DE TECLA
97             LDA PTOA
98             LDX #$0F
99             STX PTOC
```

```
100             AND #$0F
101             SUB #$0C
102             STA TIPO
103             JSR DECIDE
104             JSR CONTROLES       * EJECUTA EL TIPO DE CONTROL ELEGIDO
105             BRA MENU            * VUELVE A ITERAR
106
107   * SUBRUTINA PARA LECTURA DE PARAMETROS (DECIDE) *
108
109   DECIDE    LDA TIPO
110             CMP #$01
111             BEQ DECI0
112             CMP #$02
113             BEQ DECI0
114             CMP #$03
115             BEQ DECI1
116             CMP #$00
117             BEQ NPJM2
118             RSP
119             JMP MENU
120   DECI0     LDA #$05            * PETICION DE SET POINT
121             JSR ACTIVAE
122             JSR READ_KEY
123             LDA $063
124             STA TP1
125             LDA $064
126             STA TP0
127             LDA TP1
128             CMP #$10
129             BHI PJMS
130             BNE NPJMS
131             LDA TP0
132             CMP #$20
133             BHI PJMS
134   NPJMS     RTS
135   PJMS      JSR SEQUIV
136             JMP DECI0
137
138   DECI1     LDA #$05            * PETICION DE SET POINT
139             JSR ACTIVAE
140             JSR READ_KEY
141             LDA $063
142             STA TF1
143             LDA $064
144             STA TF0
145             LDA TF1
146             CMP #$10
147             BHI PJM1
148             BNE NPJM1
```

```
149             LDA TF0
150             CMP #$20
151             BHI PJM1
152             JMP NPJM1
153     PJM1    JSR SEQUIV
154             JMP DEC11
155     NPJM1   LDA #$06        * PETICION VARIACION DE TEMPERATURA
156             JSR ACTIVAE
157             JSR READ_KEY
158             LDA $041
159             STA VAR3
160             LDA $042
161             STA VAR2
162             LDA $043
163             STA VAR1
164             LDA $044
165             STA VAR0
166             LDA $63
167             CMP #$03
168             BHI PJM2
169             BNE NPJM2
170             LDA $64
171             CMP #$00
172             BHI PJM2
173             JMP NPJM2
174     PJM2    JSR SEQUIV
175             JMP NPJM1
176     NPJM2   LDA #$07        * PETICION PARA VARIACION DE TIEMPO
177             JSR ACTIVAE
178             JSR READ_KEY
179             LDA $041
180             STA VTIME3
181             LDA $042
182             STA VTIME2
183             LDA $043
184             STA VTIME1
185             LDA $044
186             STA VTIME0
187             RTS
```

188 ***SUBRUTINA PARA ACTIVAR LEDS DE ESTADO (ACTIVAE)** *

```
189     ACTIVAE STA PTOB
190             LDA ESTADO
191             STA PTOC
192             LDA #$0F
193     RETARDO1 DECA
194             BNE RETARDO1
195             LDA #$0F
196             STA PTOC
197             RTS
```

```
198   * SUBRUTINA PARA INDICACION DE ERROR Y PAUSA (SEQUIV) *

199   SEQUIV  LDA #$01
200           STA PTOB
201           LDA ALARM
202           STA PTOC
203           LDA #$0F
204   REPEAT  DECA
205           BNE REPEAT
206           LDA #$09
207           JSR ACTIVAE
208           LDA KBD
209           STA PTOC
210   SEQ1    BRCLR 4,PTOA,SEQ1
211   SEQ2    BRSET 4,PTOA,SEQ2
212           LDA #$0F
213           STA PTOC
214           RTS
215
216   * SUBRUTINA PARA LECTURA DE DATOS (READ_KEY) *
217
218   READ_KEY JSR CLDIS
219           LDA #$41          * ALMACENA 41h COMO LA PRIMERA
220           STA STORK         * DIRECCION EN LA CUAL SE
221           LDA #$00          * ALMACENARAN LOS DATOS
222           STA NUMT          * INICIALIZA NUMT A CERO
223           LDA KBD           * SE HABILITA LA SALIDA 4h DEL DECO
224           STA PTOC          * LA CUAL CORRESPONDE A LA
225           LDA #$04          * HABILITACION DEL TECLADO
226           LDX #$52
227   PASO1   JSR SIGMA         * C1, C2 Y C3 ALMACENAN LAS TECLAS
228           CMP #$0B          * REALES DE ACUERDO CON EL TECLADO
229           BMI PASO1         * EMPLEADO
230           LDA #$02
231   PAS2    JSR SIGMA
232           CMP #$0C
233           BMI PAS2
234           LDA #$03
235   PAS3    JSR SIGMA
236           CMP #$0A
237           BMI PAS3
238   RE_READ BRCLR 4,PTOA,RE_READ  * SENSA SI SE HA ACTIVADO TECLA.
239   CHECK_RE BRSET 4,PTOA,CHECK_RE * EVITA REBOTE DE TECLA.
240           LDA PTOA          * SE CARGA TECLA EN EL PTO A.
241           AND #$0F          * ELIMINA BIT DE DATO VALIDO.
242           CMP #$02          * LAS TECLAS 0 Y 1 SON TECLAS REALES.
243           BMI NEWKEY
244           CMP #$0C          * LAS TECLAS F1 HASTA F5 TAMBIEN SON
245           BPL RE_READ       * TECLAS REALES.
246           ADD #$50          * SE CARGA LA DIRECCION DE LA TECLA
247           TAX               * REAL Y SE TRANSFIERE AL REGISTRO X.
```

