# CONTROL DE UNA CALDERA CON UN MICROPROCESADOR

POR

LOURDES P. GIMENEZ L.

RICARDO A. MORIN S.

PROYECTO DE GRADO PRESENTADO ANTE LA ILUSTRE UNIVERSIDAD SIMON BOLIVAR COMO REQUISITO PARA OPTAR POR EL TITULO DE INGENIERO ELECTRONICO.

UNIVERSIDAD SIMON BOLIVAR

SARTENEJAS- BARUTA

VENEZUELA

JULIO 1980

UNIVERSIDAD SIMON BOLIVAR
COORDINACION DE INGENIERIA ELECTRONICA

# A C T A

Los abajos firmantes, miembros del jurado examinado del proyecto:

CONTROL DE UNA CALDERA CON UN MICROPROCESADOR

Luego de haber revisado el proyecto y realizado el examen correspondiente a los ponentes del proyecto, hemos llegado al siguiente veredicto.

| | ALUMNO | CARNET | CALIFICACION |
|---|---|---|---|
| 1) | Lourdes P. Giménez L. | 75-5566 | |
| 2) | Ricardo A. Morin S. | 75-5900 | |

En virtud de lo cual se levanta la presente acta, en Caracas, a los _____ días del mes de _____ de 1980.

_____ _____ _____

_____ _____

Observaciones: _____

_____

_____

Refrendado: _____

Coordinador Docente

# CONTROL DE UNA CALDERA CON UN MICROPROCESADOR

POR

LOURDES P. GIMENEZ L.

RICARDO A. MORIN S.

PROYECTO DE GRADO PRESENTADO ANTE LA ILUSTRE UNIVERSIDAD SIMON BOLIVAR COMO REQUISITO PARA OPTAR POR EL TITULO DE INGENIERO ELECTRONICO.

UNIVERSIDAD SIMON BOLIVAR

SARTENEJAS- BARUTA
VENEZUELA
JULIO 1980

CONTROL DE UNA CALDERA CON UN MICROPROCESADOR

P O R

LOURDES P. GIMENEZ L.

RICARDO A. MORIN S.

PROFESOR GUIA: GIULIO BEVILACQUA

## R E S U M E N

El presente trabajo consiste en el diseño y construcción de un controlador programable para una caldera de laboratorio, basado en un microprocesador. La programación y supervisión de la máquina se hace a través de un terminal CRT, pudiendo hacerse en forma mínima mediante un teclado y visualizadores numéricos.

Se ofrecen además, ciertas funciones gráficas, para facilitar la programación y supervisión.

El controlador está basado en el sistema operativo para aplicaciones en tiempo real RMX/80 INTEL.

A : Mi madre y a Hilde

Ricardo

A : Fernando, Anita y Patricia

Paloma

## A G R A D E C I M I E N T O

Hubiera sido imposible la realización de este trabajo sin la ayuda y el estímulo de muchas personas relacionadas directa o indirectamente con éste. Por esta razón deseamos agradecer:

A la Universidad Simón Bolívar, por darnos esta oportunidad.

A nuestro tutor y asesores, Giulio Bevilacqua, Víctor Guzmán y Javier Bonet, por su valiosa colaboración.

A todos nuestros compañeros, en especial al Cabo Reyes, por habernos ayudado, acompañado y soportado todo este año.

A Fernando Giménez

A Evelin Hernández

A Baldwin y Margarita

Al personal técnico de la Universidad.

# I N D I C E

PAGINA

PAGINA

# I N T R O D U C C I O N

El objeto del presente trabajo es el diseño y la - construcción de un controlador y supervisor de una caldera de laboratorio, basado en un microcomputador. El controlador sustituye las funciones de uno tradicional además de - añadir la capacidad de programación de la operación de la máquina. La comunicación entre el usuario y el controlador se realiza mediante un CRT, donde se introduce la información necesaria; se visualiza el estado de la caldera y se generan los mensajes de error y emergencia necesarios. Además, se incluye la facilidad de una operación mínima por teclado y visualizador numérico (LED) en caso de no disponerse de un CRT.

El diseño de "software" se ha realizado en base a un sistema operativo para aplicaciones en tiempo real - - (RMX/80 INTEL) a manera de nueva experiencia sobre esta forma de trabajo en microprocesadores.

## NOCIONES GENERALES:

En general, una caldera consiste en un recipiente - cerrado, sometido a presión, utilizado para generar vapor a partir de agua por calentamiento, siendo por lo tanto el su

ministro eficiente de calor y el suministro de agua los elementos más importantes a considerarse.

Existen una infinidad de tipos de calderas[1] dependiendo primordialmente del uso al que se va a destinar, de su capacidad de generación y presión de operación. La aplicación de este tipo de máquinas es muy amplia en el campo industrial para manejar máquinas de vapor, bombas y turbinas. Además, muchos procesos químicos incluyen la aplicación de vapor de agua a una cierta presión.

Las calderas pueden clasificarse fundamentalmente en dos tipos:

- Calderas de tubos de fuego
- Calderas de tubos de agua

En las calderas de tubos de fuego (como su nombre lo indica), el calor se hace pasar por el interior de tubos, los cuales están rodeados de agua. Se clasifican a su vez en tres tipos: verticales, tipo-locomotora y horizontales.

Las calderas verticales de tubos de fuego están formadas por un tanque cilíndrico con hogar de fuego en la parte inferior donde se ubican tubos verticales desde el hogar hasta la parte superior donde se han practicado agujeros para retenerlos. Si parte de estos tubos está expues-

ta al vapor, se denomina a tubos verticales expuestos, de - lo contrario, se trata de una caldera de tubos de fuego verticales sumergidos. El diseño a tubos expuestos permite un supercalentamiento del vapor, deseable en muchos casos y el de tubos sumergidos, evita el sobrecalentamiento de los elementos sometidos a exposición directa de calor.

La utilización de las calderas verticales generalmente es en máquinas móviles, ya que el diseño las hace relativamente portátiles. Se construyen para presiones de 100 P.S.I. y 5 a 75Hp.

En las calderas del tipo de locomotora, el horno se encuentra totalmente rodeado de agua, así el compartimiento de ésta, está formado por la unión del cuerpo de la caldera y el horno mismo. Aquella parte de este tipo de caldera, que no configura el horno, es un cilindro horizontal que - contiene el agua y el vapor y está atravesada en su inte-- rior por tubos de fuego horizontalmente. Estas calderas son muy económicas y almacenan gran cantidad de agua y vapor. Se fabrican para presiones de 350 P.S.I. y 250 Hp.

Las calderas de tubos de fuego del tipo horizontal, consisten en un cilindro colocado horizontalmente surcado - también horizontalmente por varios tubos de fuego, donde - circulan los gases calientes del horno a la chimenea. Den

tro de este tipo, existen muchas variaciones, dependiendo de la forma como son calentados, así, la caldera que ocupa - - nuestra atención es del tipo horizontal de tubos de fuego y flujo forzado a dos pasos, además posee un quemador re- - tractil.

Este tipo de calderas son muy compactas, de costo inicial bajo y existen para presiones de 15 a 250 P.S.I. y de 15 a 600 Hp.

Las calderas de tubos de agua estan compuestas por tubos, alrededor de las cuales circulan gases calientes. Este tipo de caldera se diseña para grandes capacidades de generación de vapor, son más rápidas y eficientes que las - vistas anteriormente pero sin embargo, su costo inicial es apreciablemente mayor. Se construyen para presiones de 300 P.S.I. y 200 a 300 h.p.

CONTROL AUTOMATICO DE CALDERAS:

Normalmente, en aquellas calderas que poseen con-- trolador automático, su operación es realizada y supervisada en base a elementos electromecánicos y relés. Sus funciones son las siguientes:

- Suplir el combustible necesario para mantener la operación de la máquina de acuerdo al consumo de vapor.

- Sensar situaciones de emergencia tales como sobre presión, apagado de la llama, nivel de agua inadecuado y mal funcionamiento del quemador cuando se trata del tipo de fuel-oil, donde se controla la relación aire combustible.
- Corregir las situaciones anormales, dar la alarma y/o detener el funcionamiento de la unidad en forma adecuada.

Este tipo de controlador convencional mantiene la presión de operación entre dos limites alrededor del punto fijado, donde este punto es pre-seleccionado, ajustando la unidad y permanece constante todo el tiempo. Si se deseara no solo alterar el punto de equilibrio sino hacerlo programable durante el día, esto implicaría una lógica basada en los mismos principios de funcionamiento (relés) prácticamente irrealizable, por lo que habría que cambiar la filosofía del diseño. Esta filosofía podría orientarse hacia el diseño lógico convencional (compuertas, flip-flops, memorias, etc.) o al diseño basado en un microprocesador.

Mantener la operación de una caldera igual que con un controlador electromecánico y añadirle además la programabilidad podría bien realizarse en base a diseño digital - convencional, resultando un controlador de una lógica no muy sencilla y probablemente muy rígida. Cuando un diseño digi-

tal convencional adquiere ciertas dimensiones, se justifica la utilización de un microprocesador, dado el bajo costo - que estos elementos poseen en el mercado, además de añadir grandes ventajas al sistema tales como:

- Sencillez del diseño en "hardware".
- Comunicación fácil, eficiente y versátil entre el usuario y la máquina.
- Posibilidad de añadir mejoras al sistema a medida que se va configurando, e incluso después de terminado.

Si además de esto, añadimos la utilización de un - sistema operativo, tal como el RMX/80 se multiplicarán estas cualidades, además de adelantar cierta investigación por experimentación que esperamos aporte nueva información a quienes estudian y trabajan en la Universidad Simón Bolívar.

# C A P I T U L O II

## DISEÑO

### 2.1.- DESCRIPCION DE LA CALDERA

La caldera para la cual se construyó el controlador, es del tipo de tubos de fuego, de hogar horizontal y de dimensiones bastante reducidas, capaz de operar hasta una presión máxima de 150 P.S.I..

En realidad no posee aplicaciones prácticas en el área industrial, pero constituye un buen equipo de prueba ya que todos los elementos que la constituyen (sensores, bomba de agua, transformador de encendido) forman parte de equipos mayores utilizados en la industria, con lo cual, las condiciones de operación permanecen muy semejantes.

Para su manejo, la caldera cuenta con el siguiente equipo y facilidades:

- Un transformador para el encendido del combustible a quemarse.
- Un quemador para gas con un par de electrodos para acoplar el transformador de encendido.
- Cuatro agujeros con rosca para introducir los elementos sensores de nivel de agua, los cuales se detallarán mas adelante.
- Agujeros con rosca para colocar un presostato, una

válvula de seguridad mecánica, una válvula manual - para salida de vapor del usuario, una válvula electromecánica para ser manipulada por el controlador, medidores ópticos de presión y temperatura y un - - transductor de presión o temperatura (depende de lo que se desee).

- Una válvula de purga para vaciar el contenido de - agua.
- Una bomba de agua de alta presión para el llenado - de la caldera en operación, manejada por un motor de inducción, y
- Un visualizador del nivel de agua compuesto por un tubo de vidrio.

El esquema total de la caldera se muestra en la Figura 2.1.

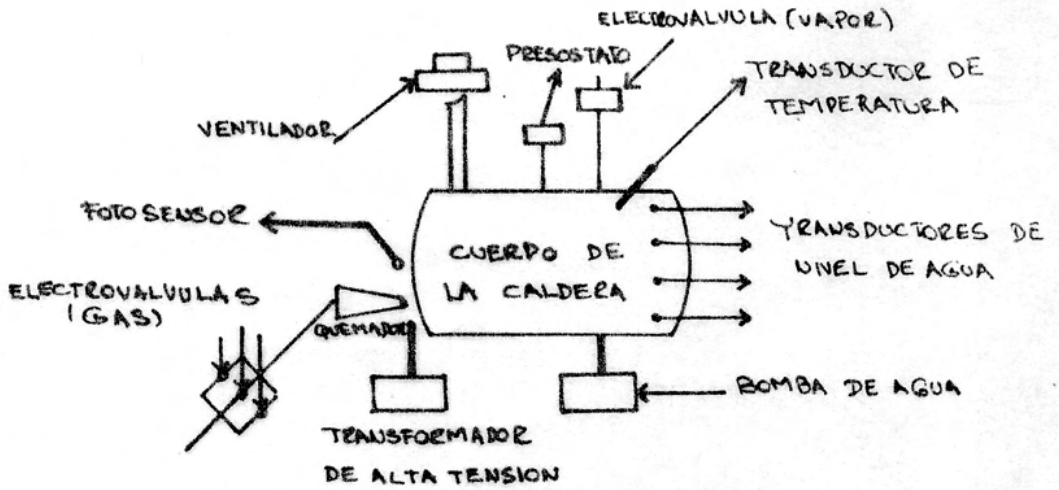



Figura 2.1.

## 2.2.- ESCOGENCIA DE LOS TRANSDUCTORES

### 2.1.1.- TRANSDUCTOR DE TEMPERATURA

Debido a la dificultad de conseguirse y a los altos costos, no se empleó directamente un transductor de presión, en cambio esta medida se realiza en forma indirecta a través de la temperatura, obteniéndose, en condiciones de vapor sobresaturado una conversión muy precisa entre estas dos variables. Por esta razón, el transductor de temperatura es el medio de funcionamiento del controlador debido a que provee además de la medición directa de temperatura una medida indirecta de la presión en el interior de la caldera.

Las alternativas en cuanto a este transductor fueron las siguientes:

- Termocuplas
- Termistores
- Bulbos resistivos.

Los termistores fueron rechazados debido a que presentan problemas de envejecimiento a las condiciones a las que serían sometidos, además sus rangos de temperatura no son adecuados. Experimentos realizados demostraron que los termistores no son óptimos en este tipo de aplicación[2].

Las termocuplas presentan un buen rango de temperatura y presión de trabajo y una inmunidad al ruido bastante elevada a pesar de su señal tan pequeña. Esto se debe a su baja impedancia. El problema con estos elementos es que la señal obtenida en ellos es proporcional a la "diferencia" entre la temperatura del medio bajo prueba y el medio ambiente, con lo cual se crea la necesidad de introducir un factor de compensación de la temperatura ambiente, ya que está ofrece rangos de variación apreciables.

Los posibles métodos de compensación son los siguientes:

- Compensación por referencia, conectando otra termocupla en oposición con la primera pero a una fuente conocida de temperatura (hielo o un horno auxiliar) la cual hace inconveniente e impráctico el método para nuestros propósitos.

- Compensación eléctrica, intercalando un puente de Weathstone en una de las ramas de la termocupla. Uno de los elementos del puente es un termistor a la temperatura ambiente. Este esquema fue descartado debido a la inconvenencia de suplir una fuente de voltaje flotante muy regulada e inmune al ruido.

- Compensación por "software", sensando separadamente las medidas de la termocupla y de un termistor a temperatu-

ra ambiente con el microcomputador y ejecutando la compensación mediante programa.

Este hubiera sido el mejor método para una termocupla, pero consideramos que obtendríamos con ello dos fuentes de error en lugar de una, además de introducir una complicación innecesaria.

Debido a las razones anteriormente expuestas, decidimos utilizar como elemento de medición de temperatura un bulbo resistivo, particularmente, una resistencia de platino. Este elemento presenta una característica muy lineal e invariante a las condiciones más rigurosas de presión y temperatura.

El circuito de medición se diseñó con una fuente de corriente constante lo suficientemente baja para no introducir autocalentamiento, pero suficiente para obtener un buen nivel de señal. Por conveniencia diseñamos la fuente flotante para alimentar con la tensión generada un amplificador diferencial, obteniendo así mayor inmunidad al ruido.

La salida del amplificador diferencial se introduce en un amplificador que ajusta el nivel DC y añade cierta ganancia para llevar la señal al conversor A/D.

El conversor A/D consiste en un D/A y un comparador de voltaje, manejado por un programa desde microprocesador en base a un algoritmo de aproximaciones sucesivas implementado.

### 2.2.2.- SENSOR DE LA PRESENCIA DE LLAMA

Este transductor consiste en una foto-resistencia ubicada cerca de la llama. De acuerdo al nivel de ésta y si el ventilador que suministra aire al hogar se encuentra encendido o apagado la consistencia de la llama varía apreciablemente y por ende la resistencia ofrecida por el foto-sensor.

Polarizando adecuadamente esta foto-resistencia, tenemos una indicación eléctrica del fenómeno, compatible con nuestro sistema de control, luego de cierto condicionamiento de la señal (Capítulo 3).

### 2.2.3.- PRESOSTATO

La caldera cuenta con un presostato con presión ajustable y con histéresis también ajustable, el cual utilizamos como generador de interrupción del proceso en caso de una emergencia por sobrepresión.

### 2.2.4.- TRANSDUCTORES DE NIVEL DE AGUA

La caldera cuenta con receptáculos para instalar cuatro bujías de automóvil preparadas, para efectuar la medición de nivel de agua. Su principio de funcionamiento se basa en que la resistividad del agua es diferente a la del vapor, con lo cual, alimentando en una fuente de voltaje pulsante (para evitar polarización) pueden discriminarse la presencia de agua o de vapor.

### 2.2.5.- VALVULAS DE GAS

Para controlar el paso del gas (cantidad de calor que se introduce al sistema), se pensó utilizar una válvula proporcional para obtener un control contínuo, pero debido a la dificultad de encontrar este tipo de válvula tan pequeña y a su elevado costo, se decidió descartar esta posibilidad y atacar el problema con un controlador "ON-OFF".

Para disminuir las oscilaciones inherentes en un controlador ON-OFF se decidió diseñar un controlador ON-OFF modificado, construyendo un arreglo de tres válvulas solenoide en paralelo obteniéndose con regulación manual previa, tres niveles de llama.

## 2.3.- ACTUADORES

Para controlar el encendido y apagado de las válvulas, transformador de encendido y bomba de agua, fue preciso con

vertir las señales provenientes de la tarjeta de control - ya que estos no podían manejar los elementos de potencia - mencionados. Para esto, se diseñaron interruptores electrónicos basados en triacs, con excepción de la bomba de - agua que por limitaciones de potencia se emplearon tiristores. Las señales de disparo fueron sincronizadas por medio de un detector de cruce por cero para conmutar los niveles mínimos de voltaje posibles, evitando así introducir - ruido adicional al sistema. Además, estas señales se aislan ópticamente de la parte de potencia, esto permite en - primer lugar evitar el ruido y en segundo lugar la presencia accidental de voltajes y corrientes elevadas hacia la parte del microprocesador.

## 2.4.- CONTROLADOR ("HARDWARE").

El controlador se basa en un microprocesador 8080 - - INTEL incorporado a una tarjeta "OEM" modelo SBC 80/10. Se decidió utilizar esta tarjeta ya que el sistema operativo a ser utilizado (RMX/80) fue diseñado y es totalmente compatible con el "hardware" allí implementado.

El SBC 80/10 posee 4K de memoria ROM, 1K de memoria RAM, seis puertos programables para comunicación del sistema con el exterior, así como una USART para establecer comunicación serial con un CRT o un teletipo. Además se permite un nivel de interrupción y se proveen dos entradas exter

nas para tal fin.

Con el objeto de alojar en la tarjeta los programas - realizados para controlar el proceso, fue preciso implementar una expansión de memoria ROM de 5K. y una expansión de memoria RAM de 1K para manejo de tablas y variables. Se añadió además la circuitería necesaria para un reloj de tiempo real controlado por puerto e interrupciones[3].

La programación y control por el usuario se realiza a través de un CRT, pudiéndose también operar en forma mínima por medio de un teclado y unos visualizadores numéricos - (LED). La parte de la decodificación del teclado y control del visualizado también se diseñó y añadió al sistema manejándose a través de los puertos del SBC 80/10.

El sistema total quedó implementado con "hardware" de la forma indicada en la Figura 2.2.

## 2.5.- CONTROLADOR (SOFTWARE)

En el diseño de un sistema basado en RMX80, (Ver - Apendice B) es necesario destacar cuales son las funciones principales a ejecutarse en forma independiente para entonces asociar cada una con su correspondiente unidad de ejecucción o "task". Estas funciones en nuestro -

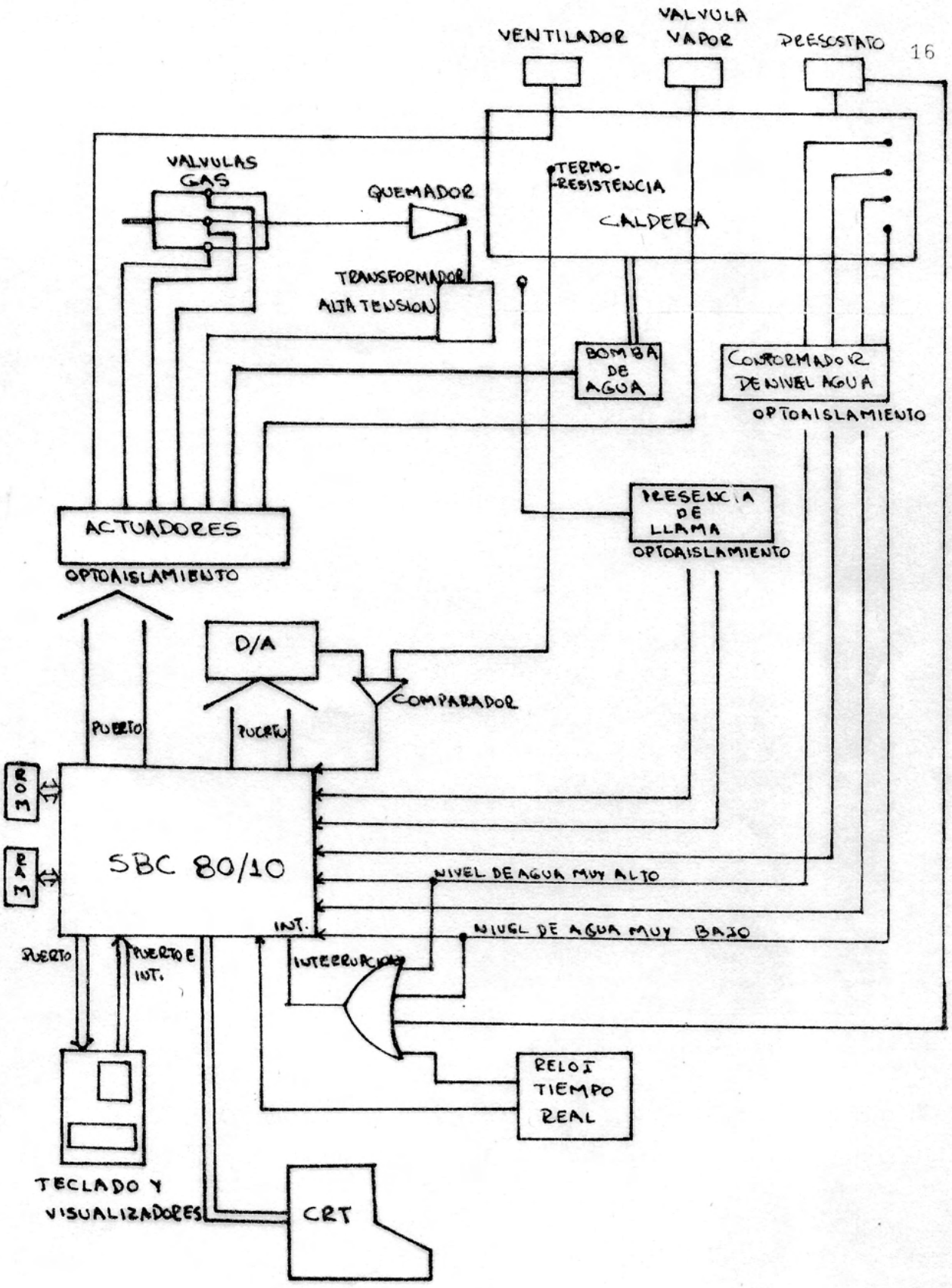

VENTILADOR
VALVULA VAPOR
PRESOSTATO
VALVULAS GAS
QUEMADOR
TERMO-RESISTENCIA
CALDERA
TRANSFORMADOR ALTA TENSION
BOMBA DE AGUA
CONFORMADOR DE NIVEL AGUA
OPTOAISLAMIENTO
ACTUADORES
OPTOAISLAMIENTO
PRESENCIA DE LLAMA
OPTOAISLAMIENTO
D/A
COMPARADOR
PUERTO
PUERTO
ROM
RAM
SBC 80/10
NIVEL DE AGUA MUY ALTO
INT.
NIVEL DE AGUA MUY BAJO
PUERTO
PUERTO E INT.
INTERRUPCION
RELOJ TIEMPO REAL
TECLADO Y VISUALIZADORES
CRT

Figura 2.2.

diseño son las siguientes:

1) Manejo de CRT: Incluye la generación de mensajes de error, mantenimiento de una entrada conversacional, validación de parámetros y presentación de información solicitada por el usuario.

2) Manejo de teclado y visualizador: Atención a los comandos ordenados por teclado y mantenimiento de información en el visualizador.

3) Algoritmo de control: Manejo de la caldera, control de ésta y supervisión de los estados de alarma y emergencia.

4) Reloj del sistema: Mantenimiento de un reloj con la hora del día ya que la caldera es programable en el tiempo.

5) Conversión analógica a digital: Algoritmo para tomar la medición de temperatura.

6) Manejo de interrupciones: Manejo de todas aquellas situaciones que involucren alguna interrupción.

La asignación de los "tasks" se hizo en base a esta tabla de funciones, con excepción de los siguientes puntos:

- La operación del visualizador se distribuyó en los tasks que manejan la información que se mantiene, con excepción del eco que se genera al operarse el teclado.

- El manejo del CRT se distribuyó entre dos "tasks", uno suplido por el RMX para manejo elemental de CRT y uno añadido por nosotros como monitor del sistema.

Por supuesto que estas funciones no son aisladas, y los lazos de comunicación entre ellos se logran mediante el empleo de "mensajes" y "exchanges". Un diagrama elemental de las funciones y los lazos de comunicación se presentan en la Figura 2.3.

En una aproximación, los rectángulos representan tasks, las flechas representan pares "mensaje", "exchange" y las rutinas de interrupción son bloques de programas más independientes.

El algoritmo de control:

En este punto es conveniente hacer una descripción del algoritmo de control empleado para alcanzar y mantener una referencia dada. Como se dijo anteriormente tipo de controlador es "on-off" modificado, por la particularidad de poseer tres niveles de inyección de calor. Sedefinen tres umbrales diferentes de comparación entre la referencia y el valor medido (3 PSI, 4 PSI, 5 PSI) asignándose cada uno de ésto a ca

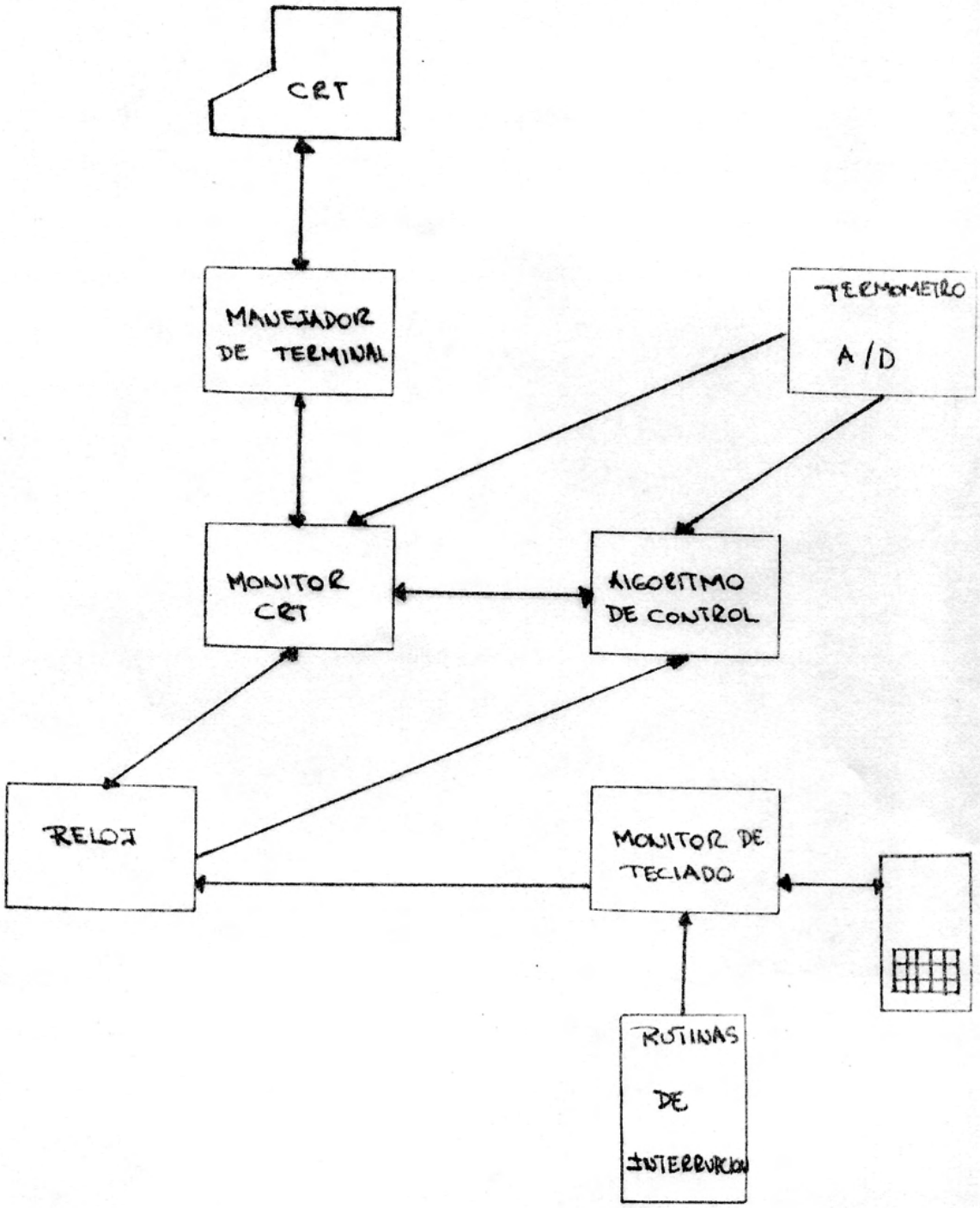
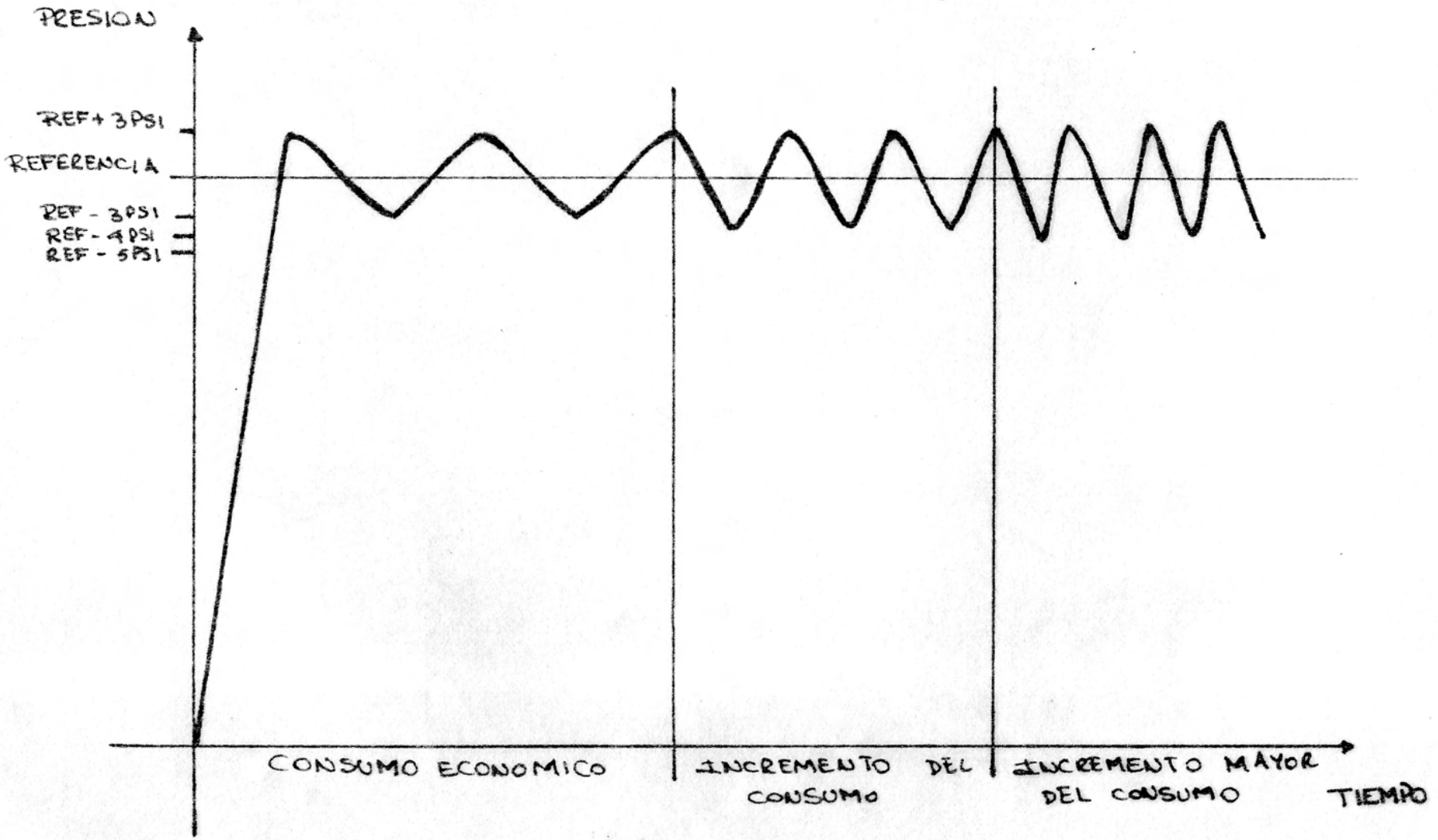

CRT
MANEJADOR DE TERMINAL
TERMOMETRO A/D
MONITOR CRT
ALGORITMO DE CONTROL
RELOJ
MONITOR DE TECLADO
RUTINAS DE INTERRUPCION

Figura 2.3.

da nivel de inyección de calor y un umbral equivalente al menor, como límite superior. Si la medición se encuentra por debajo de la referencia en 5 PSI, se activa el máximo calor hasta que se alcance el límite superior, donde se apaga el - suministro de gas' Al alcanzarse el nivel inferior de 3 - PSI, se activa el nivel de menor intensidad de calor, hasta que se llegue al nivel superior, apagándose la llama. Este proceso se repite indefinidamente, empleándose el menor suministro de gas posible hasta que se produzca un aumento en el consumo que impida esta operación. Si inyectando ca-- lor al gas mínimo, aún continúa aumentando la diferencia - (hacia los umbrales mayores inferiores), hasta superarse 4 PSI, se activa el segundo nivel de llama, tratando de mantener la estabilidad entre este umbral y el superior. Si aún continúa aumentando la diferencia, se incrementa de nuevo - el suministro de calor, para mentener al sistema en éste - peor caso por debajo de 3 P.S.I. sobre la referencia y sobre 5 P.S.I. bajo la referencia.

Se añade además ( ya que la referencia es programable) la apertura de una válvula de vapor cuando la referencia está a ménos de 5 P.S.I. del valor medido, para incrementar la velocidad de operación de la caldera al alcanzar las referencias programadas. El diagrama de tiempo de la Figura - 2.4. ilustra el comportamiento descrito.

Figura 2.4.


PRESION
REF+3PSI
REFERENCIA
REF-3PSI
REF-4PSI
REF-5PSI
CONSUMO ECONOMICO
INCREMENTO DEL CONSUMO
INCREMENTO MAYOR DEL CONSUMO
TIEMPO

El diagrama de flujo mostrado en la figura 2.5. ilustra el funcionamiento del algoritmo.

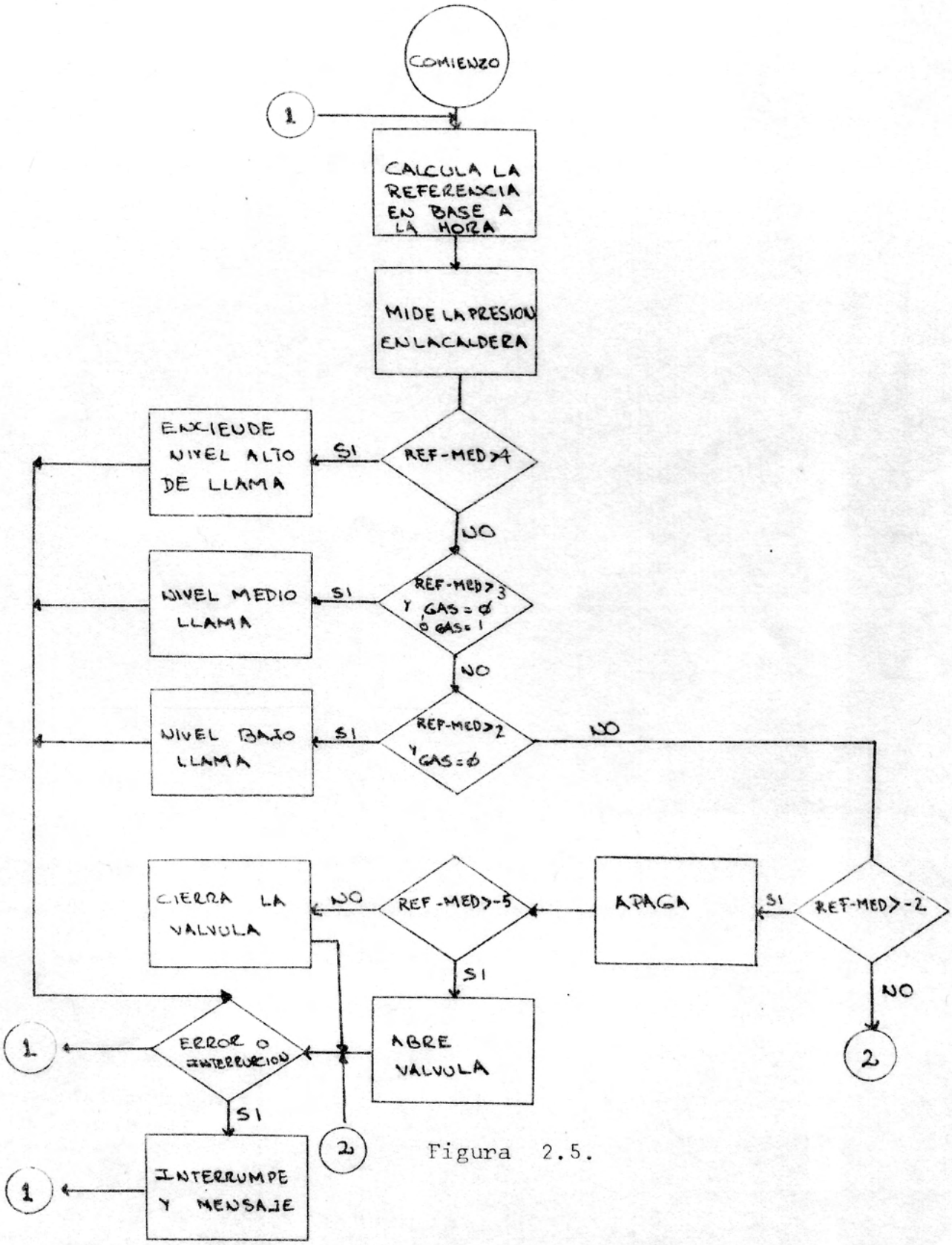



Figura 2.5.

# C A P I T U L O III

# IMPLEMENTACION

## 3.1.- TRANSDUCTORES

### 3.1.1.- CIRCUITO DE MEDICION DE TEMPERATURA

Como se señaló anteriormente el transductor escogido para hacer la medición de la temperatura (y por lo tanto de la presión) a que está trabajando la caldera, es una resistencia de platino, la cual ofrece una resistencia variable en función de la temperatura. Gracias a la gran linealidad de ésta función (Resistencia vs. temperatura), con nuestro rango de trabajo (o °C a 192°C ), se puede obtener un voltaje variable en función de la temperatura alimentando la termoresistencia con una fuente de corriente constante, como se muestra en la Figura 3.1.

El valor de la corriente se escogió de 5mA para obtener un rango de voltaje razonable con un coeficiente de error por autocalentamiento de la resistencia de platino mínimo.

Con este valor de corriente se obtiene una variación de voltaje de 0,50V. a 0,86V. para una variación de temperatura de 0 °C a 192 °C. Para hacer compatible esta señal con el conversor analógico-digital utilizado la varia--

ción de voltaje debe ser de 0V a 0 °C y 5V a 192°C, para - lo cual es necesario amplificar la señal obtenida en los - terminales de la termoresistencia y sumarle un nivel DC. - La amplificación se realizó mediante dos etapas, una diferencial, ya que ésta posee inmunidad al ruido, y una etapa seguidora, para permitir la calibración de la medición me-- diante un potenciómetro e invertir el signo de la señal obtenida de la primera etapa.

La configuración completa del circuito se muestra en la Figura 3.1.

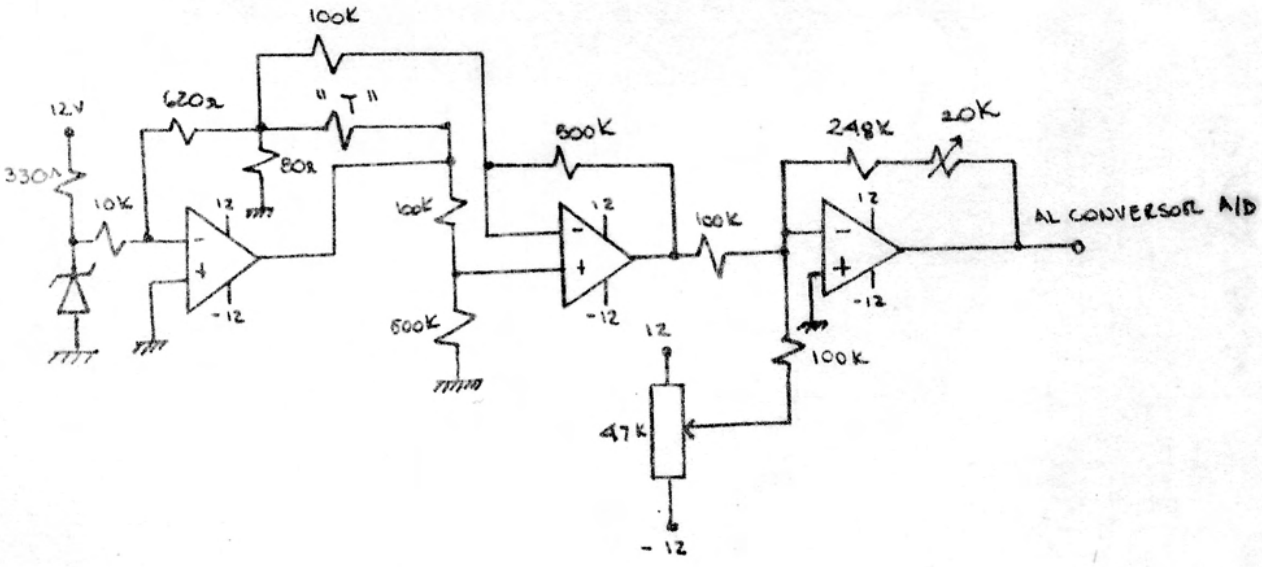



Figura 3.1.

### 3.1.2.- DETECTOR DE PRESENCIA DE LLAMA

Para detectar la presencia de la llama se utilizó una fotoresistencia ubicada cerca de ésta (Capítulo 2). Para obtener una señal compatible con el controlador utilizado se implementó el circuito mostrado en la figura 3.2.

La configuración implementada permite obtener en el terminal A de la figura 3.2. un voltaje igual a 0V cuando no hay llama y un voltaje aproximadamente igual a -0.5V cuando la llama está presente. Utilizándose este resultado para discriminar entre ausencia y presencia de llama mediante un comparador (Ver figura 3.2.). La señal obtenida a la salida del comparador se invierte y aisla del controlador mediante un optoaislador.

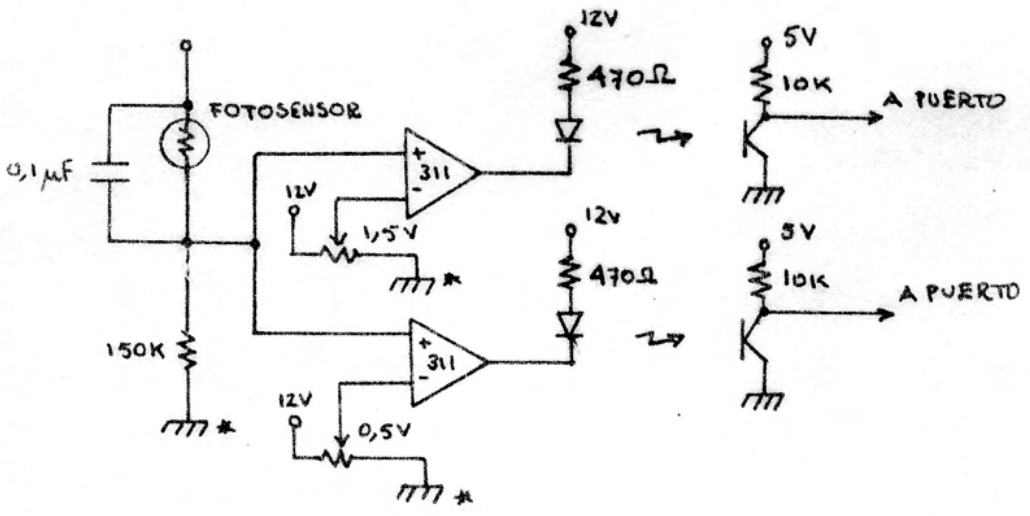



Figura 3.2.

### 3.1.3.- DETECCION DE SOBRE-PRESION

Además de poder detectar sobrepresión en la - caldera por medio del medidor de temperatura, se emplea un presostato que detecta ésta condición de emergencia. El - presostato consiste en un bulbo que contiene cierta canti-- dad de mercurio, normalmente cerrando un contacto eléctrico. Al superarse la presión prefijada, el bulbo cambia de posición, abiendo el contacto. Mediante el circuito mostrado - en la figura 3.3., se genera una interrupción al procesador, cuando hay sobre-presión. Esta interrupción puede ser enmascarada a conveniencia.

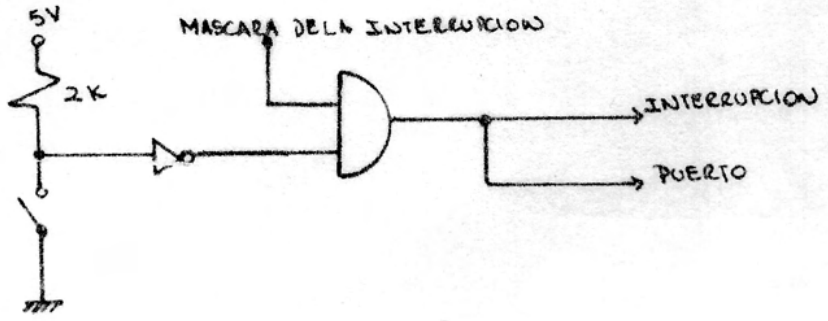



Figura 3.3.

### 3.1.4.- DETECCION DEL NIVEL DE AGUA

Para detectar el nivel de agua se implementó el circuito mostrado en la figura 3.4. en el cual el elemento transductor es una bujía de automóvil preparada.

El electrodo se alimenta con una señal cuadrada de .5KHz (para evitar polarización) y 5V de amplitud mediante una resistencia de 22KΩ. El circuito actúa como un divisor de tensión, que con el valor de resistencia empleado, al cubrirse el sensor con agua la caída de tensión es muy baja (aproximadamente 0,5V). Al estar rodeado de vapor, la caída de tensión es muy alta (aproximadamente 4,5V) debido a la resistividad del medio.

La salida de este divisor de tensión alimenta un negador CMOS, que generará un onda cuadrada o un uno lógico cuando se detecte vapor o agua respectivamente. Estas señales se traducen en "uno" o "cero" gracias a la diferencia de tiempos de conmutación del negador TTL y el CMOS.

## 3.2.- ACTUADORES

Para activar y desactivar las válvulas, el transformador de encendido y el ventilador, se diseñaron interruptores electrónicos en base a triacs (Figura 3.5.).

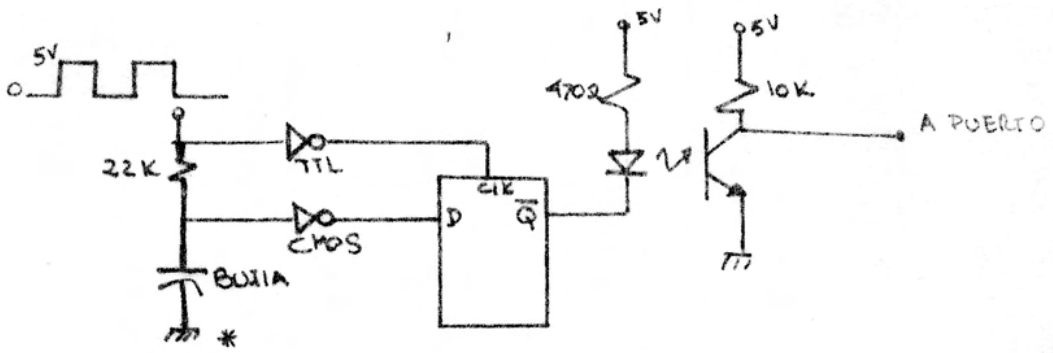



Figura 3.4.

Como se muestra en la Figura 3.5., la señal de control, la cual está sincronizada con los cruces por cero de la línea, se aisla de las señales de potencia mediante un opto-- aislador. El disparo del triac se produce por un nivel positivo de corriente aplicado al "gate".

Debido a la gran cantidad de corriente que consume el motor de la bomba de agua (40 Amper durante 4 ciclos en el arranque, decayendo luego a su valor nominal de 6.3 Amper) fue necesario implementar un circuito en base a SCR, ya que su relación costo/potencia es mucho menor que la de los - -

triacs.

La configuración empleada fue de dos SCRs conectados antiparalelos, presentándose el inconveniente de disparar - los "gates" con respecto a diferentes puntos. Para solventar esta dificultad se emplearon transformadores de pulsos, con lo que además se mantiene el aislamiento eléctrico del sistema de potencia con respecto al control (Fig. 3.6). Para esta configuración (Fig. 3.6.) los SCRs se disparan con pulsos de corriente aplicados a los "gates".

Para proteger a los SCRs y a los triacs de cambios - bruscos de voltaje se conectaron en paralelo a estos elemen tos redes "SNUBER" de R=20Ω y C=0.22μf tal como se muestra en las figuras 3.5 y 3.6.