```
248                 LDA X              * CARGA EL VALOR REAL DE LA TECLA.
249     NEWKEY      LDX STORK          * ALMACENA NUEVA TECLA (TECLA REAL).
250                 STA X
251                 CMP #$0A           * DETECTA SI LA TECLA PRESIONADA ES
252                 BPL FUNKEY         * UNA TECLA DE FUNCION.
253                 LDA NUMT           * SE DETECTA DESBORDAMIENTO DE
DATOS
254                 CMP #$04           * Y SE ESPERA CLEAR O ENTER.
255                 BEQ RE_READ
256                 JSR DISPLAY
257                 INC NUMT           * AVANCE DE CURSOR.
258                 INC STORK
259                 BRA RE_READ
260     FUNKEY      LDX NUMT           * NO SE PERMITE COMO PRIMER DATO UNA
261                 CPX #$00           * TECLA DE FUNCION.
262                 BEQ RE_READ
263                 CMP #$0B           * SI LA TECLA OPRIMIDA FUE ENTER SALE
264                 BEQ UNKEY          * DEL CICLO
265                 DEC NUMT
266                 DEC STORK
267                 JSR RIGHT_DIS
268                 BRA RE_READ
269     UNKEY       DEC NUMT
270                 LDA NUMT           * ALMACENA CEROS EN LAS POSICIONES
271     COMPAR      CMP #$03           * RESTANTES
272                 BEQ FINKEY
273                 LDA #$03
274                 SUB NUMT
275                 STA CONCOM1
276                 LDA #$00
277                 STA CONTAD
278     INIC        STA CONCOM2
279                 LDA #$44
280                 STA STORK
281     RECDER      LDX STORK
282                 DECX
283                 LDA X
284                 LDX STORK
285                 STA X
286                 DEC STORK
287                 INC CONCOM2
288                 LDX CONCOM2
289                 CPX #$03
290                 BNE RECDER
291                 LDA #$00
292                 LDX STORK
293                 STA X
294                 INC CONTAD
295                 LDX CONTAD
296                 CPX CONCOM1
297                 BNE INIC
```

```
298  FINKEY    LDX #$41
299            STX CONCOM1
300            LDX #$63
301            STX CONCOM2
302            JSR RESUME
303            RTS
304
305  * SUBRUTINA PARA DESPLEGADO DE DATOS (DISPLAY) *
306
307  DISPLAY   LDA #$00
308            STA CONDIF
309  NUMDIS    LDA STORK
310            SUB CONDIF
311            TAX
312            LDA X
313            LSLA
314            LSLA
315            ADD CONDIF
316            STA PTOB
317            JSR ACTIVAD
318            INC CONDIF
319            LDA NUMT
320            INCA
321            CMP CONDIF
322            BNE NUMDIS
323            LDA KBD
324            STA PTOC
325            RTS
326  * SUBRUTINA PARA ACTIVACION DE DISPLAYS (ACTIVAD) *
327  ACTIVAD   LDX REG_D
328            STX PTOC
329            LDX #$0F
330  WAIT2     DECX
331            BNE WAIT2
332            LDX #$0F
333            STX PTOC
334            RTS
335
336  * SUBRUTINA PARA BORRAR DATO DE LOS DISPLAYS (RIGHT_DIS) *
337
338  RIGHT_DIS LDA #$00
339            STA CONCOM1
340            LDA #$01
341            STA CONTAD
342            LDX NUMT
343            CPX #$00
344            BEQ ZEROS
```

```
345  REWIND   LDA STORK
346           SUB CONTAD
347           TAX
348           LDA X
349           LSLA
350           LSLA
351           ADD CONCOM1
352           STA PTOB
353           JSR ACTIVAD
354           INC CONCOM1
355           LDA NUMT
356           INC CONTAD
357           CMP CONCOM1
358           BNE REWIND
359  ZEROS    LDA #$3C
360           ADD CONCOM1
361           STA PTOB
362           JSR ACTIVAD
363           LDA KBD
364           STA PTOC
365           RTS
366
367  * SUBRUTINA PARA BLANQUEADO DE DISPLAYS (CLDIS) *
368
369  CLDIS    LDA #$3B
370  WHITE    INCA
371           STA PTOB
372           JSR ACTIVAD
373           CMP #$3F
374           BNE WHITE
375           RTS
376
377  * SUBRUTINA SUMA 3*
378
379  SIGMA    STA X
380           ADD #$03
381           INCX
382           RTS
383
384  * SUBRUTINA PARA COMPRENSION DE DATOS (RESUME) *
385
386  RESUME   LDA #$00
387           STA CONTAD
388     ROTA  LDX CONCOM1
389           LDA X
390           LDX #$00
391           CMP #$00
392           BEQ SZER
```

```
393   DESPLZ    LSLA
394             INCX
395             CPX #$04
396             BNE DESPLZ
397   SZER INC  CONCOM1
398             LDX CONCOM1
399             ADD X
400             LDX CONCOM2
401             STA X
402             INC CONCOM2
403             INC CONCOM1
404             INC CONTAD
405             LDX CONTAD
406             CPX #$02
407             BNE ROTA
408             RTS
409
410   * SUBRUTINA PARA EJECUCION DEL TIPO DE CONTROL (CONTROLES) *
411
412 CONTROLESLDA #$00
413             STA PARAM
414             CLI
415             LDA TIPO
416             CMP #$00
417             BEQ CONTROL1
418             CMP #$01
419             BEQ CONTROL2
420             CMP #$02
421             BEQ CONTROL3
422             CMP #$03
423             BEQ CONTROL4
424             RTS
425   CONTROL1  JSR TERMOMETRO
426             RTS
427   CONTROL2  JSR ON_OFF
428             RTS
429   CONTROL3  JSR PROPORC
430             RTS
431   CONTROL4  JSR PROGRAM
432             RTS

433   * SUBRUTINA TERMOMETRO TEMPORIZADO *

434 TERMOMETRO  LDA #$01
435             JSR ACTIVAE
436             JSR RELOJ_OFF
437   RE_TERMO  JSR LECTER
438             JSR DESP_BCD
439             JSR CARGATPO
```

```
440     LOOPTER   JSR LEE_KBD
441               LDA KBD
442               STA PTOC
443               BRSET 4,PTOA,DESP_TPO
444               JSR DESP_BCD
445     VER_FIN   LDA TESP
446               BEQ FIN_TPO
447               BRA LOOPTER
448     DESP_TPO  JSR DESP_TIME
449               BRA VER_FIN
450     FIN_TPO   JSR RELOJ_OFF
451               BRA RE_TERMO
452               RTS
453
454   * SUBRUTINA PARA CARGA DE TIEMPO DE RETARDO (CARGATPO) *

455   CARGATPO    LDA VTIME0
456               ADD #$01
457               STA TDR
458               LDA VTIME1
459               STA TIEMPO1
460               LDA VTIME2
461               STA TIEMPO2
462               LDA VTIME3
463               STA TIEMPO3
464               LDA #$01
465               STA TESP
466               JSR RELOJ_ON
467               RTS

468   * SUBRUTINA CONTROL ON-OFF *

469     ON_OFF    LDA #$02
470               JSR ACTIVAE
471               JSR ALCANZAT
472               LDA #$01
473               JSR ALARMA
474               RTS