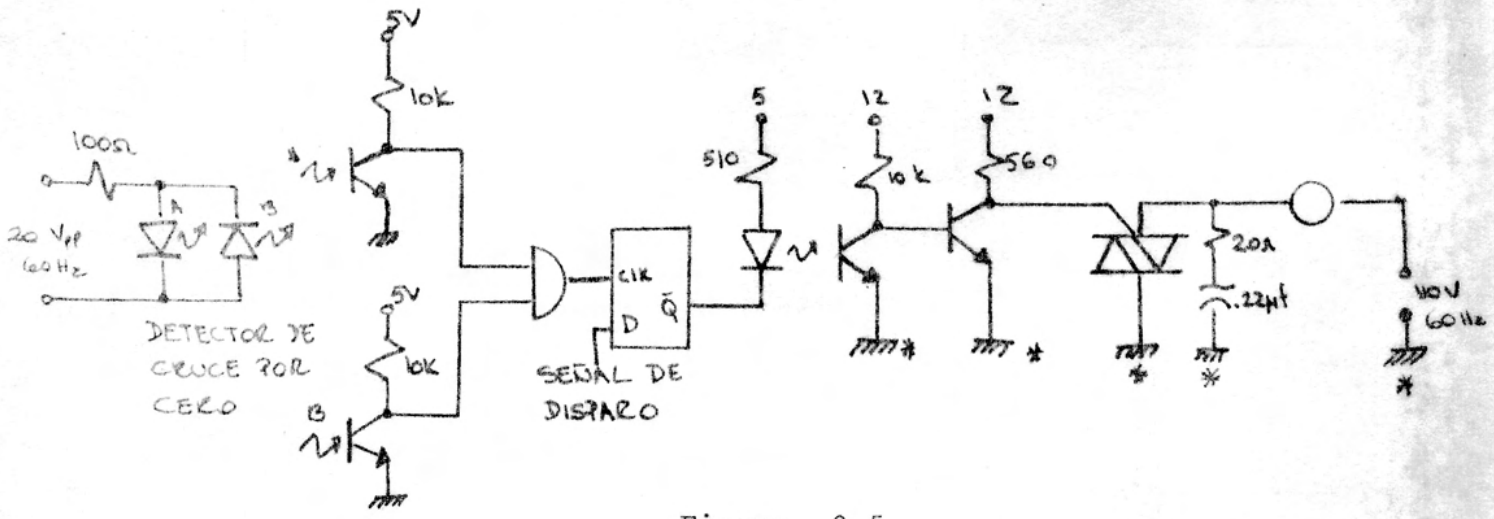



Figura 3.5.

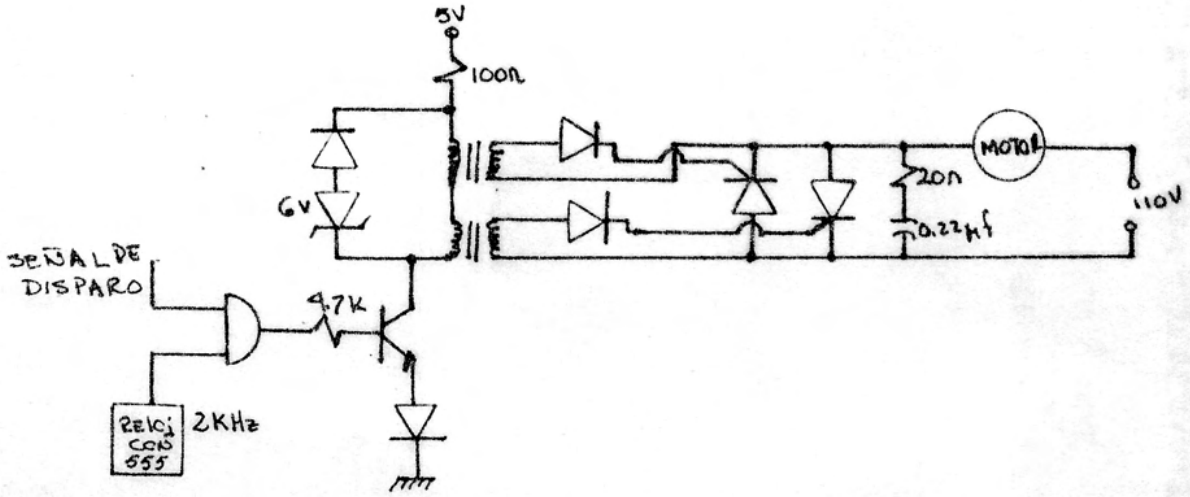



Figura 3.6.

## 3.3.- CIRCUITOS ADICIONALES AL SBC

### 3.3.1.- CONVERSOR ANALOGICO-DIGITAL

Para obtener una medida digital de la temperatura (y por lo tanto de la presión) se implementó un conversor analógico-digital de ocho "bits".

Este conversor transforma la señal analógica que entrega el circuito medidor de temperatura mediante un algoritmo de aproximaciones sucesivas. Este algoritmo consigue el valor digital más aproximado a la medida en ocho intentos, como se puede deducir del diagrama de flujo mos-

trado en la figura 3.8.

El conversor A/D trabaja en base a un conversor D/A, el cual convierte el número de ocho "bits" proveniente del puerto correspondiente (Figura 3.7) en una señal analógica que es comparada con la medida de temperatura efectuada.

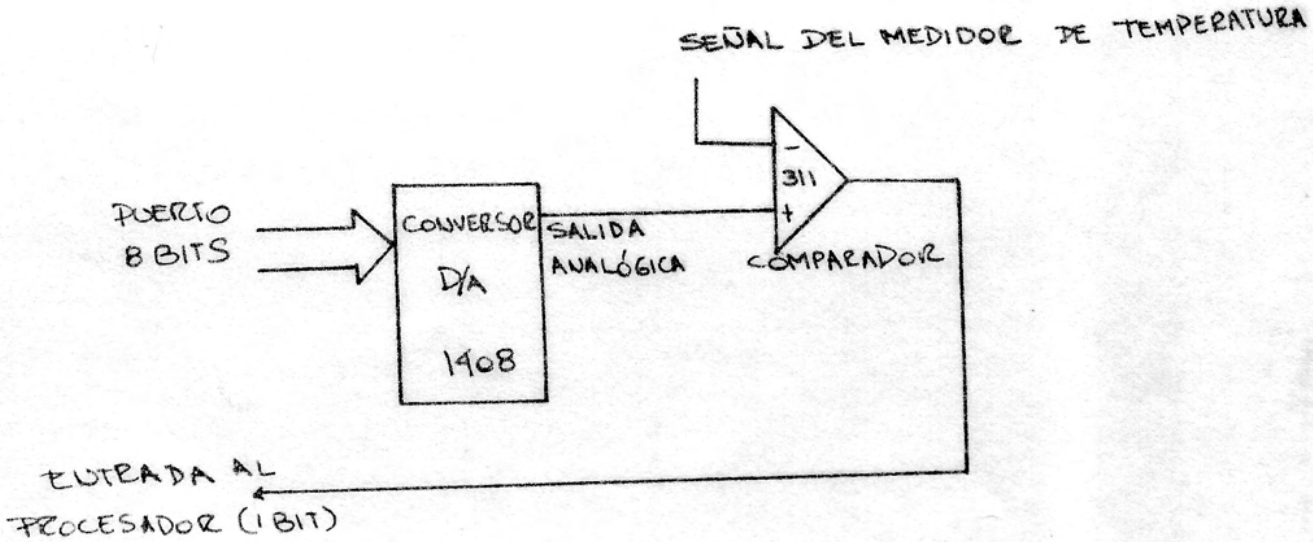



Figura 3.7.

### 3.3.2.- EXPANSION DE MEMORIA

Para soportar el "software" del controlador fue necesario espander la memoria, que suple el SBC80/10 INTEL (4K de ROM y 1K de RAM) ya que se necesitaron 9K de ROM y 1K de RAM.

Para poder utilizar el bus de direcciones y

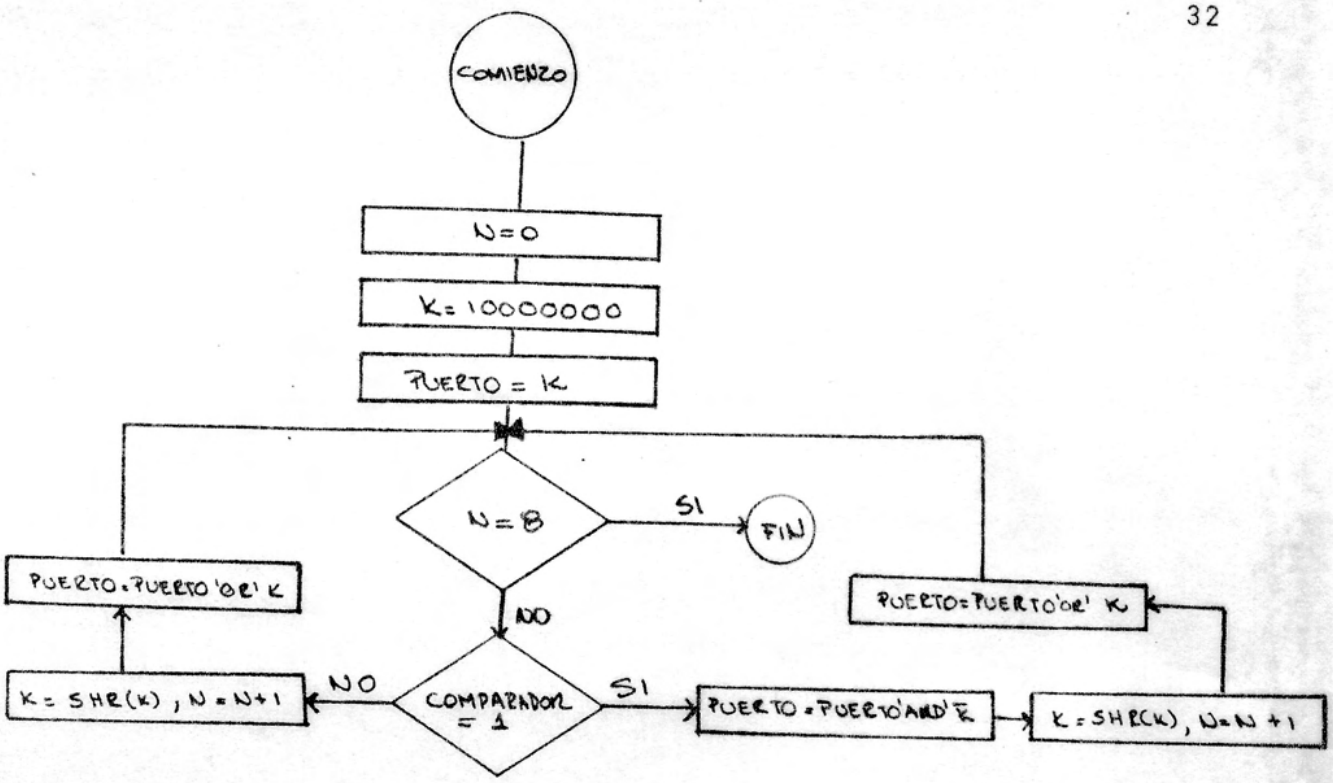



Figura 3.8.

el de datos del SBC 80/10 INTEL fue necesario conectar - "buffer three-state" y negadores a todas las líneas, y además enviar el reconocimiento ($\overline{AACK}$) al SBC a partir de los selectores de las memorias.

La expansión se realizó de forma tal que la memoria (tanto ROM como RAM) quedará contigua a la del SBC. Quedando la ROM expandida de la 1000 H a la 23FFH y la RAM de la 4000 H a la 43FFH.

### 3.3.3.- TECLADO Y VISUALIZADORES

Para proveer al controlador de la posibilidad de programas la caldera para que trabaje a presión constante, sin necesidad de un terminal CRT, se le proporcionó un teclado y visualizadores del tipo siete segmentos. Mediante el teclado se puede programar la presión e inicializar la hora; en los visualizadores se puede ver la hora y la presión a la que está trabajando la caldera.

El circuito de teclado se diseñó de forma tal de entregar al controlador el dato decodificado en cuatro bits junto a una señal de interrupción; para evitar que el microprocesador se encargara de la decodifiación.

Además se diseñó un circuito para los visualizadores donde el procesador solo enviara el dato. El circuito de visualizadores recibe el dato en BCD junto con la información que indica en que visualizador debe colocarse. Esta información la toma un decodificador que selecciona el registro deseado. Cada registro tiene un decodificador BCD a siete segmentos que maneja el visualizador numérico.

### 3.3.4.- SISTEMA DE TEMPORIZACION PARA EL RMX80

Para realizar las funciones de retardo del sistema operativo, es necesario suplir a éste de un reloj

en tiempo real que interrumpa al procesador, así como las rutinas de atención para esta interrupción, inicialización, enmascaramiento y desenmascaramiento.

El circuito utilizado se muestra en la Figura 3.9.

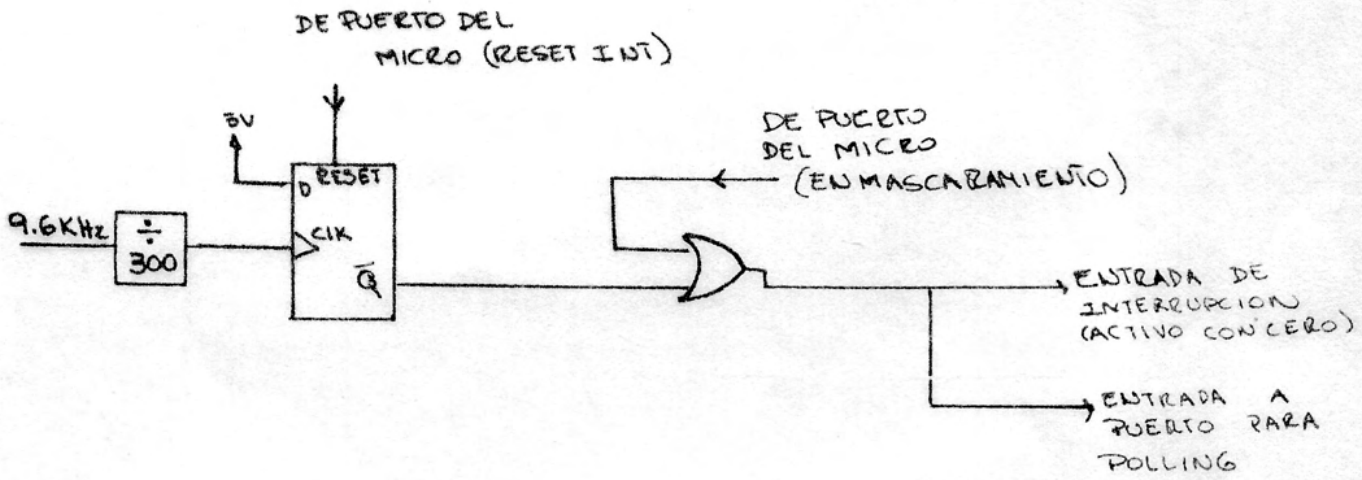



Figura 3.9.

## 3.4.- SOFTWARE

El diseño de los programas que se encargan de manejar la totalidad del proceso, basados en el RMX/80, se estructura en base al gráfico mostrado en la figura 3.10.

Los rectángulos sencillos representan los "tasks", que pueden ser considerados como programas independientes,

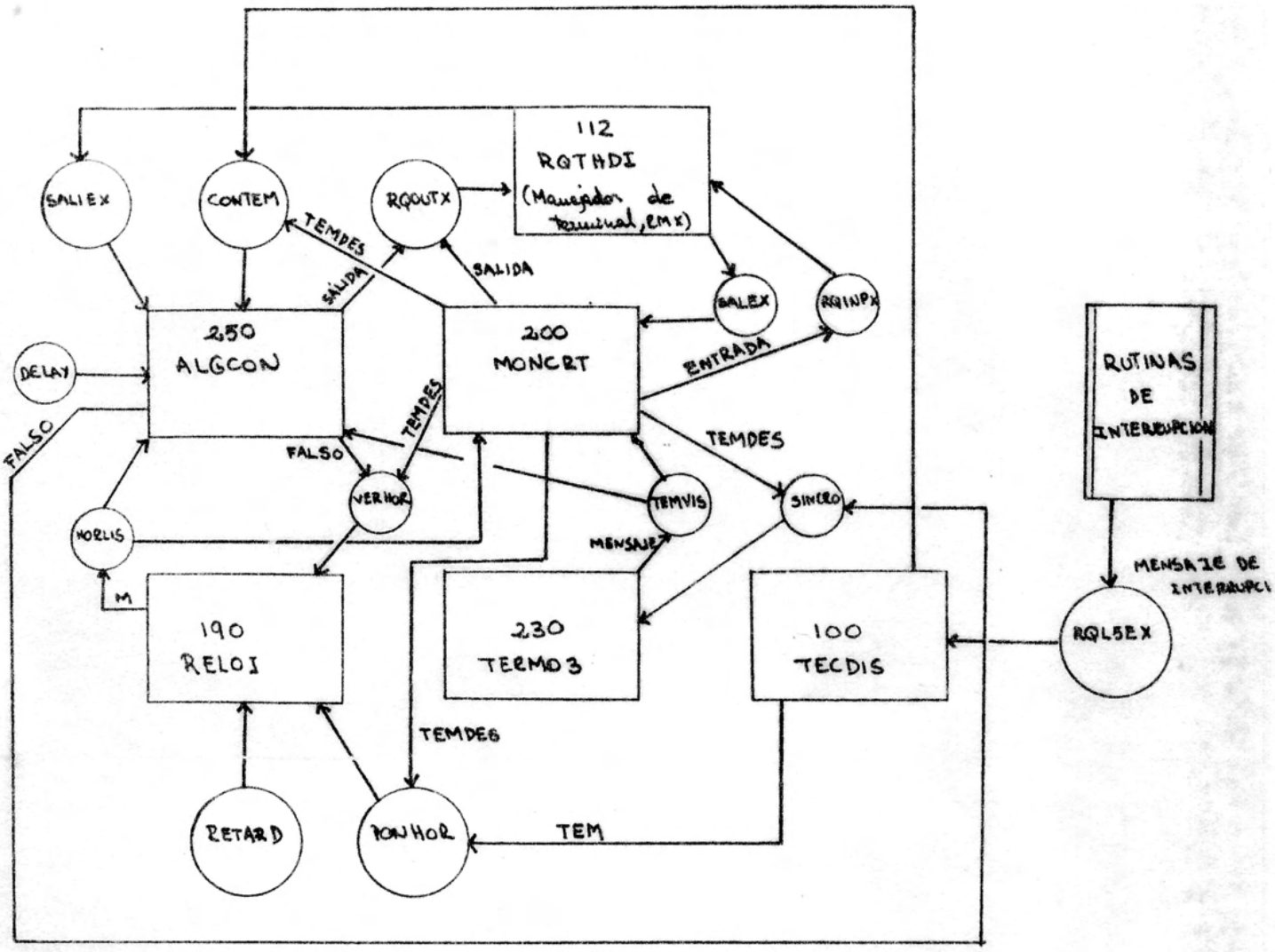



Figura 3.10.

los círculos representan los "exchanges", donde se realiza el intercambio de mensajes entre task's y las flechas representan los mensajes enviados y/o recibidos por los "tasks". El rectángulo doble representa las rutinas de interrupción que se han suplido al sistema. El manejador de terminal, - las rutinas de interrupción para atención de la USART y los

mensajes y exhanges concernientes a la comunicación entre - ese task y las rutinas, están incluídas como una utilidad del sistema operativo.

### 3.4.1.- FUNCION DE CADA TASK

En el diseño de un sistema utilizando el RMX80, deben dividirse los trabajos a realizarse en un cierto número de "tasks" de manera que cada uno opere en una forma relativamente independiente de los demas. La longitud de los - tasks viene determinada por la función que realiza, pero es conveniente que no sean demasiado extensos cuando pueden subdividirse las funciones, para evitar complejidad en la programación. Tampoco deben ser demasiado cortos con el propósito de que no se multipliquen los módulos de manera exagerada, incurriendo en el uso exagerado de memoria RAM, por la alta densidad de mensajes y exchanges que poseerá el sistema.

Debe tenerse en cuenta además la asignación de prioridades, cuidando de no asignarlas muy elevadas a los - "tasks" mas extensos para evitar un acaparamiento del procesador. Por lo tanto, los "tasks" de alta prioridad son - los que ejecutan funciones vitales pero en forma rápida; - mientras que aquellos que tienen prioridad baja, ejecutan - funciones de rutina no vitales, que pueden esperar cierto -

tiempo.

Las funciones de los módulos se especifican a continuación en orden de prioridad (recuérdese que a más bajo el número, mayor prioridad).

### 3.4.1.1.- TECDIS (PRIORIDAD 100)

Prioridades menores de 128 se asignan sólo a "tasks" de interrupción tales como TECDIS o el manejador de terminal.

Este "taks" se encarga de manejar el teclado de operación mínima de la caldera y de hacer el eco necesario en los visualizadores numéricos (display LED). Recibe un mensaje de interrupción por el "exhange" RQL5EX y envía mensajes T a CONTEM y TEM a PONHOR a los "tasks" ALGCON y RELOJ respectivamente con el objeto de comunicarse con el sistema en forma sincronizada.

La operación de este programa permite alterar el punto de operación de presión de la caldera en forma global no programada (un valor único para las 24 horas del día), cambiar la hora, arrancar el proceso e interrumpirlo cuando se desee. El "task" está estructurado por un programa principal y varios procedimientos, los cuales representaremos en la figura 3.11. con la ayuda de diagramas

de flujo. La asignación de las teclas y sus funciones son las siguientes:

"*": Cambiar la hora. (10)

"#": Cambiar la referencia de presión. (11)

"I": Interrumpir el proceso. (13)

"A": Arrancar el proceso. (15)

"REPEAT": Acepta la completación de los comandos "*" y "#".

Programa principal:

Se encarga de aceptar los comandos antes especificados y llamar a los procedimientos correspondientes. En la figura 3.11 se muestra gráficamente su funcionamiento.

Procedimientos:

PHORA: Se encarga de recibir los datos de la nueva hora, validarlos y hacer efectivo el cambio. El diagrama de flujo de este procedimiento es mostrado en la figura 3.12.

PREF: Se encarga de recibir los datos de la nueva referencia, validarlos y efectuar el cambio correspondiente. La lógica de operación es muy similar a la de PHORA, por lo cual no especificaremos su operación.

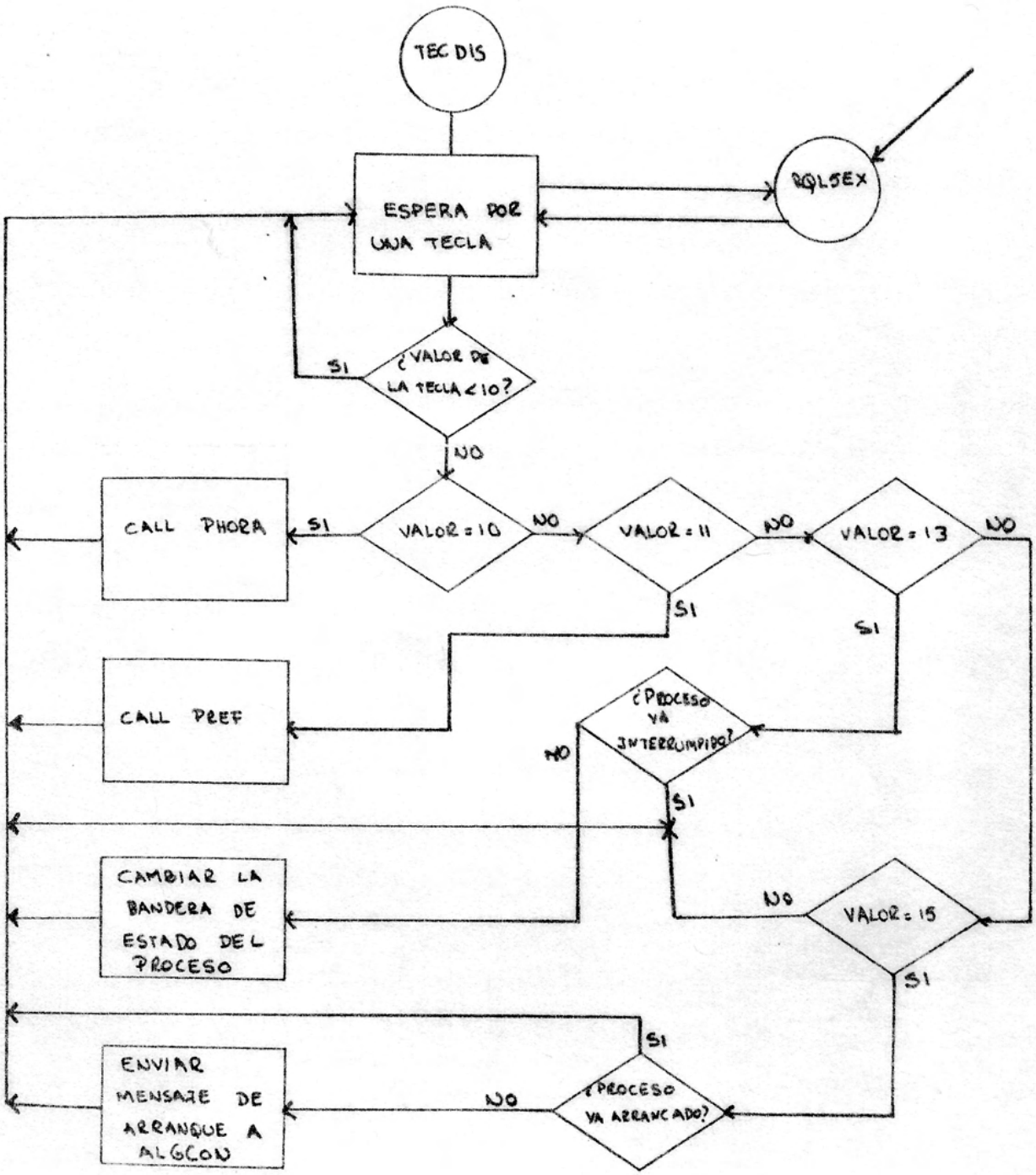

TEC DIS
RQLSEX
ESPERA POR UNA TECLA
¿VALOR DE LA TECLA < 10?
SI
NO
CALL PHORA
VALOR = 10
VALOR = 11
VALOR = 13
CALL PREF
¿PROCESO YA INTERRUMPIDO?
CAMBIAR LA BANDERA DE ESTADO DEL PROCESO
VALOR = 15
ENVIAR MENSAJE DE ARRANQUE A ALGCON
¿PROCESO YA ARRANCADO?