475   * SUBRUTINA CONTROL PROPORCIONAL *

476     PROPORC   LDA #$03
477               JSR ACTIVAE
478               JSR MANTENT
479               RTS
```

```
480  * SUBRUTINA PARA MANTENER TEMPERATURA (MANTENT) *

481  MANTENT  JSR LEE_KBD
482           LDA #$03
483           CMP TIPO
484           BNE ALOTRO
485           LDA TESP
486           BEQ SALMAN
487  ALOTRO   JSR LECTER
488           LDA KBD
489           STA PTOC
490           BRSET 4,PTOA,VER_TPO
491  VER_BCD  JSR DESP_BCD
492           BRA VER_SIG
493  VER_TPO  LDA TIPO
494           CMP #$03
495           BNE VER_BCD
496           JSR DESP_TIME
497  VER_SIG  LDA TP1
498           CMP R1
499           BMI MANTENT
500           BHI CALENTAR
501           LDA TP0
502           CMP R0
503           BLS MANTENT
504           SUB R0
505           CMP #$05
506           BHI CALENTAR
507           BRA MANTENT
508  SALMAN   JSR RELOJ_OFF
509           RTS
510  CALENTAR JSR ALCANZAT
511           BRA MANTENT

512  * SUBRUTINA CONTROL PROPORCIONAL PROGRAMADO *

513  PROGRAM  LDA #$04
514           JSR ACTIVAE
515           LDA #$00
516           STA TP0
517           STA TP1
518           JSR LECTER
519  INCREMT  JSR INCTEM
520           LDA TP1
521           CMP R1
522           BMI INCREMT
523           BHI LOOPPROG
524           LDA TP0
525           CMP R0
526           BMI INCREMT
```

```
527  LOOPPROG JSR ALCANZAT
528           JSR CARGATPO
529           JSR MANTENT
530           JSR INCTEM
531           LDA TF1
532           CMP TP1
533           BMI ASIGNAN
534           BHI LOOPPROG
535           LDA TF0
536           CMP TP0
537           BLS ASIGNAN
538           BRA LOOPPROG
539  ASIGNAN  INC CTA
540           LDA CTA
541           CMP #$02
542           BEQ PRIMER_AL
543  ASIGNAC  LDA TF0
544           STA TP0
545           LDA TF1
546           STA TP1
547           BRA LOOPPROG
548  PRIMER_AL LDA #$01
549           JSR ALARMA
550           BRA ASIGNAC
551
552  * SUBRUTINA PARA INCREMENTO DE TEMPERATURA (INCTEM) *

553  INCTEM   LDA #$00
554           STA CARRY
555           LDA VAR0
556           STA SUMA
557           LDA TP0
558           AND #$0F
559           JSR SUMADOR
560           STA PAR1
561           LDA VAR1
562           STA SUMA
563           LDA TP0
564           AND #$F0
565           JSR DESPLAZD
566           JSR SUMADOR
567           CLR TP0
568           JSR DESPLAZI
569           ADD PAR1
570           STA TP0
571           LDA VAR2
572           STA SUMA
573           LDA TP1
574           AND #$0F
575           JSR SUMADOR
576           STA PAR1
```

```
577             LDA VAR3
578             STA SUMA
579             LDA TP1
580             AND #$F0
581             JSR DESPLAZD
582             JSR SUMADOR
583             CLR TP1
584             JSR DESPLAZI
585             ADD PAR1
586             STA TP1
587             RTS
```

```
588   * SUBRUTINA PARA SUMA HEXADECIMAL (SUMADOR) *
```

```
589   SUMADOR   ADD CARRY
590             ADD SUMA
591             CMP #$09
592             BHI SEIS
593             LDX #$00
594             STX CARRY
595             BRA FINSUMA
596   SEIS      ADD #$06
597             LDX #$01
598             STX CARRY
599   FINSUMA   AND #$0F
600             RTS
```

```
601   * SUBRUTINAS PARA DESPLAZAMIENTO BINARIO (DESPLAZAD/DESPLAZAI) *
```

```
602   DESPLAZD  LDX #$04
603   CICLOD    LSRA
604             DECX
605             BNE CICLOD
606             RTS
607
608   DESPLAZI  LDX #$04
609   CICLOI    LSLA
610             DECX
611             BNE CICLOI
612             RTS
613
614
615   * SUBRUTINA PARA LECTURA DE TEMPERATURA (LECTER) *
```

```
616   LECTER    JSR LEE_ADC       * LEE LA TEMPERATURA DEL A/D
617             LDA T
618             CMP #$FF          * T ?= FF
619             BEQ POR_AR        * DE SER ASI, VERIFICA SIGUIENTE
ENTRADA
620             CMP #$00          * T ?= 00
621             BEQ POR_AB        * DE SER ASI, TOMA LA TEMP. INFERIOR
```