Figura 3.11.

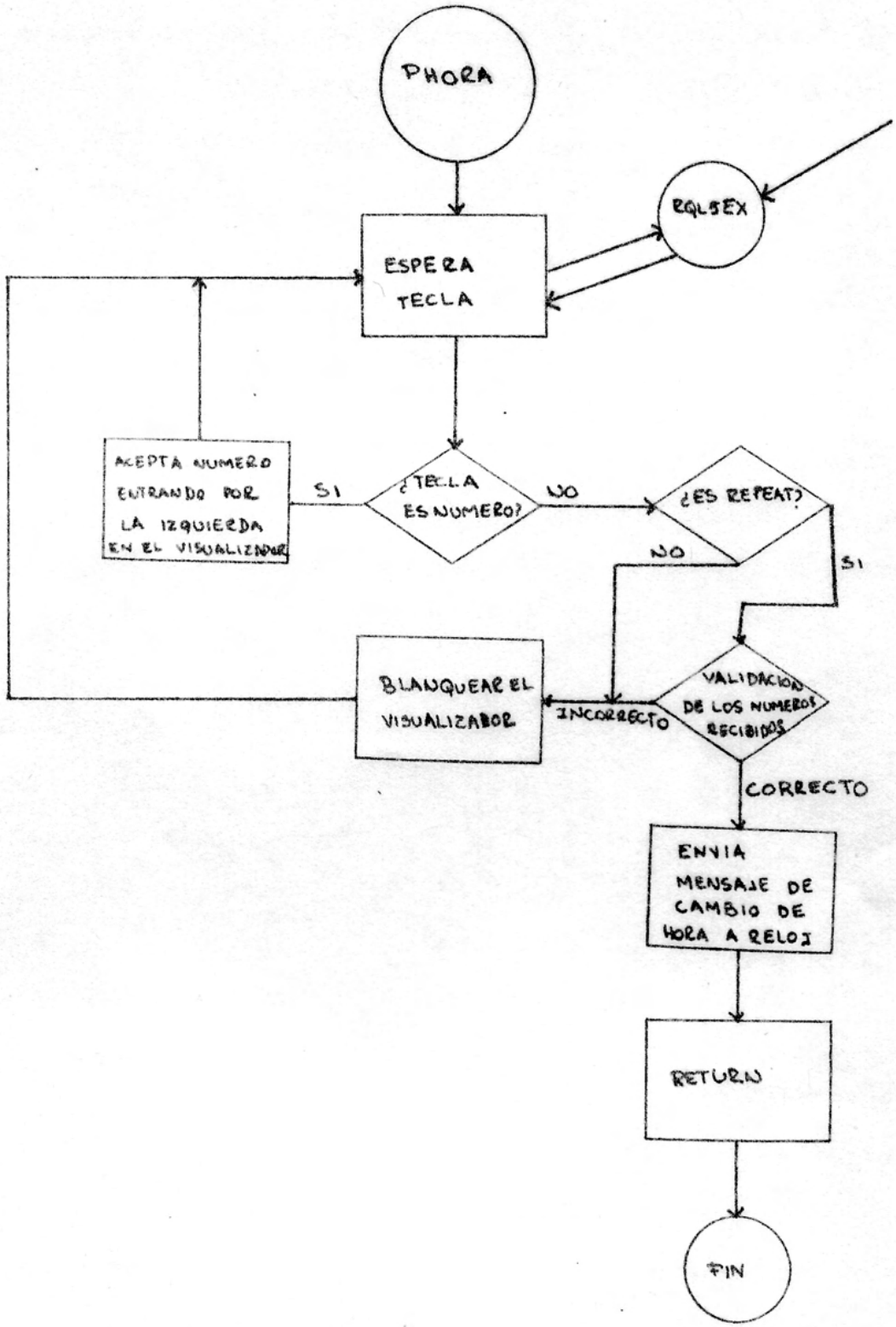

PHORA
RQL5EX
ESPERA TECLA
ACEPTA NUMERO ENTRANDO POR LA IZQUIERDA EN EL VISUALIZADOR
SI
¿TECLA ES NUMERO?
NO
¿ES REPEAT?
NO
SI
BLANQUEAR EL VISUALIZADOR
VALIDACION DE LOS NUMEROS RECIBIDOS
INCORRECTO
CORRECTO
ENVIA MENSAJE DE CAMBIO DE HORA A RELOJ
RETURN
FIN

Figura 3.12.

Procedimientos auxiliares:

PONDIS: Coloca el número específicado por el primer parámetro en el número de dígito específicado por el segundo parámetro. Para invocarlo se ejecuta un CALL PONDIS (V, D).

BLANC: Blanquea a partir del dígito específicado por el primer parámetro, el número de dígitos específicado por el segundo (CALL BLANC (N,K)).

La asignación de número de dígito se ilustra en la figura 3.13.

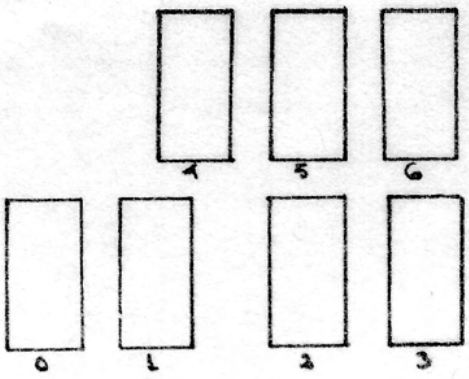



Figura 3.13.

3.4.1.2.- RQTHDI (PRIORIDAD 112):

Este "task" se suple por el RMX80 como una utilidad del sistema con el objeto de comunicarse con un terminal CRT o teletipo. Recibe mensajes de interrupción de los "exhanges" RQL7EX y RQL6EX así como mensajes de los "tasks" del usuario a través de RQINPX y RQOUTX.

La forma de comunicarse con RQTHDI es la siguiente:

- Si se desea leer algo del teclado, debe enviarse un mensaje de petición a RQINPX, especificando la dirección de un área de memoria donde se almacenarán los caracteres, el número de caracteres deseado y la dirección de un "exchange" donde se esperará por una respuesta de transferencia realizada.
- Si se desea escribir algo al CRT o teletipo, se debe enviar un mensaje a RQOUTX, especificando la dirección de una área de memoria donde se encuentra almacenada la cadena de caracteres, el número de caracteres a transferirse y la dirección de un "exchange" donde se esperará hasta la finalización de la transferencia .
- En ambos casos, al completarse la transferencia, el RQTHDI envía un mensaje al "exchange" específicado para respuesta, donde informa si hubo éxito o no y el número de caracteres efectivamente leido o escrito.

Para una información mas detallada sobre manejador de terminal, puede consultarse el manual "RMX/80 USER'S GUIDE", Capítulo 4.

### 3.4.1.3.- RELOJ (PRIORIDAD 190):

Este es el "task" de mayor prioridad entre los que no son de interrupción y su función es mantener un reloj con la hora del día previamente introducida por teclado o por terminal. Además, es "task" enseña la hora y minutos en el visualizador numérico.

Para generar el patrón de tiempo se hace esperar a RELOJ en un "exchange" cierta cantidad de unidades de tiempo del sistema (las necesarias para que sumen un segundo) donde nunca se enviará un mensaje. Al finalizar el tiempo de espera este "task" entrará a correr por ser el de mayor prioridad, ejecutará su función de actualizar el reloj, refrescará la información del visualizador y preguntará si no hay cambio de hora, para entrar nuevamente en su estado de espera.

Todo el procedimiento entre tiempo y tiempo de espera no debe durar mas de una unidad elemental del sistema (31,25 mseg) para evitar retrasos en el reloj. El diagrama de flujo del reloj se muestra en la figura 3.14.

El procedimiento PONDIS, explicado anteriomente, también se empleó en este "task", para enviar información al visualizador numérico.

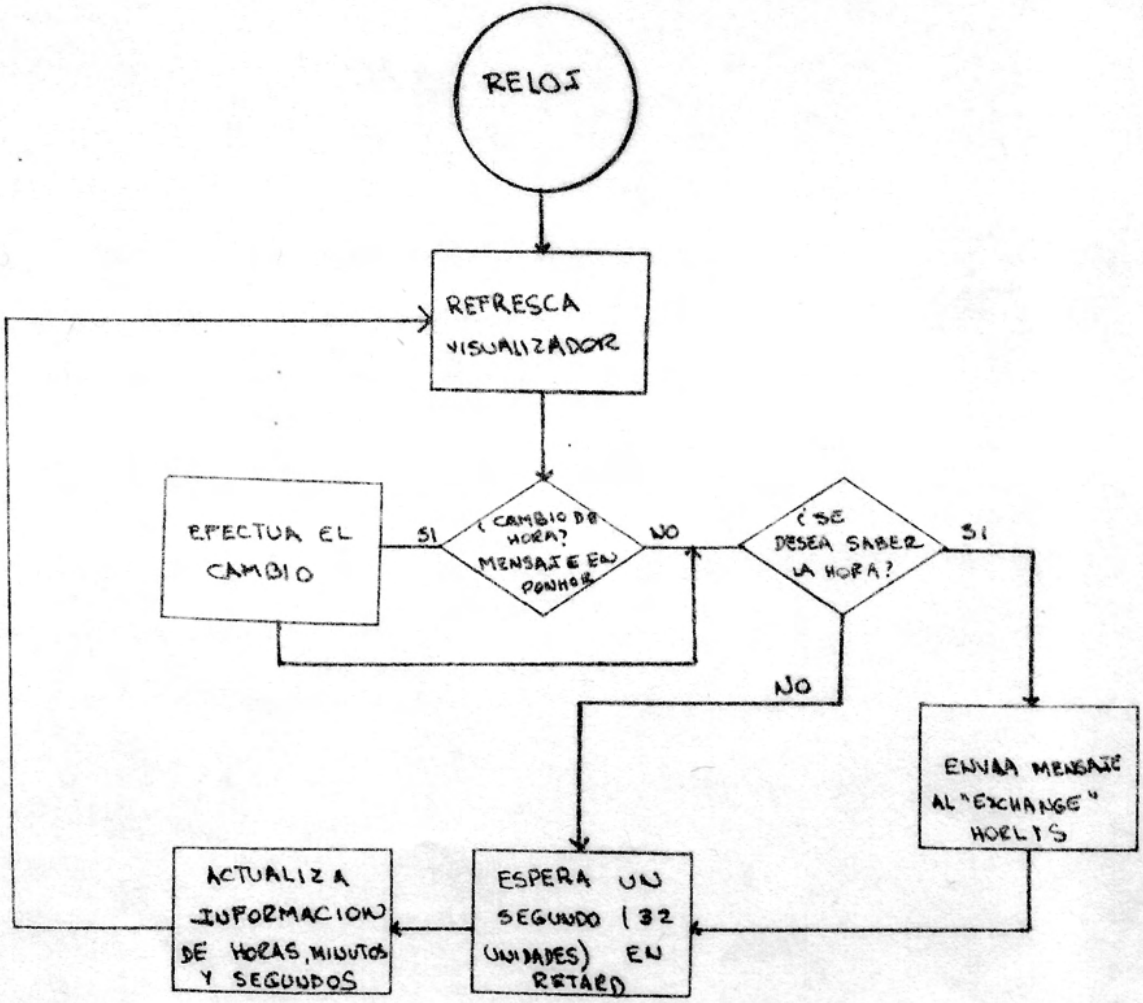



Figura 3.14.

Para cambiar la hora, hay que llenar un arreglo (estructura) público llamado H(H.H, H.M, H.S) y luego enviar un mensaje al "exhange" PONHOR. Para leer la hora, hay que enviar un mensaje al "exchange" VERHOR y esperar en HORLIS por un mensaje que contiene la hora al momento de ser solicitada.

### 3.4.1.4.- MONCRT (PRIORIDAD 200)

Este programa ("task") hace las veces de monitor para CRT de manera de establecer una comunicación mas completa entre el usuario y el controlador que aquella que se obtiene con el uso del teclado y el visualizador.

Las funciones que realiza MONCRT a través de un menú de comandos son las siguientes:

- Comando I: Interrumpe el proceso de control.
- Comando A: Arranca el proceso.
- Comando E: Enseña el estado del proceso dando información de temperatura, presión y la hora del día. Si en el momento la llama debe estar prendida, suministra información relativa a su color.
- Comando H: Permite modificar la hora del día en horas, minutos y segundos.
- Comando P: Sirve para programar el funcionamiento de la caldera durante el día, permitiendo fijar la referencia en un intervalo mínimo de media hora. El controlador entonces cada vez que analiza la referencia, lo hace en relación con la hora del día. Esto permite una utilización muy versátil del equipo.

También cuenta el comando P, con la opción de introducir una referencia fija para todo el día, tal como si se programara con el teclado.

- Comando M: Enseña en la pantalla un mensaje de información sobre la manera de utilizar los comandos de operación (Menú).

- Comando G: Permite mostrar en la pantalla de un terminal gráfico TEK-4006-1 una figura con el programa almacenado para la operación de la caldera. El gráfico enseña la presión deseada vs. la hora del día.

- Comando V: Cuando el proceso es arrancado, cada media hora se toma una muestra de la presión de la caldera y se almacena en un arreglo que se enseñará en forma gráfica en el terminal antes mencionado, con el mismo formato que el del comando R.

Este programa también genera los diversos mensajes de error que debe escribirse en pantalla cuando se introducen los datos o cuando se pide información al sistema (errores de sintaxis y datos no válidos) en forma equivocada.

Para enseñar la temperatura, este "task" envía un mensaje llamado TEMDES (que se utiliza para varios propósitos) al "exchanges" SINCRO para arrancar el "task" de conversión analógica a digital y espera en TEMVIS por el resultado de la conversión. Para arrancar el proceso envía TEMDES al -

"exchange" CONTEM y para comunicarse con el CRT se envían mensajes (SALIDA y ENTRADA) a RQOUTX y RQINPX esperando en - SALEX hasta la completación de la transferencia. Con el objeto de poner el reloj en hora, se envía el mensaje TEMDES a PONHOR y para recibir el valor de la hora, se remueve un mensaje de HORLIS donde se suministra la información deseada.

Programa principal:

A cada función especificada en el menú corresponde un procedimiento donde se lleva a cabo la acción deseada, además hay procedimientos auxiliares para facilitar la programación tales como de conversión de ASCII a binario y viceversa, rutinas de salida para el terminal y rutinas de error para cada caso. Figura 3.15.

Procedimientos

PONE$HORA: Se encarga de recibir y analizar los datos para el cambio de hora y ejecutarlo. Genera los mensajes de error que correspondan.

PROGRAMA: Se encarga de recibir y validar los datos para programar la referencia y si son correctos el programa se ajusta a los datos suministrados por el usuario.
Se generan también los mensajes de error correspondientes.

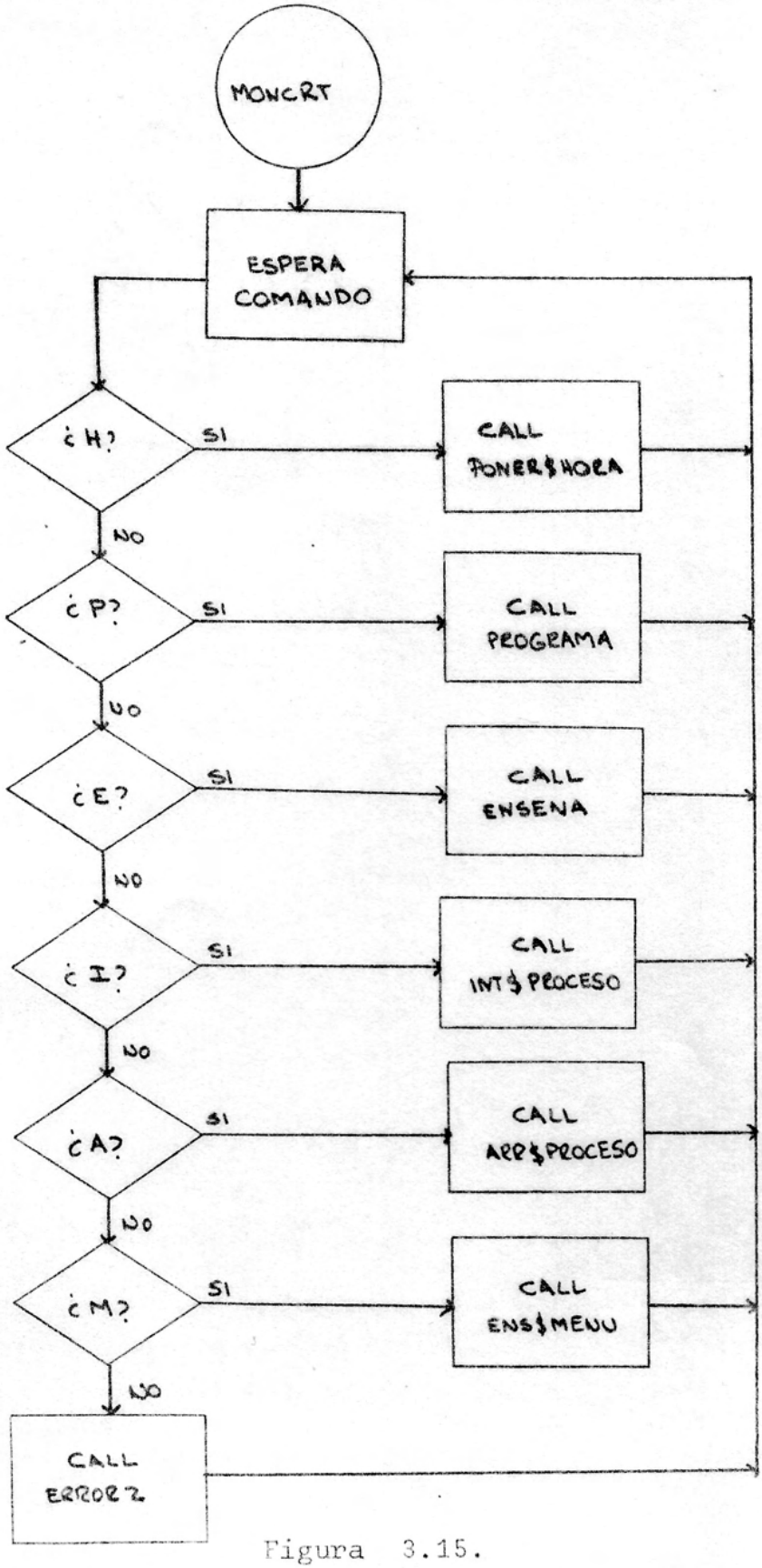



Figura 3.15.

Estas operaciones se logran haciendo uso de un proce dimiento llamado MENU.

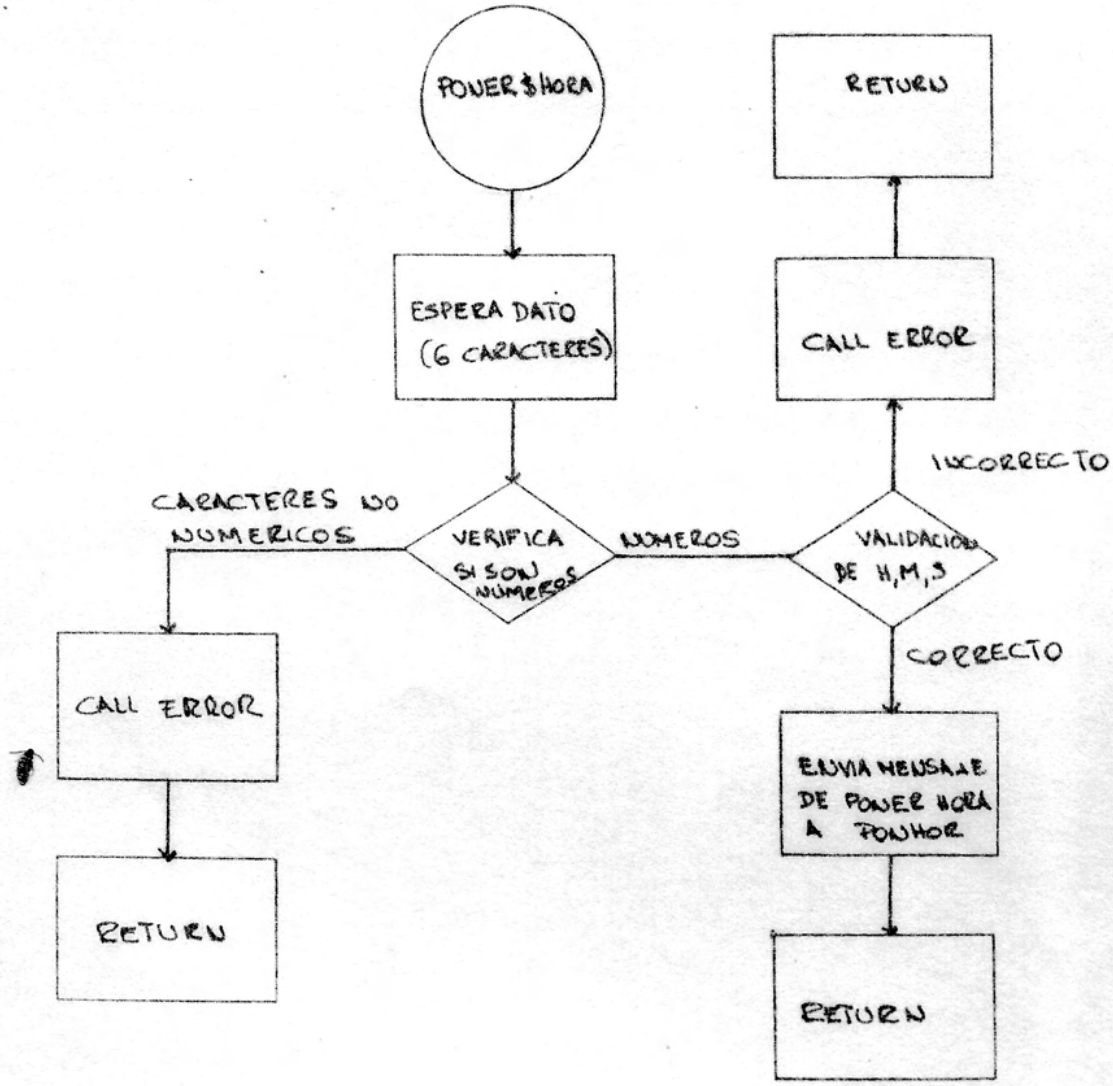



Figura 3.16.

ENSENA: Se encarga de enseñar el estado de la caldera en la pantalla del CRT.

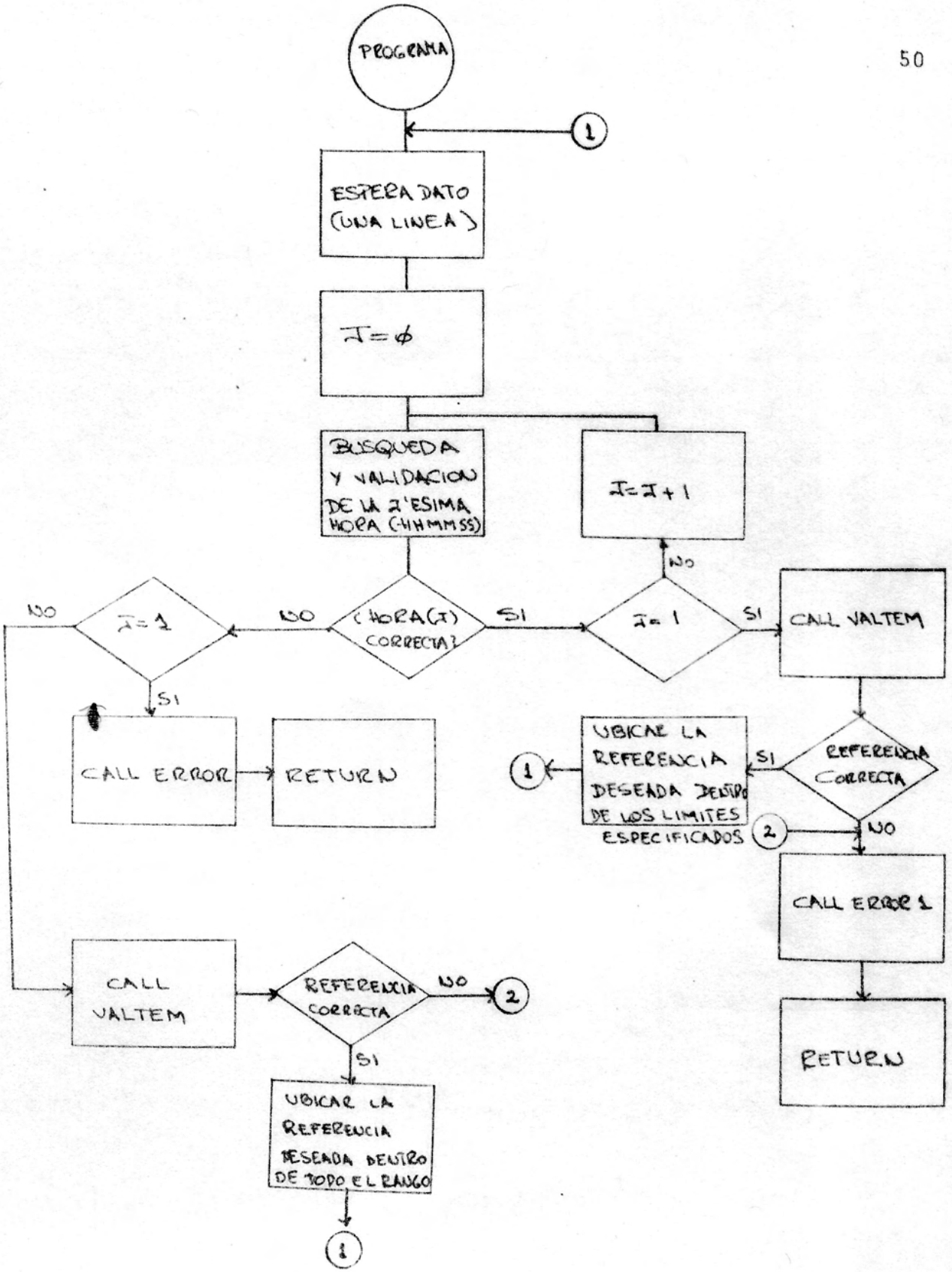

PROGRAMA
1
ESPERA DATO (UNA LINEA)
I=ϕ
BUSQUEDA Y VALIDACION DE LA I'ESIMA HORA (-HHMMSS)
I=I+1
NO
(HORA(I) CORRECTA?
NO
SI
I=1
NO
SI
I=1
SI
CALL VALTEM
CALL ERROR
RETURN
UBICAR LA REFERENCIA DESEADA DENTRO DE LOS LIMITES ESPECIFICADOS
1
SI
REFERENCIA CORRECTA
2
NO
CALL ERROR 1
RETURN
CALL VALTEM
REFERENCIA CORRECTA
NO
2
SI
UBICAR LA REFERENCIA DESEADA DENTRO DE TODO EL RANGO
1

Figura 3.17.

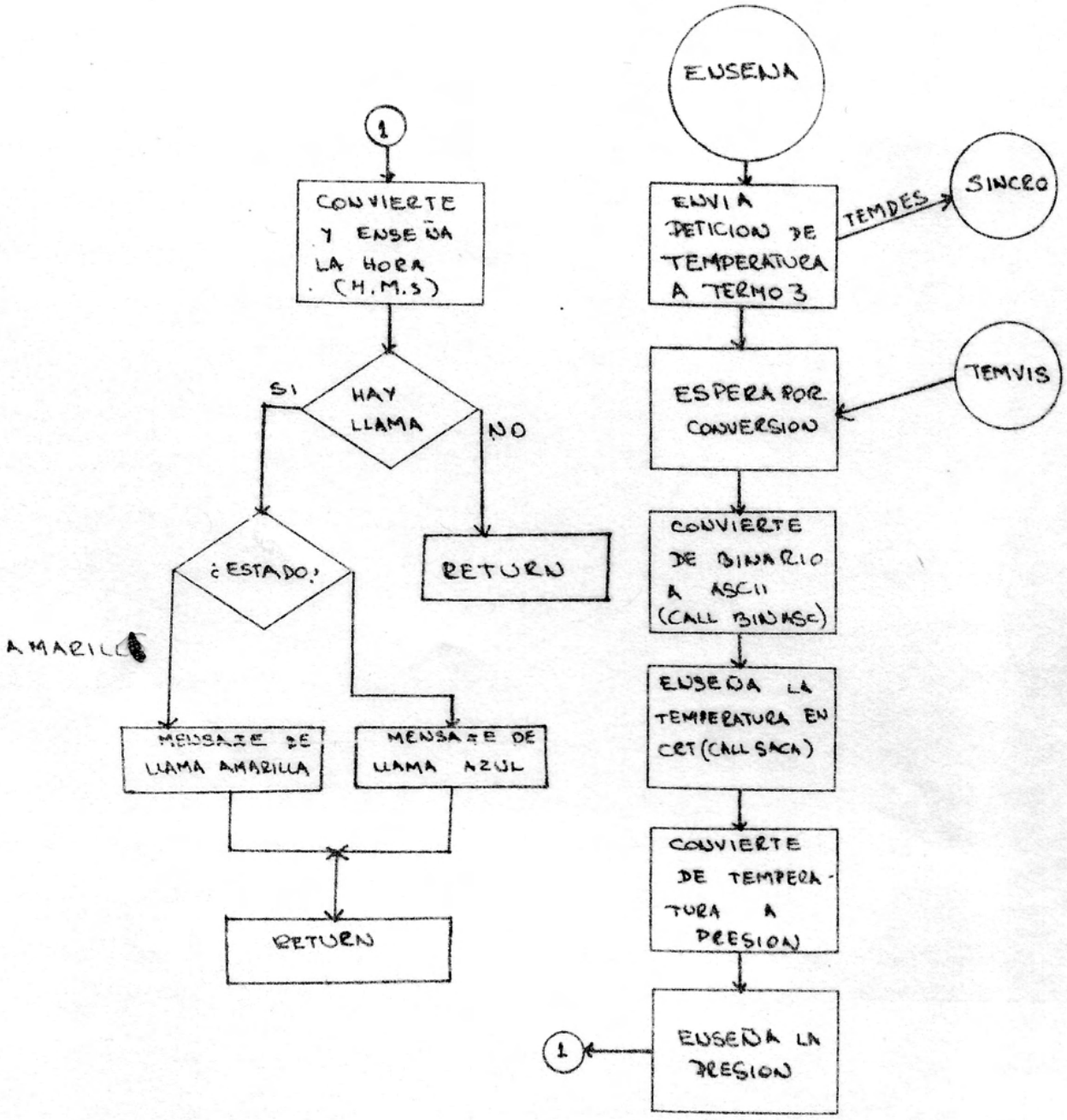



Figura 3.18.

INT$PROCESO: Interrumpe el controlador alterando la palabra de estado del proceso (FF=ØFFH).

Si la palabra de estado ya fue alterada se envía mensaje de error.

ARR$PROCESO: Se arranca el proceso, enviando un mensaje (TEMDES) al "task" de control por medio de CONTEM, esto hace que el proceso comience. Si la palabra de estado del proceso indica que éste ya fue arrancado, se envía un mensaje de error.

ENS$MENU: Este procedimiento se limita a enseñar un texto en el CRT que contiene el menú de comandos.

LIMPIA: Este procedimiento ordena al CRT Tektronix el borrado de la pantalla e introduce un retardo para esperar por la estabilización del proceso.

SALTA: Permite entrar en modo gráfico. También se utiliza para (dentro del modo gráfico) obtener discontinuidades o saltos en el trazo continuo que se dibuja en la pantalla.

MODALF: Con esta rutina se ordena al terminal entrar en modo alfanumérico.

GRAFIC: Se obtiene la presentación en la pantalla del punto definido por los parámetros del procedimiento (X, Y), que son las coordenadas horizontal y vertical respectivamente. Los valores X e Y no deben ser mayores de

1024.

Al invocarse al procedimiento GRAFIC, se posiciona un cursor en la coordenada especificada, y cada vez que se la vuelva a llamar, se unirán los puntos con trazos contínuos. Si se desea interrumpir la unión de puntos, basta - con llamar al procedimiento SALTA.

EJES: Este procedimiento dibuja sobre la pantalla los ejes horizontal y vertical, calibrados en horas (48 divisiones) y P.S.I. (16 divisiones).

GRAFBU: Permite graficar sobre los ejes dibujados en pantalla, un arreglo de 48 elementos, cuyo valor máximo puede ser hasta 160, en forma escalonada.

Procedimientos auxiliares:

SACA: Enseña en el CRT un área d e memoria cuya dirección está apuntada por el primer parámetro y el número de caracteres viene dado por el segundo (CALL SACA (DIR,K)).

ASCBIN: Ejecuta la conversión de un cadena de caracteres ASCII a binario almacenados en forma contigua a partir de la dirección apuntada por el primer parámetro. El número de caracteres a convertir viene dado por el segun

do parámetro y el resultado de la conversión se devuelve asociado con el nombre del procedimiento (siendo del tipo ADDRESS) la forma de invocarlo es: X=ASCBIN(AP$DATO, N).

BINASC: Convierte un número binario cuyo valor lo da el primer parámetro a una secuencia contigua de caracteres ASCII a almacenarse a partir de la dirección apuntada por el segundo parámetro. El número de dígitos esperado se especifica en el tercer parámetro (CAN BINASC (DATO, AP$RESULTADO, N).

MUESTR: Enseña en el CRT el número binario pasado en su único parámetro. Se asume que no resultan mas de dos dígitos decimales (CALL MUESTR(V)).

RUTINAS DE ERROR: Cada una enseña el mensaje de error correspondiente al caso en que fue llamada.

RUTINAS ESPECIALES: Se ubicaran en este módulo las dos rutinas de manejo del reloj de tiempo real que necesita el RMX/80 para arrancarlo y detenerlo cuando éste así lo requiera

ROSTRC: Se encarga de hacer un limpiado del Flip-Flop del reloj y desenmascarar éste, permitiendo que interrumpa, mediante la salida al puerto correspondiente (ØE8H).

RQSTPC: Sirve para enmascarar la operación del reloj de tiempo real, evitando que interrumpa. Estas dos rutinas no poseen ninguna relación con el "task" MONCRT, sólo que el sistema - operativo las necesita para su funcionamiento con esos mismos nombres, por lo tanto se declaran allí públicos.

### 3.4.1.5.- TERMO3 (PRIORIDAD 230)

El "task" TERMO3 se encarga de realizar la conversión analógica/digital para obtener el valor de temperatura en el interior de la caldera mediante un algoritmo de aproximaciones sucesivas que maneja un conversor D/A, tal como se ha específicado en la sección correspondiente de este capítulo. El "task" espera por un mensaje de petición de conversión durante 0,3 segundos, si no se - recibió ningún mensaje en el "exchange" SINCRO, se efectúa la conversión pero sólo para refrescar la información de - presión del visualizador numérico. Si se recibió el mensaje, se efectúa la conversión y se envía (al estar lista) - un mensaje (MENSAJE) a TEMVIS, donde el "task" que ha pedido la información lo retirará.

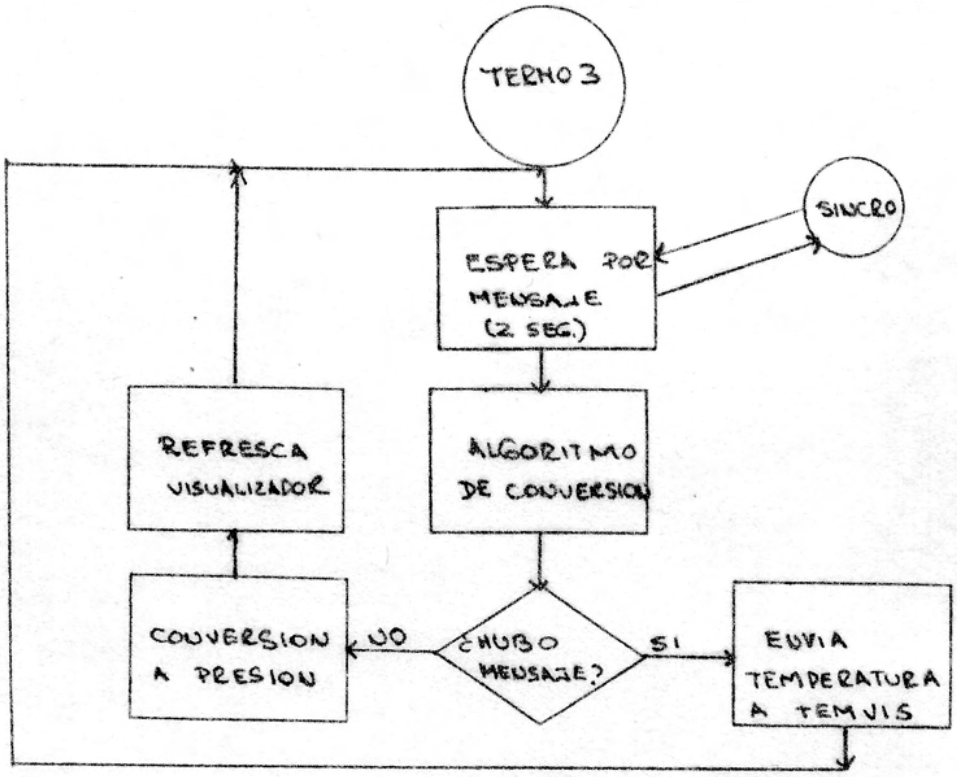



Figura 3.19.

### 3.4.1.6.- ALGCON (PRIORIDAD 250)

El "task" ALGCON es el que se encarga directamente del manejo de la caldera para su control y supervisión mediante las siguientes funciones:

- Se efectúan las mediciones y comparaciones directamente ligadas con el algoritmo de control (explicado en el Capítulo anterior) y se ejecuta dicho algoritmo, manejando los puertos que corresponden a válvulas, bomba de agua, ventilador y transformador de encendido.

- Se hace el encendido del gas en el momento oportuno, verificándose que efectivamente se en

cienda y se supervisa, mientras deba estar encendida, que - realmente así sea, si no, se detiene el proceso y se envía un mensaje de error.

- Se efectúa el llenado de agua de la caldera y se habilitan las interrupciones de los niveles extremos de nivel de agua en los momentos apropiados (en general, siempre que el proceso esté arrancado).

Los niveles intermedios de agua se revisan constantemente para efectuar el llenado inmediatamente que haga falta.

Programa principal:

El "task" ALGCON se encuentra inicialmente en espera de un mensaje de arranque del proceso en el "exchange" - CONTEM. Inmediatamente después que éste ha arrancado, se pone a cero la palabra de estado (FF) del proceso, lo cual permite la ejecución del algoritmo; si se produce una situación de peligro que amerite la interrupción del proceso o si el usuario lo desea, la palabra de estado se altera, el "task" al final de ejecución del algoritmo pregunta por ésta y continua la ejecución o vuelve al estado inicial dependiendo del caso. Si el proceso es interrumpido, así se informa al usuario a través del CRT añadiendo el mensaje de error correspondiente a la interrupción detectada o el caso

de emergencia detectado.

La información de temperatura que necesita ALGCON, la obtiene enviando un mensaje llamado FALSO (que no contiene información pues solo sirve para arrancar la conversión A/D) al "exchange" SINCRO y esperando por una respuesta en TEMVIS. El mensaje removido de este exchange contiene la información deseada.

Una vez obtenida la temperatura del interior de la caldera, se hace la conversión a presión por medio de la tabla correspondiente. Para obtener la referencia, es necesario conocer la hora para direccionar la tabla que contiene la programación del día. Con el objeto de obtener la hora, se envía un mensaje de sincronización (FALSO) a VERHOR y se espera por el mensaje de respuesta (que contiene la hora) en HORLIS.

Con el objeto de obtener retardos del tiempo aprovechando la temporización del sistema operativo se espera las unidades de tiempo deseadas en el "exchange" DELAY, donde nunca se enviará un mensaje.

Para enviar mensajes de error al CRT, se envía un mensaje (SALIDA) a RQOUTX y se espera la finalización de la transferencia en SAL1EX.

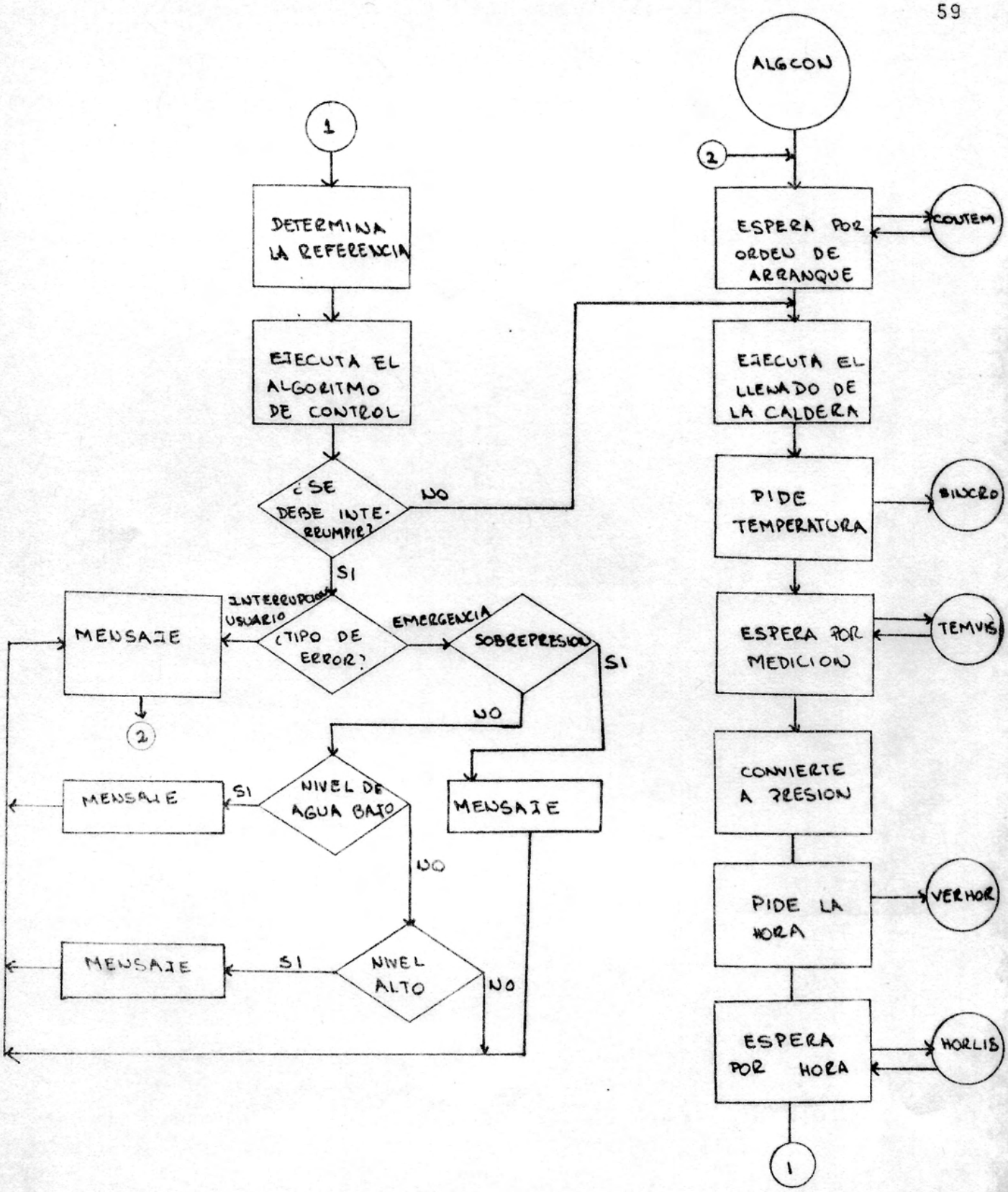

ALGCON
1
2
DETERMINA LA REFERENCIA
EJECUTA EL ALGORITMO DE CONTROL
¿SE DEBE INTERRUMPIR?
NO
SI
INTERRUPCION USUARIO
EMERGENCIA
MENSAJE
¿TIPO DE ERROR?
SOBREPRESION
SI
NO
2
MENSAJE
SI
NIVEL DE AGUA BAJO
MENSAJE
NO
MENSAJE
SI
NIVEL ALTO
NO
ESPERA POR ORDEN DE ARRANQUE
COUTEM
EJECUTA EL LLENADO DE LA CALDERA
PIDE TEMPERATURA
SINCRO
ESPERA POR MEDICION
TEMUIS
CONVIERTE A PRESION
PIDE LA HORA
VERHOR
ESPERA POR HORA
HORLIS
1

Figura 3.20.

Procedimientos:

DIS: Al ser llamado, se encarga de desabilitar las interrupciones de nivel de agua, enmascarando estas interrupciones con una salida de puerto y con una bandera (por software).

EN: Este procedimiento hace lo contrario que el anterior, es decir, habilita las interrupciones de nivel de agua. La necesidad de crear estos procedimientos fue que por efecto del ruido al encender el transformador de alta tensión se producían interrupciones fantasma que detenían la ejecución del proceso.

ENCIEN: Este procedimiento sirve para encender el generador de la caldera en el nivel de llama especificado por el parámetro (CALL ENCIEN (V)), cuando así se requiera. Durante su ejecución se desabilitan las interrupciones de nivel de agua para evitar problemas de ruido. Si el nivel de llama que corresponde encenderse es el menor, se detiene la marcha del ventilador ubicado en la chimenea y se procede al encendido. Al lograrse el encendido se pone de nuevo en funcionamiento el ventilador.

En caso de que no se logre encender el quemador, se interrumpe el proceso y se da el mensaje de error correspon

diente. Es de hacer notar que cada vez que se recorre el lazo del algoritmo de control se inspecciona el estado de la llama: si ésta debe estar encendida y no se detecta, se interrumpe el proceso y se envía un mensaje de error; si estando encendida la llama es amarilla o azul, se mantiene esta información para uso del comando E del monitor al igual que si la llama se encuentra apagada cuando debe estarlo.

La Figura 3.2.1. visualiza la operación de éste procedimiento.

### 3.4.2.- RUTINAS DE INTERRUPCION

Como el SBC 80/10 solo permite un nivel de interrupción, todas las distinciones entre niveles adicionales debe hacerse muestreando ("Polling") entre las posibles interrupciones hasta determinar cuál es la que debe ser atendida. Este mecanismo introduce un esquema de prioridades que depende de el orden con que se muestrean las posibles interrupciones.

Con el sistema operativo RMX/80, el manejo de interrupciones no es directo. Para su implementación debe informase por medio de llamadas al procedimiento RQSETP, cuales son las rutinas de interrupción y cuál es el nivel de prioridad asociado con cada una (se permiten ocho niveles de

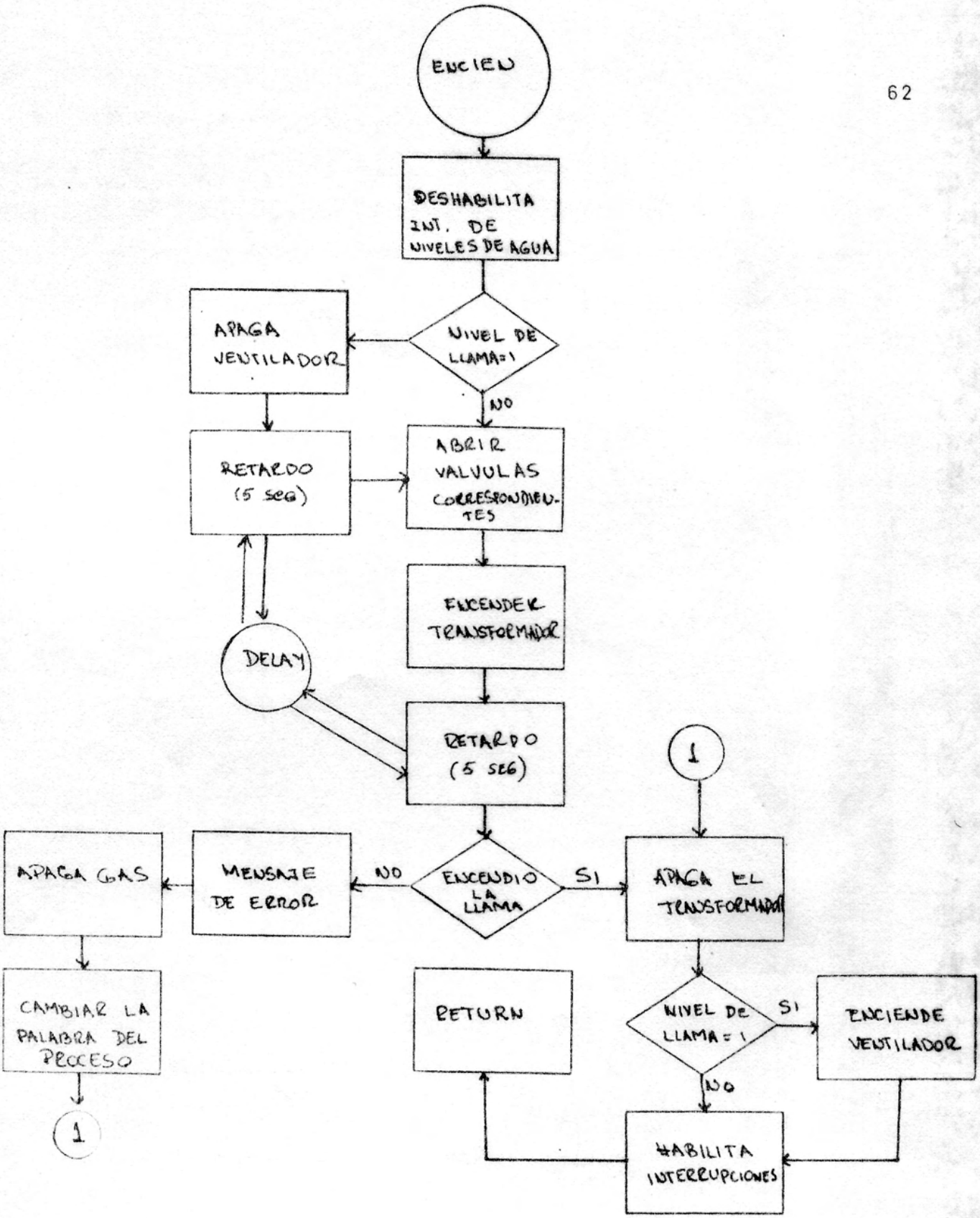

ENCIEN
DESHABILITA INT. DE NIVELES DE AGUA
NIVEL DE LLAMA=1
APAGA VENTILADOR
NO
RETARDO (5 seg)
ABRIR VALVULAS CORRESPONDIENTES
ENCENDER TRANSFORMADOR
DELAY
RETARDO (5 SEG)
1
APAGA GAS
MENSAJE DE ERROR
NO
ENCENDIO LA LLAMA
SI
APAGA EL TRANSFORMADOR
CAMBIAR LA PALABRA DEL PROCESO
RETURN
NIVEL DE LLAMA = 1
SI
ENCIENDE VENTILADOR
NO
1
HABILITA INTERRUPCIONES

Figura 3.21.

interrupción). En el procedimiento RQINTI se debe inicializar toda la lógica de manejo de interrupciones (programación de puertos, enmascaramiento y desenmascaramiento e inicialización de banderas) pues es llamado por el RMX80 estando las interrupciones desabilitadas cuando se estan configurando las tablas dinámicas del sistema.