```
622  SAL_TER   RTS
623  POR_AR    LDA DMUX          * VERIFICACION DE SIGUIENTE ENTRADA
624            CMP #$03          * DMUX = 3
625            BEQ ERROR_L       * DE SER ASI, HAY UN ERROR EN EL DATO
626            INC DMUX          * ++DMUX
627            BRA LECTER        * VUELVE A ITERAR
628  POR_AB    LDA DMUX
629            BEQ SAL_TER       * DE SER ASI, LA TEMP. ES CORRECTA
630            DEC DMUX          * --DMUX
631            JSR LEE_ADC       * LEE LA TEMPERATURA ANTERIOR
632            BRA SAL_TER       * CONTINUA
633  ERROR_L   JSR RELOJ_OFF
634            LDA #$02
635            JSR ALARMA
636            RSP
637            JMP $009F         * REGRESA A MENU
638
639  * SUBRUTINAS DE ENCENDIDO/APAGADO DE CRONOMETRO (RELOJ_ON/OFF) *

640  RELOJ_ON  BSET 4,TCR
641            BCLR 6,TCR
642            RTS
643  RELOJ_OFF      BCLR 4,TCR
644            BSET 6,TCR
645            RTS

646  * SUBRUTINAS DE ENC/APAG DE POTENCIA (POTENCIA_ON/POTENCIA_OFF)

647 POTENCIA_ON LDA #$01
648            BRA SPOT
649 POTENCIA_OFF LDA #$00
650  SPOT      STA PTOB
651            LDA POT
652            STA PTOC
653            LDA #$0F
654  RETPOT    DECA
655            BNE RETPOT
656            LDA #$FF
657            STA PTOC
658            RTS
659
660  * SUBRUTINA PARA ALCANCE DE TEMPERATURA (ALCANZAT) *

661  ALCANZAT  JSR POTENCIA_ON
662  ALCANCE   JSR LEE_KBD
663            JSR LECTER
664            JSR DESP_BCD
665            LDA #$03
666            CMP TIPO
667            BNE NOEST
```

```
668                 LDA TESP
669                 BEQ SALTO
670     NOEST LDA TP1
671                 CMP R1
672                 BMI SALTO
673                 BHI ALCANCE
674                 LDA TP0
675                 CMP R0
676                 BLS SALTO
677                 BHI ALCANCE
678     SALTO       JSR POTENCIA_OFF
679                 RTS
680
681     * SUBRUTINA PARA ACCIONAMIENTO DE LA ALARMA (ALARMA) *
```

```
682     ALARMA      STA PTOB
683                 LDX ALARM
684                 STX PTOC
685                 LDX #$0F
686     ESPERA1     DECX
687                 BNE ESPERA1
688                 LDX #$0F
689                 STX PTOC
690                 CMP #$01
691                 BEQ CONTFL
692                 JSR ESPTEC
693     CONTFL      LDA #$00            * APAGA ALARMA
694                 STA PTOB
695                 LDA ALARM
696                 STA PTOC
697                 LDA #$0F
698     ESPERA2     DECA
699                 BNE ESPERA2
700                 LDA #$0F
701                 STA PTOC
702                 RTS
```

```
703     * SUBRUTINA PARA ESPERA DE TECLA (ESPTEC) *
```

```
704     ESPTEC      LDA KBD
705                 STA PTOC
706     ESPT1       BRCLR 4,PTOA,ESPT1
707     ESPT2       BRSET 4,PTOA,ESPT2
708                 LDA PTOA
709                 AND #$0F
710                 LDX #$0F
711                 STX PTOC
712                 RTS
713
```

```
714     * SUBRUTINA PARA LEER EL SUCESO DE CLR EN EL BUS (LEE_KBD) *

715     LEE_KBD  LDA PARAM
716              CMP #$00
717              BNE PORESP
718              LDA KBD
719              STA PTOC
720              LDA #$04
721     DELAYKBD DECA
722              BNE DELAYKBD
723              LDA PTOA
724              AND #$0F
725              LDX #$0F
726              STX PTOC
727              BRA OTROL
728     PORESP   JSR ESPTEC
729     OTROL    CMP #$04
730              BEQ ABORTA
731              RTS
732     ABORTA.  JSR RELOJ_OFF
733              JSR POTENCIA_OFF
734              LDA #$00
735              STA PARAM
736              RSP
737              JMP $009F
738
739     * SUBRUTINA PARA LECTURA DEL ADC (LEE_ADC) *

740     LEE_ADC  LDA DMUX
741              STA PTOB
742              LDA MUX_A
743              STA PTOC
744              LDA #$0F
745     DELAY1   DECA
746              BNE DELAY1
747              LDA #$00
748              STA PTOC
749              LDA #$1E
750     DELAY2   DECA
751              BNE DELAY2
752              LDX #$0F
753              STX PTOC
754              LDA #$1E
755     DELAY3   DECA
756              BNE DELAY3
757              LDA #$04
758              STA PTOC
759     FINCV    BRCLR 0,PTOA,FINCV
760              LDA OEAD
761              STA PTOC
762              LDA #$27
```

```
763   DELAY4   DECA
764            BNE DELAY4
765            LDA PTOA
766            STA T
767            LDA #$0F
768            STA PTOC
769            LDA DMUX
770            STA R1
771            LDA T
772            STA R0
773            JSR CONVER
774            RTS
775
776   * SUBRUTINA PARA CONVERSION HEX-BCD (CONVER) *

777   CONVER   LDA #$26        * INICIALIZA LA PRIMERA POSICION DE
778            STA POS         * MEMORIA DONDE SE ALMACENA EL
779            LDA #$00        * NUMERO BCD
780            STA FLAG        * INICIALIZA A CERO LA BANDERA DE FIN
DE
781            STA COCIENT     * CONVERSION
782   CONV_BCD LDA R0          * CARGA EL NUMERO HEX PARTE BAJA Y
783            BMI CICLO       * CHECA SI ES NEGATIVO O MENOR A (Ah)
784            CMP #$0A        * R0-A >= 0
785            BPL CICLO
786            LDX R1          * CARGA PARTE ALTA DE NUMERO HEX
787            CPX #$00        * Y CHECA SI EXISTE
788            BEQ ALMAC       * SI R1=0 EL NUMBCD<10 (DECIMAL).
789            CMP #$0A
790            BMI SMSB
791   CICLO    LDX #$01        * INICIALIZA EL CONTADOR DE COCIENTE
792            STX COCIENT     * Y LO ALMACENA EN LA DIRECCION 22h
793   RESTA    SUB #$0A        * RESTA NHEX DE (Ah)
794            BMI IGNORE      * SI ES NEGATIVO BRINCA
795            CMP #$0A        * CHECA QUE SEA MENOR A (Ah)
796            BPL IGNORE
797   SMSB     LDX R1
798            CPX #$00
799            BEQ CHECA_C
800            DEC R1
801            INCA
802            STA REGTEM
803            LDA #$0A
804            SUB REGTEM
805            STA REGTEM
806            INC COCIENT
807            LDA #$FF
808            SUB REGTEM
809   IGNORE   INC COCIENT     * INCREMENTA CONTADOR DE COCIENTE
810            BRA RESTA
```

```
811  CHECA_C  BRA STORE          * ALMACENA LAS PRIMERAS CIFRAS BCD
812  P2       LDA COCIENT
813           CMP #$0A
814           BMI ALMAC          * SALTA A ALMAC
815           BRA CICLO          * SALTA A CICLO
816  ALMAC    LDX #$01           * PONE BANDERA DE FIN DE CONVERSION
817           STX FLAG
818  STORE    LDX POS            * ALMACENA EL NUMERO BCD
819           STA X
820           CPX #$23
821           BEQ FIN
822           DEC POS
823           LDX FLAG
824           CPX #$01
825           BEQ SIG
826           BRA P2
827  FIN      LDX #$23           * ALMACENA PARAMETROS A UTILIZAR EN
828           STX CONCOM1        * RESUME
829           LDX #$65
830           STX CONCOM2
831           JSR RESUME         * COMPRIME LOS BCDS
832           RTS
833  SIG      LDA #$00           * ALMACENA CEROS EN LAS POSICIONES
834           BRA STORE          * RESTANTES
835
836  * SUBRUTINA PARA DESPLEGAR DATOS BCD EN LOS DISPLAYS (DESP_BCD) *

837  DESP_BCD LDA #$00
838           STA CAUX
839           LDA R0
840           JSR PROCD1
841           LDA R0
842           JSR PROCD2
843           LDA R1
844           JSR PROCD1
845           LDA R1
846           JSR PROCD2
847           RTS