Para informar de las rutinas de interrupción, se llama un procedimiento del sistema operativo (CALL RQSETP (DIR, N)) supliendo como primer parámetro la dirección de la rutina y como segundo el nivel de prioridad correspondiente. Las rutinas de atención de interrupción deben realizar las siguientes funciones:

- Verificar si la interrupción debe ser atendida por esa rutina.
- Si se debe atender, se pueden tomar las acciones necesarias en la propia rutina o enviar un mensaje de interrupción al "exchange" de interrupción correspondiente al nivel de prioridad (RQLØEX aRQ17EX), en ambos casos se debe retornar con ØFFH para indicar que la interrupción fue atendida.
- Si no se debe atender, simplemente no se toman acciones y se retorna con Ø para indicar que la interupción no corresponde con ese nivel.

### 3.4.2.1.- RQTPOL (NIVEL DE PRIORIDAD 1)

Esta rutina se encarga de manejar - la interrupción del reloj de tiempo real para temporización del sistema operativo. Sus funciones son:

- Si la verificación de interrupción es positiva, se limpia el "flip-flop" del reloj (ver sección de "hardware. en este capítulo) y se llama una rutina del - sistema operativo (CALL RQCTCK) que se encarga de actualizar los retardos de los "tasks" que se encuentran en estado de espera y toman las acciones necesarias cuando se ha cumplido la totalidad del tiempo de espera de algun "task".

- Si la verifiación es negativa, no se toma ninguna acción.

### 3.4.2.2.- NALPOL (NIVEL DE PRIORIDAD 2)

Este procedimiento atiende la interrupción correspondiente al nivel extremo superior de agua de la caldera.

Una vez que se ha determinado que - efectivamente el nivel superior de agua ha interrumpido, se espera un tiempo de 2,3 mseg. (Un período aproximado de la onda cuadrada que se utiliza para medir los niveles de agua) y se pregunta de nuevo si la interrupción permanece. Si la doble verificación es positiva (esta verificación reduce la

probabilidad de error por efecto de ruido) se ordena interrumpir el proceso y se actualiza una bandera para generar el mensaje de error, si no, no se toman acciones, pero se considera la interrupción atendida.

### 3.4.2.3.- NBAPOL (NIVEL DE PRIORIDAD 3)

NBAPOL es exactamente igual que el anterior, pero la interrupción corresponde al nivel extremo inferior de agua.

### 3.4.2.4- SPRPOL (NIVEL DE PRIORIDAD 4)

Se encarga de atender la interrupción correspondiente a sobrepresión, determinada por un interruptor de mercurio ubicado en la caldera como norma de seguridad. Si la interrupción es válida, se ordena detener el proceso y se hace generar un mensaje de error en el CRT.

### 3.4.2.5.- TECPOL (NIVEL DE PRIORIDAD 5)

Se encarga de manejar el teclado de operación mínima de la caldera. Esta interrupción es la única que no se maneja directamente por la rutina de atención, sino a través de un "task" de interrupción ya explicado. Son estas las dos formas de atender interrupciones en el sistema operativo RMX/80: atención directa y "task" de interrupción.[4]

Al verificarse que el teclado ha interrumpido, se lee el dato escrito y se limpia el flip-flops interrupción de teclado (ver sección de hardware en este capítulo), seguidamente, se envía un mensaje de interrupción a través de - - RQL5EX y se retorna con ØFFH para indicar que la interrupción fue atendida.

### 3.4.3.- MODULO DE CONFIGURACION

Cuando se implementa un sistema basado en el sistema operativo RMX/80, el último paso es definir el módulo de configuración, donde se definen todas las tablas estáticas de los tasks y se definen las direcciones de los "exchanges" del sistema. Para realizar este módulo es preciso haber definido totalmente la estructura del "software", haber asignado las prioridades de los "tasks" y conocer el - "stack" requerido para cada uno. El "stack" necesario para cada "task" es el definido por el compilador de PLM (más - un 10% recomendado en el manual del RMX/80) además de 24 - bytes que siempre requiere el RMX/80.(5)

Una vez definidos los parámetros mencionados se escribe el módulo de configuración con ayuda de macros que suple el sistema RMX/80. El procedimiento es el siguiente:

- Se definen los descriptores estáticos de "tasks" ("static taste descriptor") llamando a la macro STD

con el nombre del "task", la longitud del "stack" y la prioridad como parámetros.

- Se definen las tablas iniciales de "exchanges" invocando a la macro XCHADR con el nombre de "exchange" como parámetro.

- Localizar espacio RAM para los descriptores de los "tasks" (dinámicos) que se generarán cuando el sistema se inicialice. Esto se logra llamando a la macro GENTD.

- Construir la tabla de creación, la cual posee información del número de "tasks" y "exchanges" y las direcciones de las tablas estáticas.

- Se deben definir las variables públicas de uso del sistema, como RQRATE que determina la rata de baudios de operación del terminal y RQTCNT que define el número de interrupciones del reloj de tiempo real que se considerarán como una unidad de tiempo del sistema[3].

# C A P I T U L O IV

## CONCLUSIONES

Luego del montaje y prueba del controlador se ha podido llegar a las siguientes conclusiones:

- El sistema se comporta dentro de los rangos especificados, y se obtiene un seguimiento de la referencia muy aproximado a los márgenes esperados. Con el comportamiento obtenido, podemos decir que el controlador es bastante preciso dado el tipo de máquina y la forma de control utilizada. Utilizando la caldera dentro de un rango de 60 PSI a 120 PSI, un error de 5 PSI siempre es menor de un 10%, lo cual es bastante aceptable.

Si se deseara mejorar la precisión del controlador, sería necesario utilizar un conversor con un mayor número de "bits", ya que una variación de 1°C representa un cambio de hasta 3 P.S.I. para presiones altas. Además habría que utilizar una válvula proporcional de gas, para evitar el excesivo desgaste de las válvulas solenoide, debido a la apertura y cerraje de éstas en forma muy rápida. Por supuesto, la precisión deseada dependerá de la aplicación particular (si se compensara el alto costo de la válvula proporcional), la cual no está definida en nuestro caso por tratarse de un prototipo de laboratorio.

- Las facilidades de programación y graficación, añaden una gran versatilidad en el manejo y supervisión de la caldera, la cual podría ser deseable en alguna aplicación industrial.

- La utilización del sistema operativo RMX/80 permite, una vez comprendida la filosofía de diseño y obtenida cierta práctica en su aplicación, una mayor sencillez en el diseño total, ya que el usuario solo debe preocuparse de elaborar cada módulo por separado. Por supuesto que la factibilidad de aplicación de este sistema depende fuertemente de la complejidad del diseño global, ya que exige un mínimo de 2K de memoria ROM para su implementación, sin incluir los programas del usuario.

Para ilustrar las facilidades del RMX/80 podríamos especular acerca de alguna modificación que añadir al sistema, por ejemplo un manejador y supervisor de alarmas de la industria donde se ubique la caldera. Sería necesario añadir los sensores y puertos que hagan falta, como modificación del "hardware". Como modificación del "software", habría que codificar los nuevos programas de aplicación, sin añadir ningún cambio a los programas ya diseñados. Tendríamos como resultado, dos sistemas de control en un procesador, diseñados en forma casi independiente.

- Una desventaja de trabajar con un lenguaje de alto nivel como PLM/80 es la gran cantidad de memoria ROM que - debe utilizarse para alojar los programas, sin embargo, el tiempo de programación (cuyo costo es elevado) se reduce notablemente, pudiendo compensarse los costos en muchas aplicaciones.

# A P E N D I C E  A

## PROCEDIMIENTO DE OPERACION DEL CONTROLADOR

El procedimiento de operación del controlador puede resumirse en los siguientes pasos:

- EN GENERAL:

Asegurarse antes de suministrar potencia al equipo, que los sensores u los aditamientos de la caldera se encuentran enchufados CADA UNO EN SU SITIO. Esto es muy importante para evitar daños al controlador.

- AL TRABAJAR SIN TERMINAL:

Al encenderse el equipo, luego de un instante se presentará el valor de presión de la caldera en el visualizador superior y la hora (de valor aleatorio) en el inferior. El proceso de control se encuentra inicialmente interrumpido y el valor de referencia es "cero".

- HORA:

Si se desea alterar el valor de la hora se presiona la tecla "*". Inmediatamente se limpiará el campo de las horas indicando que pueden introducirse los números correspondientes.

La entrada de números es por la izquierda, y el vi

sualizador funcionará como un registro circular donde se permite introducir datos hasta presionar la tecla "repeat" , la cual finaliza la operación. Si durante el proceso, o al presionarse "REPEAT", el visualizador se apaga, esto indica una condición de error (tecla no válida o valor numérico de la hora no válido) y se deben introducir los datos de nuevo.

- REFERENCIA:

Para cambiar la referencia de presión de la caldera se debe presionar la tecla "#", con lo cual se limpiará el campo de la presión, permitiendo de allí en adelante la entrada de datos. La entrada de números es por la derecha y se podrán introducir indefinidamente hasta presionar la tecla "repeat". Un apagado de campo de presión indica una condición de error tal como haber presionado una tecla no numérica u otra que "repeat" o haber introducido un valor de referencia mayor de 150 PSI, lo cual no está permitido.

- ARRANQUE:

Para arrancar el proceso se debe presionar la tecla "A". Si el proceso ya ha sido arrancado, no se producira ningún dato.

- INTERRUPCION:

Presionando la tecla "I" se detiene la ejecución del proceso. No se toma ninguna acción si el proceso ya fue in

terrumpido por el usuario o por alguna condición de error. - Es de hacer notar que para alterar lo hora o la referencia NO es necesario interrumpir previamente el proceso pues todas estas operaciones pueden ejecutarse simultáneamente.

- AL TRABAJAR CON TERMINAL:

Cuando se trabaja con terminal, el teclado opera simultáneamente manteniendo la misma forma de operación explicada anteriormente.

La rata de baudios del terminal debe ser de 2400 para la correcta operación del equipo.

Al encender el aparato, se mostrará en el terminal un menú de comandos a escogerse, inmediatamente después se solicitará la introducción de la hora y luego se pedirá la programación de la referencia para las próximas 24 horas (detalles mas adelante).

Luego de introducirse la programación de la referencia, el monitor del CRT se encuentra esperando alguno de los comandos especificados en el menú.

- HORA:

Para cambiar la hora del reloj del sistema, deberá

introducirse el comando H, y el monitor responderá el siguiente mensaje:

- POR FAVOR INTRODUZCA LA HORA EXACTA (HHMMSS):

Deberán introducirse seis números seguidos correspondientes a horas, minutos y segundos y luego <CR> . Si el valor introducido es correcto, no se muestra ningún mensaje, y el sistema está en monitor. Por el contrario, si el valor corresponde a una hora válida aparecerá el mensaje:

HORA INVALIDA

y el sistema entrará en monitor, debiendo introducirse el comando H para colocar la hora correcta.

- PROGRAMACION DE REFERENCIA:

El comando P permite programar la referencia de la caldera para su operación en las 24 horas del día. El sistema responderá con el siguiente mensaje:

POR FAVOR INTRODUZCA EL PROGRAMA PARA EL DIA DE HOY
(S PARA TERMINAR):

La referencia se debe introducir por intervalos de valor constante, en forma de hora inicial, hora final y valor deseado para el intervalo de tiempo escogido.

Ya que el monitor resuelve la entrada en base a los -

dos puntos (":") que contienen las horas, el formato de entrada es totalmente libre como por ejemplo:

TEXTO 12:30 TEXTO 18:00 TEXTO 80 TEXTO

u otra forma:

12:30 18:00 80

Las horas solo pueden programarse en pasos de media - hora, por lo tanto, los minutos sólo permiten valores de Ø ó 30.

Si se desea programar una única referencia para todo el día, bastará con introducir el valor deseado y <CR>, el sistema efecutará la acción y volverá a monitor. Si se ha introducido el programa completo con una "S" y <CR> se entratá a monitor de nuevo.

Si se detecta error en las horas, la referencia o de otro tipo, se desplegarán los siguientes mensajes y se entrará en monitor:

HORA INVALIDA, PRESION INVALIDA o ERROR DE SINTAXIS

- ESTADO DE LA CALDERA:

Para conocer el estado de la caldera, es preciso utilizar el comando E, obteniéndose las siguientes salidas:

- Proceso interrumpido o arrancado con llama apagada:

086 GRADOS C., 000 P.S.I., 14:08:26

- Proceso arrancado y llama azul:

086 GRADOS C, 000 PSI, 14:08:26, ESTADO DE LA LLAMA: CORRECTO.

- Proceso arrancado y llama rojiza:

086 GRADOS C., ØØØ P.S.I., 14:08:26, ESTADO DE LA LLAMA= DUDOSO.

- VALOR DE LA REFERENCIA:

Para obtener información acerca del valor de la referencia, se introduce el comando G. Esto produce un gráfico de presión v.s. hora del día en la pantalla del CRT.

- VISUALIZACION DEL ESTADO DE LA CALDERA:

Mientras el proceso se encuentre arrancado, cada media hora se toma una muestra de la presión en la caldera. El comando V permite graficar estos valores en función de la hora del día como en el caso de la referencia.

- ARRANQUE:

El comando A arranca el proceso. Si éste ya fue - arrancado, se muestra el mensaje:

PROCESO YA ARRANCADO

- INTERRUPCION:

Con el comando I se interrumpe el proceso. Si éste ya fue interrumpido así se informa con el mensaje:

PROCESO YA INTERRUMPIDO

- MENU:

Para obtener una tabla de los comandos del monitor del controlador, se introduce el comando M.

Si se oprime cualquier tecla no incluída dentro del menú de comandos, se enviará un mensaje de ERROR DE SINTAXIS.

- MENSAJES DE EMERGENCIA:

Al detectarse una situación de emergencia, el proceso se interrumpe automáticamente quitando la energía a todos los elementos de potencia de la caldera y enviando cualquiera de los siguientes mensajes, dependiendo del caso particular:

- ERROR DE INSTALACION DE GAS:

El sistema no logró encender el gas o éste se apagó repentinamente. También cabe la posibilidad de inutilización del sensor.

-****** PELIGRO: NIVEL DE AGUA DEMASIADO ALTO******

-****** PELIGRO: NIVEL DE AGUA DEMASIADO BAJO******

-****** PELIGRO: SOBRE PRESION******.

Luego de cada uno de estos mensajes aparece el siguiente:

- PROCESO INTERRUMPIDO.

indicando la interrupción del proceso.

Luego de subsanarse la falla, puede arrancarse el proceso nuevamente mediante el comando A.

# A P E N D I C E B

## NOCIONES GENERALES SOBRE EL SISTEMA OPERATIVO RMX/80

En un sistema de tiempo real (como un controlador de procesos) pueden distinguirse dos características fundamentales:

- El sistema debe respoden a eventos externos totalmente asíncronos que dependen únicamente de la naturaleza del proceso a ser controlado y sincronizar apropiadamente sus funciones de control con la ocurrencia de estos eventos. Por ejemplo, en el caso de la caldera, el sistema debe obedecer a situaciones tan imprevistas como el manejo por teclado y visualizador, el manejo por CRT o cualquier situación de emergencia que requiera atención inmediata.

- El sistema debe ser capaz de realizar multiples funciones en un mismo intervalo de tiempo, es decir, funciones concurrentes. En nuestro caso particular, el sistema puede ser manejado simultáneamente por teclado y por CRT a la vez estarse realizando las funciones de control, actualización de los visualizadores y manejo del reloj.

Todas estas operaciones se realizan en forma concurrente gracias a la gran velocidad del microcomputador.

Si un sistema de esta naturaleza es manejado por un so

lo programa muy largo y complejo, para lograr el efecto de atención a eventos asincronos deberán emplearse lazos de espera o añadir una lógica de temporización por medio de interrupciones para evitar la pérdida de tiempo del procesador. Para lograr el efecto de concurrencia, deberán muestrearse secuencialmente las diversas condiciones del proceso o emplear un manejo de interrupciones bastante complejo.

Otra forma de atacar el problema es empleando un sistema operativo para aplicaciones en tiempo real, donde el usuario programa separadamente aquellas funciones que pueden realizarse en forma simultánea, y el núcleo del sistema operativo se encargará de compartir el CPU en base a una lógica de prioridades. También es conveniente que exista una comunicación entre estos módulos, lo cual se logra mediante llamadas a rutinas del sistema operativo.

Estos módulos reciben el nombre de "tareas" o "tasks" y con los programas de aplicación del usuario que dan sentido al sistema. En el RMX/80, la comunicación entre "tasks" se realiza a través de mensajes que se envían a los lugares de "intercambio" o "exchanges". Un "exchange" es una localidad donde uno o más "tasks" esperan por mensajes o donde uno o mas mensajes esperan ser removidos por "tasks".

Este manejo de "tasks" y mensajes se realiza en base

a una lógica de colas (FIFO).

Dentro de esta forma de operación del sistema, pueden distinguirse ciertos "estados" de los "tasks" dependiendo de su capacidad de competencia por el procesador. Estos estados son los siguientes:

- El "task" que "corre", es aquel que se encuentra operando o corriendo en un cierto momento.
- Los "tasks listos", son aquellos que compiten directamente por el procesador, es decir, que no esperan por ningún mensaje o sus retardos de tiempo han sido satisfechos. - El "task" que se encuentra corriendo siempre es el "task listo" de mayor prioridad.
- Los "tasks suspendidos" son aquellos que no compiten por los recursos del sistema, es decir, es como si fueran inexistentes. Esto se logra mediante una llamada de suspensión al sistema operativo, generalmente para facilitar la corrección de errores en el período de prueba del sistema.

Como se dijo anteriormente, el "task" que se encuentra corriendo, es el de más alta prioridad de la lista de "tasks listos", y se mantendrá corriendo hasta que deba esperar por un mensaje o por un retardo de tiempo (o ambas: por un mensaje cierto tiempo) o hasta que un "task" de mayor prioridad se encuentre listo para correr. En el primer caso (el "task" en forma "voluntaria" deja de correr), se remueve el task de

la lista de "tasks listos" y se introduce en la lista de - "tasks esperando" o en la de "tasks suspendidos" (un "task" puede suspenderse a sí mismo) y se pasa el control al "task listo" de mayor prioridad. El sistema operativo requiere - de una temporización que consiste en una interrupción (prio ridad 1) que cada vez que ocurre pasa el control al RMX para actualizar los retardos de la lista de espera y hacer los movimientos de listas necesarios. En caso de que no haya "tasks listos", el RMX define un "task ocioso" con la mínima prioridad, el cual consiste en colocar la procesador en "HALT" con las interrupciones habilitadas.

En el segundo caso (el "task" deja de correr por otro "task" de mayor prioridad), el task corriendo puede remover se de su estado y permanecer en la "lista de tasks listos" - por el hecho de que un "task" de mayor prioridad haya satisfecho su espera. Esto puede ocurrir ya sea por la actualización de su retardo o por recibir un mensaje del "task"corriendo.

Cuando ocurre este segundo caso, el RMX/80 salva toda la información vital necesaria para que el "task" que se ha interrumpido pueda continuar sin alteración de sus resultados.

Estos estados pueden visualizarse en la figura B-1.

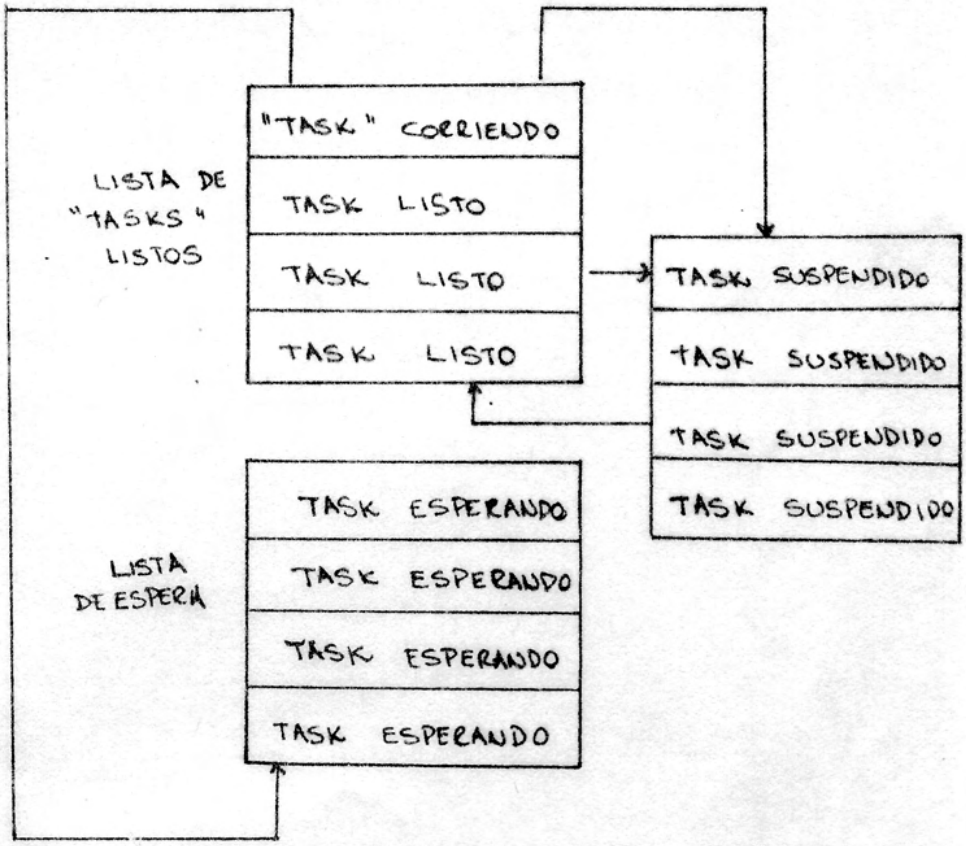



Figura B-1.

Las prioridades a asignarse a los "tasks" se representan por un número entre 0 y 255, para mayor y menor prioridad respectivamente. Los valores entre 0 y 127 se reservan para "tasks" de interrupción. El RMX/80 deshabilita las interrupciones mientras un "task" con prioridad entre Ø y 127 se encuentre corriendo y los habilita para valores de prioridad entre 128 y 255.

El valor 255 se asigna al "task ocioso" del RMX/80 y el valor Ø se asigna a un "task" de prueba ("debugger") si

se piensan utilizar las facilidades de "DEBUG" del RMX/80.

Es de hacer notar que dos o mas "tasks" pueden tener exactamente la misma prioridad, pero la competencia entre - ellos se basará en cuál sea el primero en llegar a la lista de "tasks" listos (FIFO).

Las precauciones que deben tenerse principalmente al configurar un sistema basado en esta filosofía son las si-- guientes:

- Asignar las prioridades correctamente.
- Evitar que simultáneamente, dos o más "tasks" esperen mensajes mutuamente , originando el llamado "abrazo mortal" o tranca del sistema ("deadlock").

Utilizando esta forma de diseñar sistemas de tiempo real, puede incrementarse la capacidad de procesamiento del microcomputador ("throughput"), ya que se eliminan casi totalmente los lazos de espera y los ciclos de muestreo para la ejecución de ciertas funciones, logrando en una forma - mas sencilla la concurrencia y el manejo de eventos asíncronos que se mencionó al comienzo.

# A P E N D I C E C

## INTERFASE CON EL TERMINAL

Se diseñó un circuito en base a optoacopladores para convertir el lazo de 20 mA. de comunicación asíncrona que suple el SBC80/10 INTEL al estandard RS-232-C para adaptar el terminal CRT a nuestro procesador. Los niveles lógicos en RS-232-C son:

1 = de -3 a -25 V

0 = de +3 a +25 V.

El circuito se muestra en la figura C-1.

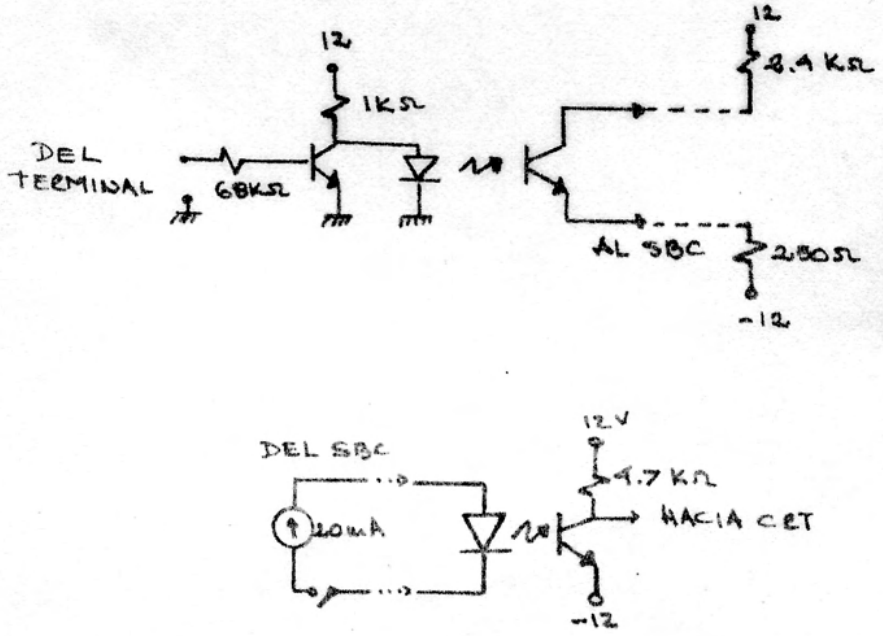



Figura C-1

# A P E N D I C E D

## CONEXIONES GENERALES

El "HARDWARE" del sistema está dispuesto con 6 tarjetas:

Tarjeta 1: SBC 80/10 INTEL

Tarjeta 2: Circuitos digitales y analógicos

Tarjeta 3: Potencia

Tarjeta 4: Teclado y visualizadores

Tarjeta 5: Fuentes de tierra digital

Tarjeta 6: Fuentes de tierra de potencia

### DISPOSICION DE PUERTOS Y SEÑALES EN EL PEINE. TARJETA Nº 2.

Dirección E6

Puerto 3 Bit Ø : Nivel bajo agua (1=agua; Ø =vapor), para 56.

Puerto 3 Bit 1: Muestra de gas ("Ø" llama roja), pata 63.

Puerto 3 Bit 2: Salida comparador conversor, pata 47

Puerto 3 Bit 3: Avisa Int. Clock (RTC) (Negado), pata 49.

Puerto 3 Bit 4: Nivel muy bajo agua (1= interrumpe), pata 55.

Puerto 3 Bit 5: Nivel alto agua (1=agua, Ø=vapor), pata 57.

Puerto 3 Bit 6: Nivel muy alto agua (1=interrumpe), pata 54.

Puerto 3 Bit 7: Muestra de gas ("1" apagado), pata 64.

Dirección E5

Puerto 2 Bit Ø: Vávula Ø gas

Puerto 2 Bit 1: Vávula 1 gas

Puerto 2 Bit 2: Vávula 2 gas

Puerto 2 Bit 3: Ventilador

Puerto 2 Bit 4: Vávula vapor

Puerto 2 Bit 5: Transformador A.T.

Puerto 2 Bit 6: Reset Flip-Flops

Puerto 2 Bit 7: Motor bomba de agua

Dirección E4

Puerto 1 Bit - 0 1 7: Al conversor D/A

Dirección E8

Puerto 4 Bit Ø: Enmascara Int. sobre presión ("1"), pata 66.

Puerto 4 Bit 1: Enmascara Int. teclado ("0"), pata 68

Puerto 4 Bit 2: Pata 69.

Puerto 4 Bit 3: Pata 70.

Puerto 4 Bit 4: Pata 67.

Puerto 4 Bit 5: Enmascara Int. Nivel de agua ("0"),

pata 53.

Puerto 4 Bit 6: Reset clock (RTC) (Negado), Pata 51.

Puerto 4 Bit 7: Enmascara clock (RTC)(Negado), Pata 50.

Dirección EA

Puerto 6 Bit Ø: Dato teclado

Puerto 6 Bit 1: Dato teclado

Puerto 6 Bit 2: Dato teclado

Puerto 6 Bit 3: Dato teclado

Puerto 6 Bit 4

Puerto 6 Bit 5

Puerto 6 Bit 6: Avisa Int. Sobre presión "Ø", pata 76.

Puerto 6 Bit 7: Avisa Int. teclado ("Ø"), pata 75.

| | |
|---|---|
| Pata 77<br>Pata 78 | Polarización fotoresistencia (gris y blanco) |
| Pata 71<br>Pata 72 | Presostato (morado y azul) |
| Pata 74 | Entrada Int. teclado |

Alimentación de bujías

Nivel muy alto : Pata 60 (Naranja)

Nivel alto: Pata 61 (Amarillo)

Nivel bajo: Pata 62 (Verde)

Nivel muy bajo: Pata 59 (Rojo)

Tierra Común: Pata 58 (Marron)

Dirección E9

Puerto 5 Bit 7: Reset teclado

Puerto 5 Bit 6-5-4: Selector visualizador

Puerto 5 Bit 0-3: Dato visualizador

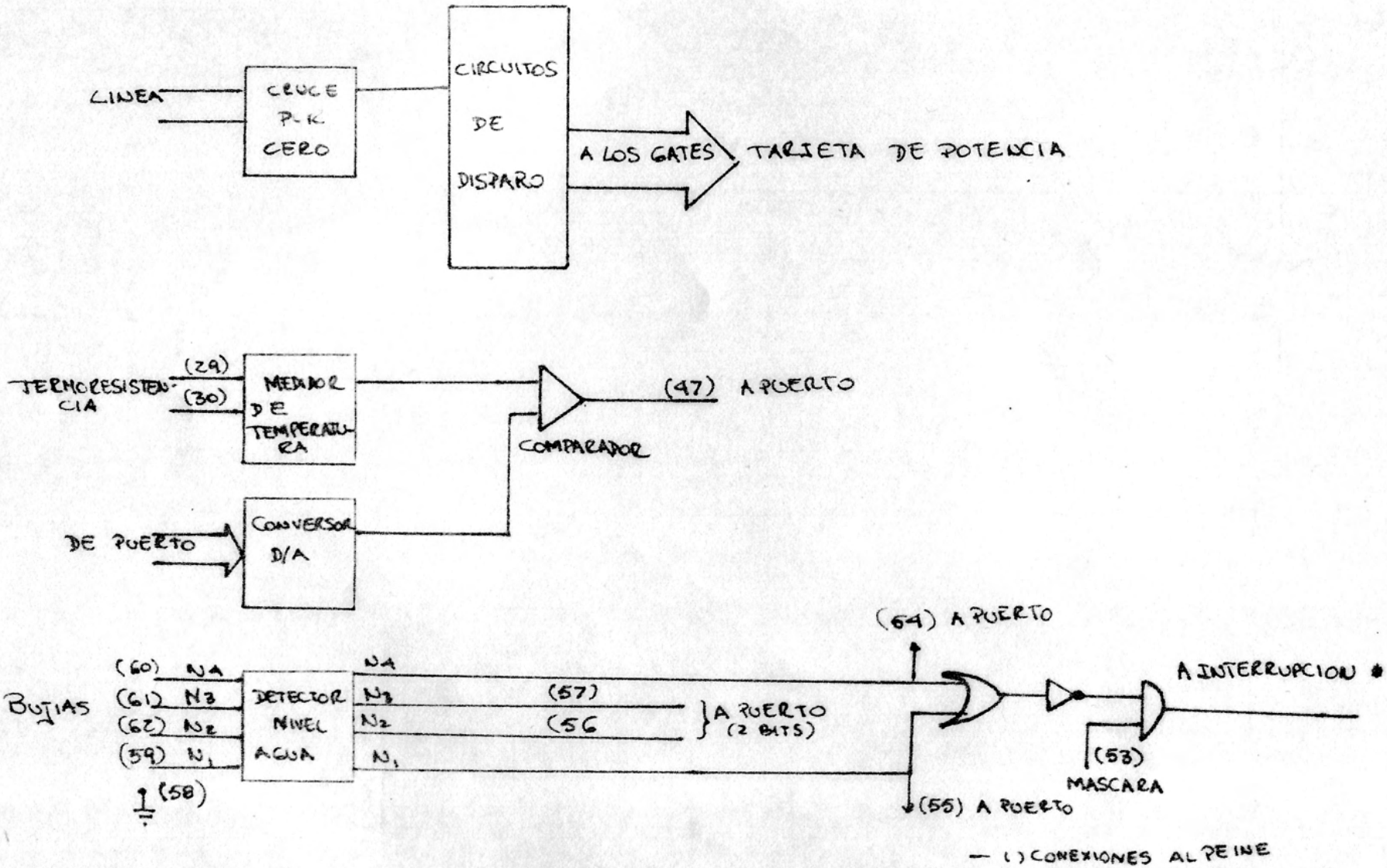



Figura D-1.- Conexiones tarjeta N° 2

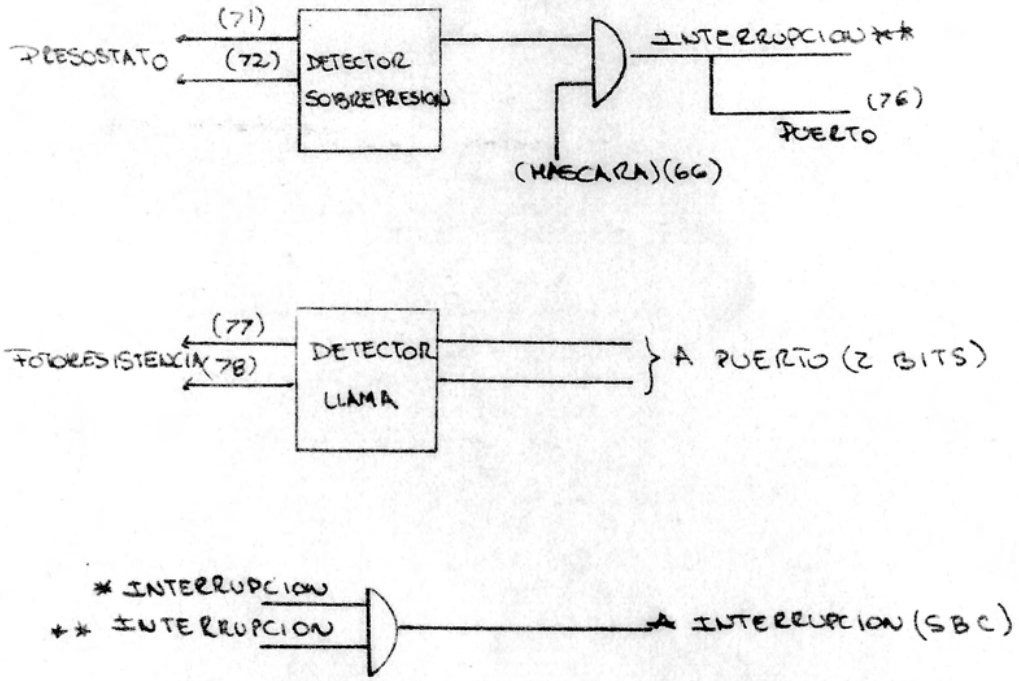



Figura D-2.- Conexiones tarjeta N° 2

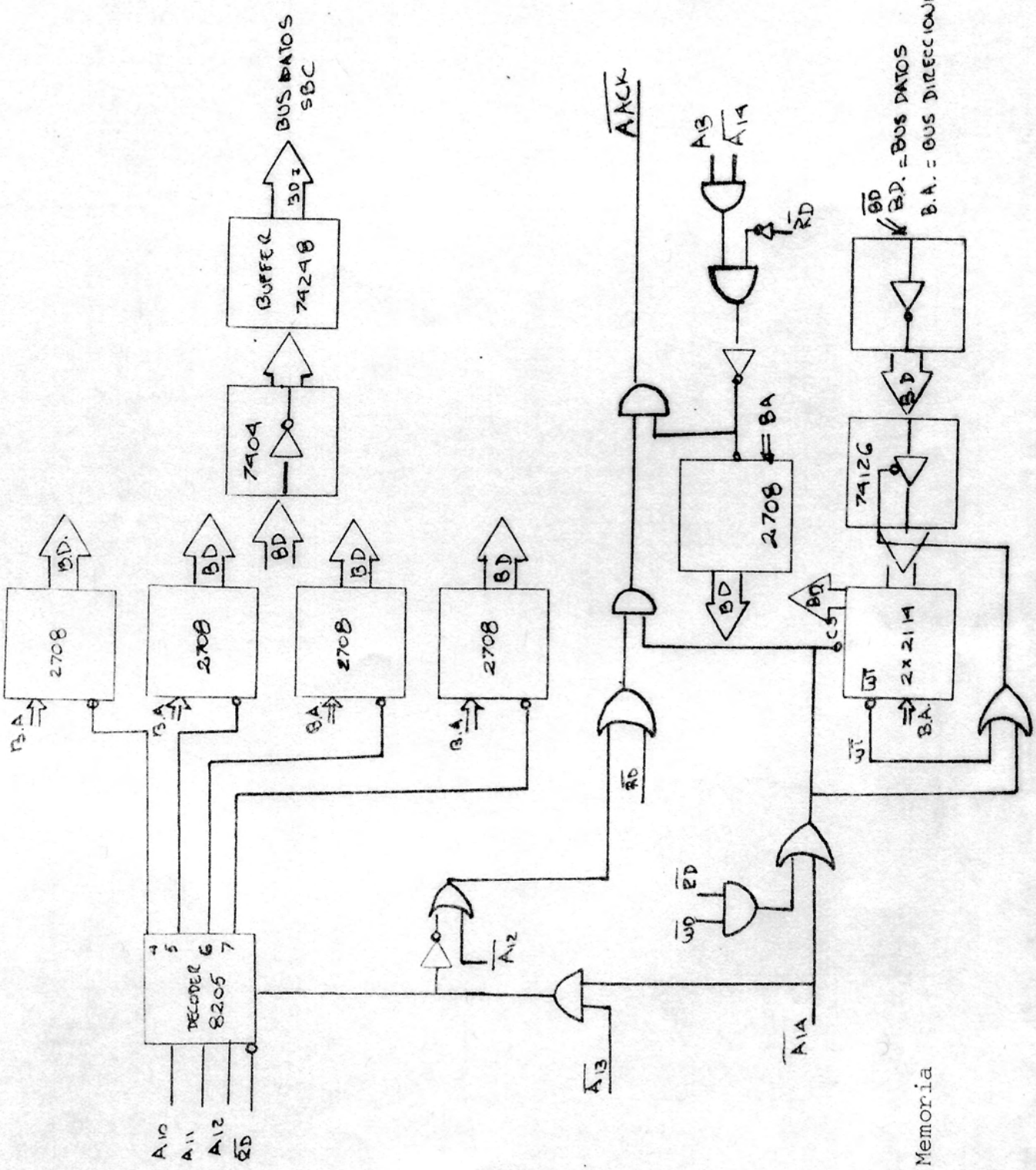



Figura D.3.

Expansión de Memoria

Tarjeta Nº 2

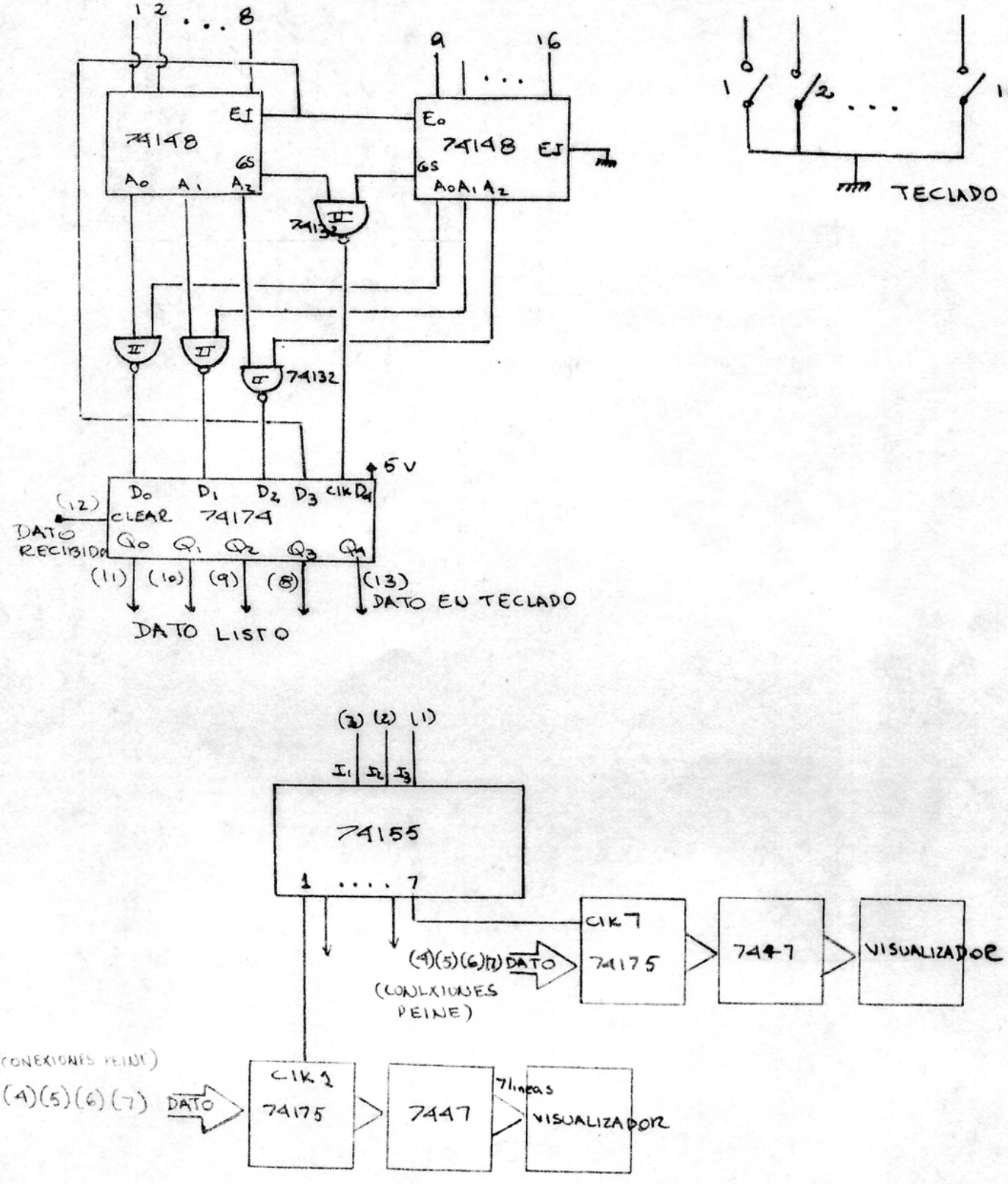



Figura D.4.- Conexiones tarjeta Nº 4

# A P E N D I C E E

## FUENTES DE POWER

Las Figuras E-1 y E-2, contienen los esquemas - - eléctricos de las fuentes de poder implementados para el con trolador. Las fuentes de la figura E-1 corresponden a la - tierra digital y analógica, mientras que los de la figura - E-2 corresponden a la tierra de potencia.

Figura E-1

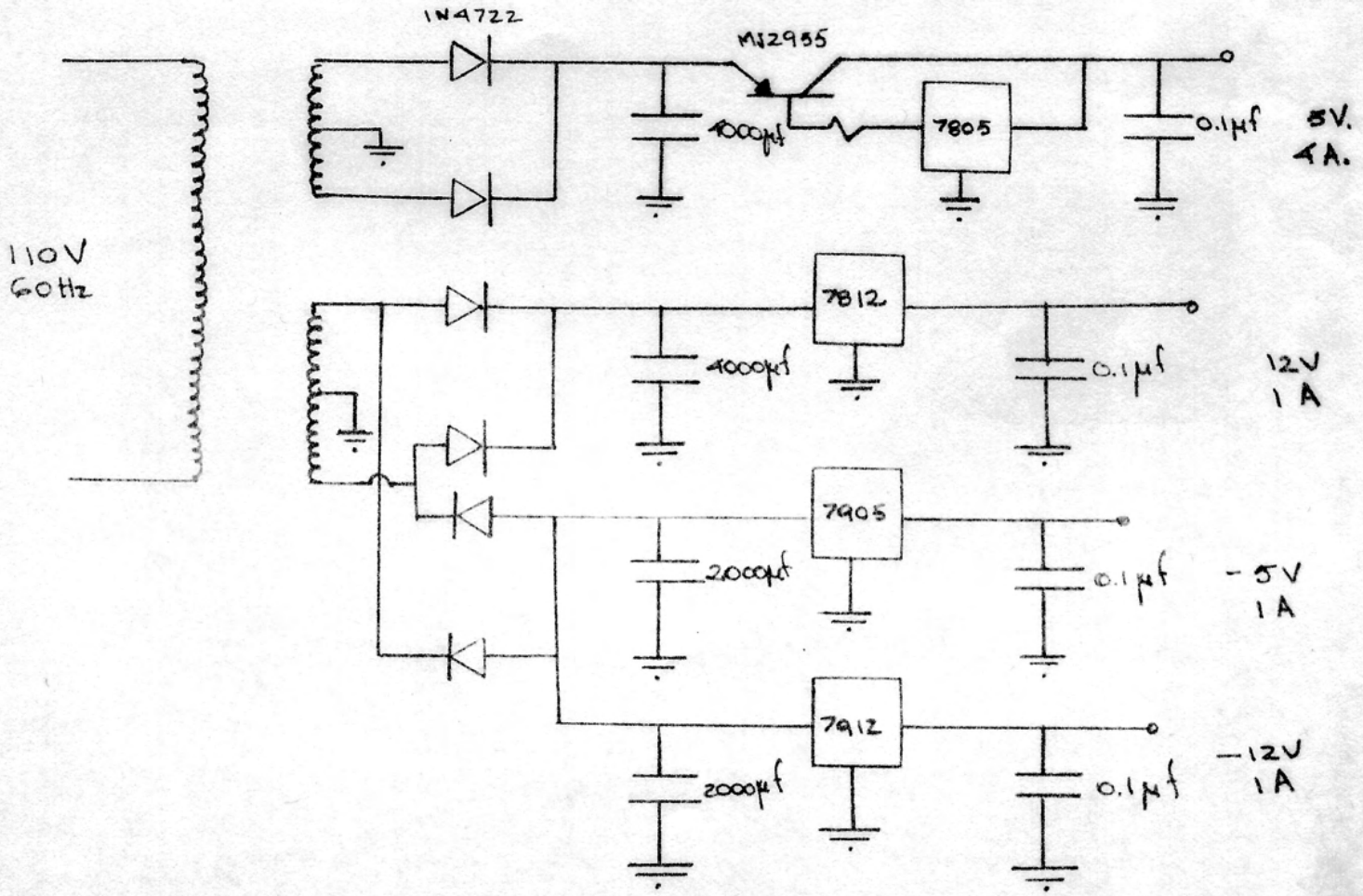
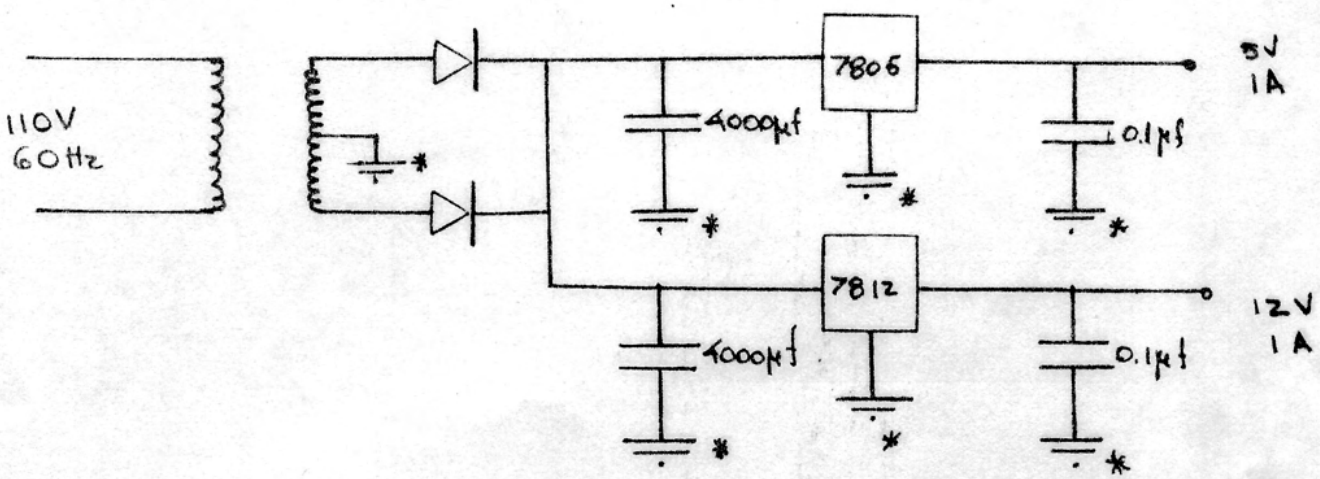


Figura E.2.