848  * SUBRUTINA PARA DESPLEGADO DEL TIEMPO (DESP_TIME) *

849  DESP_TIME LDA #$00
850           STA CAUX
851           LDA TDR
852           SUB #$01
853           JSR PROCD1
854           LDA TIEMPO1
855           JSR PROCD1
856           LDA TIEMPO2
857           JSR PROCD1
```

```
858             LDA TIEMPO3
859             JSR PROCD1
860             RTS
861
862  * SUBRUTINA PARA MANEJO DEL CRONOMETRO (TIMER) *

863  TIMER      LDA TIEMPO1
864             ORA TIEMPO2
865             ORA TIEMPO3
866             BEQ FINTEMP
867             LDA #$0A
868             STA TDR
869             LDA TIEMPO1
870             BEQ VERS
871             DEC TIEMPO1
872  CONTEMP    LDA #$01
873             STA TESP
874             BCLR 7,TCR
875             RTI
876  VERS       LDA TIEMPO2
877             BEQ VERSS
878             LDA #$09
879             STA TIEMPO1
880             DEC TIEMPO2
881             BRA CONTEMP
882  VERSS      LDA TIEMPO3
883             BEQ FINTEMP
884             LDA #$09
885             STA TIEMPO1
886             STA TIEMPO2
887             DEC TIEMPO3
888             BRA CONTEMP
889  FINTEMP    LDA #$00
890             STA TESP
891             BCLR 7,TCR
892             JSR RELOJ_OFF
893             RTI
894
895  *SUBRUTINAS PARA PREPARAR DATO A VISUALIZAR (PROC1/PROC2)*
896

897  PROCD1     LSLA
898             LSLA
899             AND #$3C
900             ADD CAUX
901             STA PTOB
902             JSR ACTIVAD
903             INC CAUX
904             RTS
905
```

```
906   PROCD2   LSRA
907            LSRA
908            AND #$3C
909            ADD CAUX
910            STA PTOB
911            JSR ACTIVAD
912            INC CAUX
913            RTS
914
915   * SUBRUTINA PARA MANEJO DE INTERRUPCION (INTERRUP) *
```

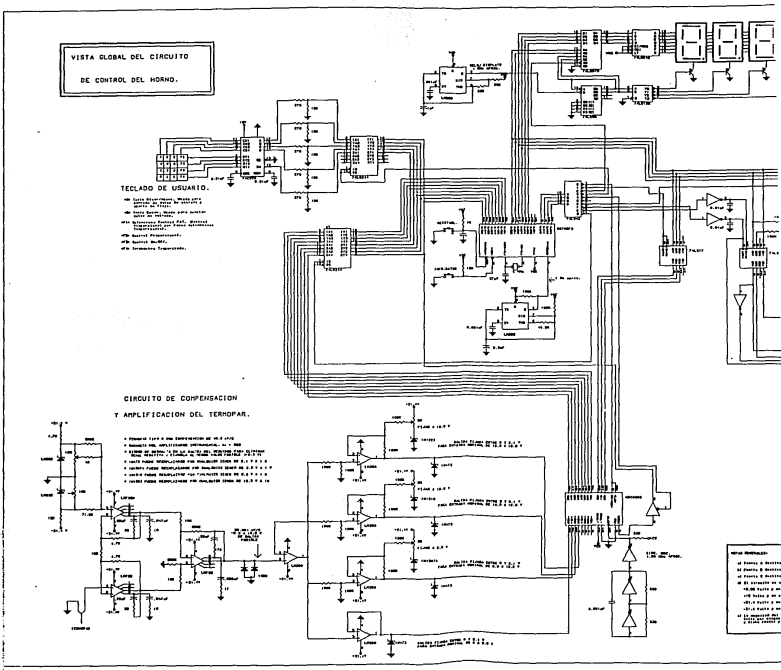
```
916   INTERRUP JSR RELOJ_OFF
917            JSR POTENCIA_OFF
918            LDA #$02
919            STA PARAM
920            LDA #$00
921            STA CAUX
922            LDA TIPO
923            ADD #$01
924            JSR ACTIVAE
925            LDA TIPO
926            CMP #$01
927            BEQ PRONOF
928            CMP #$02
929            BEQ PRONOF
930            CMP #$03
931            BEQ PRG
932            CMP #$00
933            BEQ DESTIM
934            RTI
935   PRONOF   LDA #$05          * DESPLIEGA SET-POINT
936            JSR ACTIVAE
937            LDA TP0
938            JSR PROCD1
939            LDA TP0
940            JSR PROCD2
941            LDA TP1
942            JSR PROCD1
943            LDA TP1
944            JSR PROCD2
945            JSR LEE_KBD
946            JSR POTENCIA_ON
947            JMP SALIDAI
948   PRG      LDA #$05          * DESPLIEGA SET-POINT
949            JSR ACTIVAE
950            LDA TF0
951            JSR PROCD1
952            LDA TF0
953            JSR PROCD2
954            LDA TF1
```

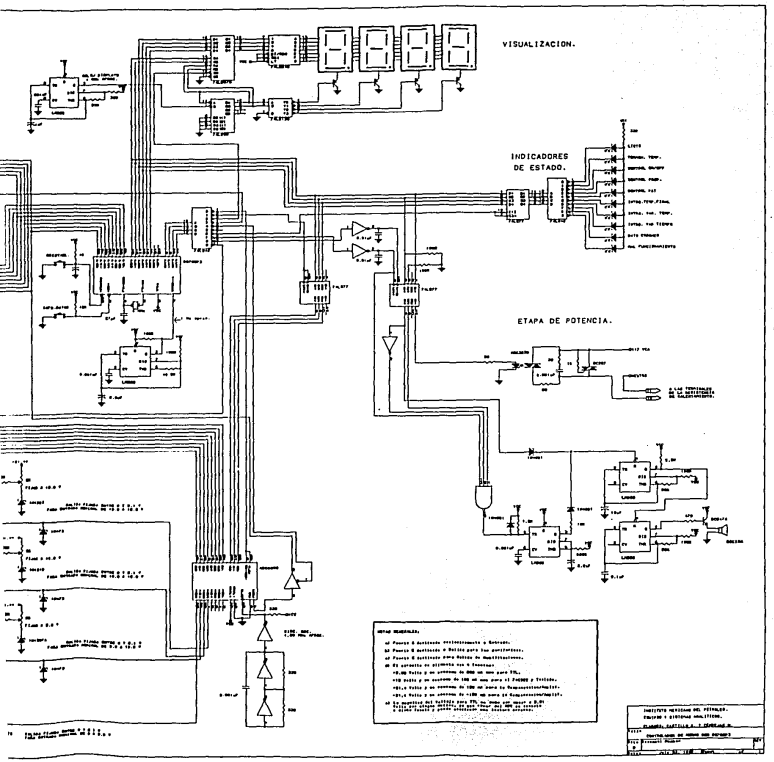
```
955                 JSR PROCD1
956                 LDA TF1
957                 JSR PROCD2
958                 JSR LEE_KBD
959                 LDA #$06            * DESPLIEGA DELTA-TEM.
960                 JSR ACTIVAE
961                 LDA #$00
962                 STA CAUX
963                 LDA VAR0
964                 JSR PROCD1
965                 LDA VAR1
966                 JSR PROCD1
967                 LDA VAR2
968                 JSR PROCD1
969                 LDA VAR3
970                 JSR PROCD1
971                 JSR LEE_KBD
972   DESTIM        LDA #$07            * DESPLIEGA DELTA-TIME.
973                 JSR ACTIVAE
974                 LDA #$00
975                 STA CAUX
976                 LDA VTIME0
977                 JSR PROCD1
978                 LDA VTIME1
979                 JSR PROCD1
980                 LDA VTIME2
981                 JSR PROCD1
982                 LDA VTIME3
983                 JSR PROCD1
984                 JSR LEE_KBD
985                 LDX TIPO
986                 CPX #$00
987                 BEQ NOPOTEN
988                 JSR POTENCIA_ON
989   NOPOTEN       JSR RELOJ_ON
990   SALIDA1       LDA TIPO
991                 ADD #$01
992                 JSR ACTIVAE
993                 LDA #$00
994                 STA PARAM
995   FININ         RTI
996
997   * VECTORES DE INTERRUPCION *
```

```
998                 ORG $07F8
999                 RMB TIMER
1000                RMB INTERRUP
1001                RMB INTERRUP
1002                RMB $0080
1003
1004                ORG $0784
1005                FCB $20
```

VISTA GLOBAL DEL CIRCUITO

DE CONTROL DEL HORNO.




VISUALIZACION.
INDICADORES
DE ESTADO.
ETAPA DE POTENCIA.