# A P E N D I C E F

## LISTADOS DE LOS PROGRAMAS

```
ISIS-II PL/M-80 V2.0 COMPILATION OF MODULE TECDIS
OBJECT MODULE PLACED IN :F1:TECDIS.OBJ
COMPILER INVOKED BY:  PLM80 :F1:TECDIS.PLM


  1          TECDIS:     DO;
  2     1            DECLARE TRUC BYTE PUBLIC;
  3     1            DECLARE CONTEM ADDRESS EXTERNAL;
  4     1            DECLARE T STRUCTURE (L ADDRESS,LE ADDRESS,T BYTE);
  5     1            DECLARE REFERENCIA(48) BYTE EXTERNAL;
  6     1            DECLARE H STRUCTURE (H BYTE,M BYTE,S BYTE) EXTERNAL;
  7     1            DECLARE (BLOQH,BLOQP) BYTE PUBLIC;
  8     1            DECLARE FF BYTE EXTERNAL;
  9     1            DECLARE I BYTE;
 10     1            DECLARE AP$VALOR ADDRESS;
 11     1            DECLARE VALOR BYTE EXTERNAL;
 12     1            DECLARE TEM STRUCTURE (L ADDRESS,LE ADDRESS,T BYTE);
 13     1            DECLARE (PONHOR) ADDRESS EXTERNAL;
 14     1            DECLARE RQL5EX(15) BYTE PUBLIC;
 15     1     RQSEND:     PROCEDURE (E,M) EXTERNAL;
 16     2            DECLARE (E,M) ADDRESS;
 17     2            END RQSEND;
 18     1     RQWAIT:     PROCEDURE (EX,T) ADDRESS EXTERNAL;
 19     2            DECLARE (EX,T) ADDRESS;
 20     2            END RQWAIT;
 21     1     PONDIS:     PROCEDURE(V,D) REENTRANT PUBLIC;
 22     2            DECLARE (V,D) BYTE;
 23     2            OUTPUT(0E9H)=((INPUT(0E9H) AND 80H) OR V) OR SHL((6-D),4);
 24     2            OUTPUT(0E9H)=(INPUT(0E9H) OR 70H);
 25     2            END PONDIS;
 26     1     BLANC:      PROCEDURE (N,K);
 27     2            DECLARE (N,K) BYTE;
 28     2            DO I=N TO N + K - 1;
 29     3            CALL PONDIS(15,I);
 30     3            END;
 31     2            END BLANC;
 32     1     PHORA:      PROCEDURE;
 33     2            DECLARE HOR(6) BYTE;
 34     2            DO WHILE 1;
 35     3            CALL BLANC(0,4);
 36     3            VALOR,I=0;
 37     3            DO WHILE VALOR<10;
 38     4            AP$VALOR=RQWAIT(.RQL5EX,0);
 39     4            IF VALOR<10 THEN DO;
 41     5            HOR(I)=VALOR;
 42     5            CALL PONDIS(VALOR,I);
 43     5            I=I+1;
 44     5            IF I>3 THEN I=0;
 46     5            END;
 47     4            END;
 48     3            IF VALOR=14 THEN DO;
 50     4            H.H=HOR(0)*10 + HOR(1);
 51     4            H.M=HOR(2)*10 + HOR(3);
 52     4            H.S=0;
 53     4            IF (H.HC=23) AND (H.MC=59) THEN DO;
 55     5            CALL RQSEND(.PONHOR,.TEM);
 56     5            RETURN;
 57     5            END;
 58     4            END;
```

```
 59  3            END;
 60  2            END PHORA;
 61  1    PREF:       PROCEDURE;
 62  2            DECLARE REF ADDRESS;
 63  2            DECLARE PRES(3) BYTE;
 64  2            DO WHILE 1;
 65  3            CALL BLANC(4,3);
 66  3            DO I=0 TO 2;
 67  4            PRES(I)=15;
 68  4            END;
 69  3            VALOR, I=0;
 70  3            DO WHILE VALOR<10;
 71  4            AP1VALOR=ROWAIT(.ROLSEX,0);
 72  4            IF VALOR<10 THEN DO;
 74  5            PRES(0)=PRES(1);
 75  5            PRES(1)=PRES(2);
 76  5            PRES(2)=VALOR;
 77  5            DO I=4 TO 6;
 78  6            CALL PONDIS(PRES(I-4),I);
 79  6            END;
 80  5            END;
 81  4            END;
 82  3            IF VALOR=14 THEN DO;
 84  4            DO I = 0 TO 2;
 85  5            IF PRES(I)=15 THEN PRES(I)=0;
 87  5            END;
 88  4            REF=PRES(2)+PRES(1)*10+PRES(0)*100;
 89  4            IF REF<=150 THEN DO;
 91  5            DO I=0 TO 47;
 92  6            REFERENCIA(I)=REF;
 93  6            END;
 94  5            RETURN;
 95  5            END;
 96  4            END;
 97  3            END;
 98  2            END PREF;
 99  1    TECDIS:     PROCEDURE PUBLIC;
100  2            TRUC=0;
101  2            TEM.LE,T.LE=5;
102  2            BLOOH,BLOOP=0;
103  2            CALL BLANC(0,7);
104  2            DO WHILE 1;
105  3            AP1VALOR=ROWAIT(.ROLSEX,0);
106  3            IF VALOR=10 THEN DO;
108  4            BLOOH=0FFH;
109  4            CALL PHORA;
110  4            BLOOH=0;
111  4            END;
112  3            ELSE DO;
113  4            IF VALOR=11 THEN DO;
115  5            BLOOP=0FFH;
116  5            CALL PREF;
117  5            BLOOP=0;
118  5            END;
119  4            ELSE DO;
120  5            IF (VALOR=13) AND (FF=0H) THEN DO;
122  6            FF=0FFH;
123  6            END;
124  5            ELSE DO;
125  6            IF (VALOR=15) AND (FF=0FFH) THEN DO;
```

```
127  7          CALL RQSEND(.CONTEM,.T);
128  7          END;
129  6          END;
130  5          END;
131  4          END;
132  3          END;
133  2          END TECDIS;
134  1          END;
```

MODULE INFORMATION:

```
    CODE AREA SIZE     = 02DFH    735D
    VARIABLE AREA SIZE = 002CH     44D
    MAXIMUM STACK SIZE = 000AH     10D
    122 LINES READ
    0 PROGRAM ERROR(S)
```

END OF PL/M-80 COMPILATION

ISIS-II PL/M-80 V2.0 COMPILATION OF MODULE RELOJ
OBJECT MODULE PLACED IN :F1:RELOJ.OBJ
COMPILER INVOKED BY: PLM80 :F1:RELOJ.PLM

```
 1          RELOJ:      DO;
 2    1           DECLARE BLOQH BYTE EXTERNAL;
 3    1           DECLARE H STRUCTURE(H BYTE,M BYTE,S BYTE) EXTERNAL;
 4    1           DECLARE (RETARD) STRUCTURE (MESSAGE$HEAD
                  ADDRESS,MESSAGE$TAIL ADDRESS,TASK$HEAD ADDRESS,TASK$TAIL
                  ADDRESS,
                  EXCHANGE$LINK ADDRESS) PUBLIC;
 5    1           DECLARE (VERHOR,HORLIS,PONHOR) STRUCTURE(MESSAGE$HEAD ADDRESS,
                  MESSAGE$TAIL ADDRESS,TASK$HEAD ADDRESS,TASK$TAIL ADDRESS)
                  EXTERNAL;
 6    1           DECLARE M STRUCTURE (LINK ADDRESS,LENGTH ADDRESS,TYPE BYTE,
                  H BYTE,M BYTE,S BYTE);
 7    1           DECLARE TIEMPO ADDRESS;
 8    1           DECLARE I BYTE;
 9    1      PONDIS:     PROCEDURE (V,D) EXTERNAL;
10    2           DECLARE (V,D) BYTE;
11    2           END PONDIS;
12    1      LISTO:      PROCEDURE;
13    2           CALL PONDIS(M.H/10,0);
14    2           CALL PONDIS(M.H MOD 10,1);
15    2           CALL PONDIS(M.M/10,2);
16    2           CALL PONDIS(M.M MOD 10,3);
17    2           END LISTO;
18    1      RQSEND:     PROCEDURE (EXCHANGE,MENSAJE) EXTERNAL;
19    2           DECLARE (EXCHANGE,MENSAJE) ADDRESS;
20    2           END RQSEND;
21    1      RQACPT:     PROCEDURE (EXCHANGE) ADDRESS EXTERNAL;
22    2           DECLARE EXCHANGE ADDRESS;
23    2           END RQACPT;
24    1      RQWAIT:     PROCEDURE (EXCHANGE,UN$TMP) ADDRESS EXTERNAL;
25    2           DECLARE (EXCHANGE,UN$TMP) ADDRESS;
26    2           END RQWAIT;
27    1      RELOJ:      PROCEDURE PUBLIC;
28    2           M.LENGTH=0;
29    2           DO WHILE 1;
30    3           DO WHILE M.H<24;
31    4           DO WHILE M.M<60;
32    5           IF BLOQH=0 THEN CALL LISTO;
34    5           DO WHILE M.S<60;
35    6           IF RQACPT(.PONHOR) <> 0 THEN DO;
37    7           M.H=H.H;
38    7           M.M=H.M;
39    7           M.S=H.S;
40    7           CALL LISTO;
41    7           END;
42    6           IF RQACPT(.VERHOR) <> 0 THEN
43    6           CALL RQSEND(.HORLIS,.M);
44    6           TIEMPO=RQWAIT(.RETARD,32);
45    6           M.S=M.S+1;
46    6           END;
47    5           M.S=0;
48    5           M.M=M.M+1;
49    5           END;
50    4           M.M=0;
```

```
 51  4           M.H=M.H+1;
 52  4           END;
 53  3           M.H=0;
 54  3           END;
 55  2           END RELOJ;
 56  1           END;
```

MODULE INFORMATION:

```
    CODE AREA SIZE     = 0107H    263D
    VARIABLE AREA SIZE = 0015H     21D
    MAXIMUM STACK SIZE = 0006H      6D
    60 LINES READ
    0 PROGRAM ERROR(S)
```

END OF PL/M-80 COMPILATION

ISIS-II PL/M-80 V2.0 COMPILATION OF MODULE MONCRT
OBJECT MODULE PLACED IN :F1:MONGRA.OBJ
COMPILER INVOKED BY: PLM80 :F1:MONGRA.PLM

```
 1          MONCRT:   DO;
 2  1               DECLARE GRAFICA(48) BYTE EXTERNAL;
 3  1               DECLARE DELAY ADDRESS;
 4  1               DECLARE PRESION(90) BYTE EXTERNAL;
 5  1               DECLARE R BYTE EXTERNAL;
 6  1               DECLARE REFERENCIA(48) BYTE PUBLIC;
 7  1               DECLARE H STRUCTURE(H BYTE,M BYTE,S BYTE) PUBLIC;
 8  1               DECLARE (REF,K) BYTE;
 9  1               DECLARE FF BYTE PUBLIC;
10  1               DECLARE ESTRUCTURA$EXCHANGE LITERALLY 'STRUCTURE (MESSAGE$HEAD
                    ADDRESS,MESSAGE$TAIL ADDRESS,TASK$HEAD ADDRESS,TASK$TAIL
                    ADDRESS,EXCHANGE$LINK ADDRESS)';
11  1               DECLARE CRLF(2) BYTE DATA(0DH,0AH);
12  1               DECLARE DOSPUN BYTE DATA(3AH);
13  1               DECLARE CR LITERALLY '0DH';
14  1               DECLARE LF LITERALLY '0AH';
15  1               DECLARE (ESP,SALEX,CONTEM,SINCRO,TENVIS,VERHOR,PONHOR,HORLIS)
                    ESTRUCTURA$EXCHANGE PUBLIC;
16  1               DECLARE (RQINPX,RQOUTX) ESTRUCTURA$EXCHANGE EXTERNAL;
17  1               DECLARE TEMDES STRUCTURE(LINK ADDRESS,LENGTH ADDRESS,TYPE BYTE,
                    REMAINDER ADDRESS);
18  1               DECLARE (SALIDA,ENTRADA) STRUCTURE(LINK ADDRESS,LENGTH ADDRESS,
                    TYPE BYTE,HOME$EXCHANGE ADDRESS,RESPONSE$EXCHANGE ADDRESS,
                    STATUS ADDRESS,BUFFER$ADR ADDRESS,COUNT ADDRESS,ACTUAL ADDRESS);
19  1               DECLARE (RESPUESTA) ADDRESS;
20  1               DECLARE BUFFER(75) BYTE;
21  1               DECLARE BUFFER5(*) BYTE DATA(CR,LF,'PROCESO YA '
                    ,'ARRANCADO',CR,LF);
22  1               DECLARE BUFFER6(*) BYTE DATA(CR,LF,'PROCESO YA '
                    ,'INTERRUMPIDO',CR,LF);
23  1          RQSTRC:   PROCEDURE PUBLIC;
24  2               OUTPUT(0E8H)=(INPUT(0E8H) AND 0BFH);
25  2               OUTPUT(0E8H)=(INPUT(0E8H) OR 040H) AND 7FH;
26  2               END RQSTRC;
27  1          RQSTPC:   PROCEDURE PUBLIC;
28  2               OUTPUT(0E8H)=INPUT(0E8H) OR 80H;
29  2               END RQSTPC;
30  1          RQSEND:   PROCEDURE (EXCHANGE,MESSAGE) EXTERNAL;
31  2               DECLARE (EXCHANGE,MESSAGE) ADDRESS;
32  2               END RQSEND;
33  1          RQWAIT:   PROCEDURE (EXCHANGE,TIEMPO) ADDRESS EXTERNAL;
34  2               DECLARE (EXCHANGE,TIEMPO) ADDRESS;
35  2               END RQWAIT;
36  1          SACA:     PROCEDURE (DIR,K) PUBLIC;
37  2               DECLARE (DIR,K) ADDRESS;
38  2               SALIDA.BUFFER$ADR=DIR;
39  2               SALIDA.COUNT=K;
40  2               CALL RQSEND(.RQOUTX,.SALIDA);
41  2               RESPUESTA=RQWAIT(.SALEX,0);
42  2               END SACA;
43  1          ASCBIN:   PROCEDURE (AP$DATO,N) ADDRESS PUBLIC;
44  2               DECLARE (K,AP$DATO,RESULTADO) ADDRESS;
45  2               DECLARE (DATO BASED AP$DATO) BYTE;
46  2               DECLARE (N,I,NH) BYTE;
```

```
 47  2            K=AP$DATO;
 48  2            RESULTADO=0;
 49  2            HH=1;
 50  2            DO I = 1 TO N;
 51  3            AP$DATO=K + N - I;
 52  3            RESULTADO=RESULTADO + ((DATO - 48)*HH);
 53  3            HH=HH*10;
 54  3            END;
 55  2            RETURN RESULTADO;
 56  2            END ASCBIN;
 57  1    BINASC:     PROCEDURE (DATO,AP$RESULTADO,N) PUBLIC;
 58  2            DECLARE (K,DATO,AP$RESULTADO) ADDRESS;
 59  2            DECLARE (RESULTADO BASED AP$RESULTADO) BYTE;
 60  2            DECLARE DIGITOS(*) BYTE DATA('0123456789');
 61  2            DECLARE (N,I) BYTE;
 62  2            K=AP$RESULTADO;
 63  2            DO I=1 TO N;
 64  3            AP$RESULTADO=K + N - I;
 65  3            RESULTADO=DIGITOS(DATO MOD 10);
 66  3            DATO=DATO/10;
 67  3            END;
 68  2            END BINASC;
 69  1    MUESTR:     PROCEDURE(V);
 70  2            DECLARE V BYTE;
 71  2            DECLARE HORA(2) BYTE;
 72  2            CALL BINASC(V,.HORA,2);
 73  2            CALL SACA(.HORA,2);
 74  2            END MUESTR;
 75  1    LIMPIA:     PROCEDURE;
 76  2            DECLARE A(2) BYTE DATA(27,12);
 77  2            CALL SACA(.A,2);
 78  2            DELAY=RQWAIT(.ESP,100);
 79  2            END LIMPIA;
 80  1    MODALF:     PROCEDURE;
 81  2            DECLARE A BYTE DATA(31);
 82  2            CALL SACA(.A,1);
 83  2            END MODALF;
 84  1    SALTA:      PROCEDURE;
 85  2            DECLARE A BYTE DATA(29);
 86  2            CALL SACA(.A,1);
 87  2            END SALTA;
 88  1    GRAFIC:     PROCEDURE (X,Y);
 89  2            DECLARE (X,Y) ADDRESS;
 90  2            DECLARE A BYTE;
 91  2            A=Y/32 OR 32;
 92  2            CALL SACA(.A,1);
 93  2            A=((Y AND 1FH)OR 60H);
 94  2            CALL SACA(.A,1);
 95  2            A=(X/32 OR 32);
 96  2            CALL SACA(.A,1);
 97  2            A=((X AND 1FH) OR 40H);
 98  2            CALL SACA(.A,1);
 99  2            CALL TIME(50);
100  2            END GRAFIC;
101  1    X:      PROCEDURE (N) ADDRESS;
102  2            DECLARE N BYTE;
103  2            RETURN ((816*(N))/48)+100;
104  2            END X;
105  1    Y:      PROCEDURE (A) ADDRESS;
106  2            DECLARE A BYTE;
```

```
107  2          RETURN ((A)*7)/2 + 100;
108  2          END V;
109  1    EJES:     PROCEDURE;
110  2          DECLARE (K,J) BYTE;
111  2          DECLARE I ADDRESS;
112  2          CALL LIMPIA;
113  2          CALL SALTA;
114  2          CALL GRAFIC(916,100);
115  2          CALL GRAFIC(100,100);
116  2          CALL GRAFIC(100,680);
117  2          CALL SALTA;
118  2          K,J=1;
119  2          DO I=117 TO 916 BY 17;
120  3          CALL GRAFIC(I,100);
121  3          CALL GRAFIC(I,90);
122  3          CALL SALTA;
123  3          IF (J MOD 6)=0 THEN DO;
125  4          CALL GRAFIC(I-10,60);
126  4          CALL MODALF;
127  4          CALL MUESTR(J/2);
128  4          CALL SALTA;
129  4          END;
130  3          J=J+1;
131  3          END;
132  2          DO I=135 TO 680 BY 35;
133  3          CALL GRAFIC(100,I);
134  3          CALL GRAFIC(90,I);
135  3          CALL SALTA;
136  3          CALL GRAFIC(60,I);
137  3          CALL MODALF;
138  3          CALL MUESTR(K);
139  3          CALL SALTA;
140  3          K=K+1;
141  3          END;
142  2          END EJES;
143  1    GRAFBU:   PROCEDURE (A$B);
144  2          DECLARE (A$B) ADDRESS;
145  2          DECLARE (BUFFER BASED A$B)(48) BYTE;
146  2          DECLARE I BYTE;
147  2          CALL EJES;
148  2          CALL SALTA;
149  2          DO I=0 TO 47;
150  3          CALL GRAFIC(X(I),Y(BUFFER(I)));
151  3          CALL GRAFIC(X(I+1),Y(BUFFER(I)));
152  3          IF (I<>47) THEN CALL GRAFIC(X(I+1),Y(BUFFER(I+1)));
154  3          END;
155  2          CALL SALTA;
156  2          CALL GRAFIC(0,770);
157  2          CALL MODALF;
158  2          END GRAFBU;
159  1    ENSENA:   PROCEDURE;
160  2          DECLARE I BYTE;
161  2          DECLARE (DIR$MENSAJE,AP$HORA) ADDRESS;
162  2          DECLARE (MENSAJE BASED DIR$MENSAJE) BYTE;
163  2          DECLARE (HORA BASED AP$HORA) STRUCTURE(H BYTE,M BYTE,S BYTE);
164  2          DECLARE TEMPE$VISIB(3) BYTE;
165  2          DECLARE BUFFER7(*) BYTE DATA(',ESTADO DE LA LLAMA:');
166  2          DECLARE BUFFER3(*) BYTE DATA('CORRECTO',CR,LF);
167  2          DECLARE BUFFER4(*) BYTE DATA('DUDOSO',CR,LF);
168  2          DECLARE BUFFER1(*) BYTE DATA(' GRADOS C., ');
```

```
169  2            DECLARE BUFFER2(*) BYTE DATA(' PSI, ');
170  2            CALL RQSEND(.VERHOR,.TEMDES);
171  2            AP$HORA=RQWAIT(.HORLIS,0)+5;
172  2            CALL RQSEND(.SINCRO,.TEMDES);
173  2            DIR$MENSAJE=RQWAIT(.TEMVIS,0)+5;
174  2            CALL BINASC(MENSAJE,.TEMPE$VISIB,3);
175  2            CALL SACA(.TEMPE$VISIB,3);
176  2            CALL SACA(.BUFFER1,LENGTH(BUFFER1));
177  2            IF MENSAJE>100 THEN DO;
179  3            CALL BINASC(PRESION(MENSAJE - 101),.TEMPEVISIB,3);
180  3            END;
181  2            ELSE DO;
182  3            DO I=0 TO 2;
183  4            TEMPEVISIB(I)='0';
184  4            END;
185  3            END;
186  2            CALL SACA(.TEMPE$VISIB,3);
187  2            CALL SACA(.BUFFER2,LENGTH(BUFFER2));
188  2            CALL RQSEND(.VERHOR,.TEMDES);
189  2            AP$HORA=RQWAIT(.HORLIS,0)+5;
190  2            CALL MUESTR(HORA.H);
191  2            CALL SACA(.DOSPUN,1);
192  2            CALL MUESTR(HORA.M);
193  2            CALL SACA(.DOSPUN,1);
194  2            CALL MUESTR(HORA.S);
195  2            IF R<>0 THEN DO;
197  3            IF R=100 THEN DO;
199  4            CALL SACA(.BUFFER7,LENGTH(BUFFER7));
200  4            CALL SACA(.BUFFER3,LENGTH(BUFFER3));
201  4            END;
202  3            ELSE DO;
203  4            IF R=0FFH THEN DO;
205  5            CALL SACA(.BUFFER7,LENGTH(BUFFER7));
206  5            CALL SACA(.BUFFER4,LENGTH(BUFFER4));
207  5            END;
208  4            END;
209  3            END;
210  2            END ENSENA;
211  1    INT$PROCESO:    PROCEDURE;
212  2            FF=0FFH;
213  2            END INT$PROCESO;
214  1    ARR$PROCESO:    PROCEDURE;
215  2            CALL RQSEND(.CONTEM,.TEMDES);
216  2            END ARR$PROCESO;
217  1    ENS$MENU:   PROCEDURE;
218  2            DECLARE BUFFER1(*) BYTE DATA(CR,LF,
             '   COMANDOS DE OPERACION:',CR,LF,LF,
             '        H: CAMBIO DE HORA,',CR,LF,
             '        P: PROGRAMAR LA CALDERA,',CR,LF,
             '        G: GRAFICO DEL PROGRAMA,',CR,LF,
             '        V: GRAFICO DE PRESION,',CR,LF,
             '        E: ESTADO DE LA CALDERA,',CR,LF,
             '        A: ARRANCAR,',CR,LF,
             '        I: INTERRUMPIR,',CR,LF,
             '        M: MENU',CR,LF,LF);
219  2            CALL LIMPIA;
220  2            CALL SACA(.BUFFER1,LENGTH(BUFFER1));
221  2            END ENS$MENU;
222  1    ERROR1:     PROCEDURE;
223  2            DECLARE BUFFER1(*) BYTE DATA(CR,LF,'PRESION INVALIDA',
```

```
                CR,LF);
224   2         CALL SACA(.BUFFER1,LENGTH(BUFFER1));
225   2         END ERROR1;
226   1   ERROR2:   PROCEDURE;
227   2         DECLARE BUFFER1(*) BYTE DATA(CR,LF,'ERROR DE SINTAXIS',
                CR,LF);
228   2         CALL SACA(.BUFFER1,LENGTH(BUFFER1));
229   2         END ERROR2;
230   1   ERROR:    PROCEDURE;
231   2         DECLARE BUFFER1(*) BYTE DATA(CR,LF,'HORA INVALIDA',CR,LF);
232   2         CALL SACA(.BUFFER1,LENGTH(BUFFER1));
233   2         END ERROR;
234   1   MENU:     PROCEDURE BYTE;
235   2         DECLARE BUFFER1(3) BYTE;
236   2         ENTRADA.BUFFER$ADR=.BUFFER1;
237   2         ENTRADA.COUNT=1;
238   2         CALL RQSEND(.RQINPX,.ENTRADA);
239   2         RESPUESTA=RQWAIT(.SALEX,0);
240   2         IF (BUFFER1(0)='I') THEN DO;
242   3         IF FF=0 THEN DO;
244   4         RETURN 3;
245   4         END;
246   3         ELSE RETURN 8;
247   3         END;
248   2         IF (BUFFER1(0)='A') THEN DO;
250   3         IF FF=0FFH THEN DO;
252   4         RETURN 4;
253   4         END;
254   3         ELSE RETURN 7;
255   3         END;
256   2         IF BUFFER1(0)='V' THEN RETURN 9;
258   2         IF BUFFER1(0)='M' THEN RETURN 5;
260   2         IF BUFFER1(0)='G' THEN RETURN 6;
262   2         IF BUFFER1(0)='P' THEN RETURN 1;
264   2         IF BUFFER1(0)='H' THEN RETURN 0;
266   2         IF BUFFER1(0)='E' THEN RETURN 2;
268   2         RETURN 10;
269   2         END MENU;
270   1   PONER$HORA:  PROCEDURE;
271   2         DECLARE I BYTE;
272   2         DECLARE INICIO(8) BYTE;
273   2         DECLARE BUFFER1(*) BYTE DATA (CR,LF,'POR FAVOR INTRODUZCA',
                ' LA HORA EXACTA (HHMMSS):');
274   2         CALL SACA(.BUFFER1,LENGTH(BUFFER1));
275   2         ENTRADA.BUFFER$ADR=.INICIO;
276   2         ENTRADA.COUNT=6;
277   2         CALL RQSEND(.RQINPX,.ENTRADA);
278   2         RESPUESTA=RQWAIT(.SALEX,0);
279   2         DO I=0 TO 5;
280   3         IF INICIO(I)<30H OR INICIO(I)>39H THEN DO;
282   4         CALL ERROR;
283   4         RETURN;
284   4         END;
285   3         INICIO(I)=INICIO(I) - 30H;
286   3         END;
287   2         H.H=INICIO(1) + INICIO(0)*10;
288   2         H.M=INICIO(3) + INICIO(2)*10;
289   2         H.S=INICIO(5) + INICIO(4)*10;
290   2         IF H.H>23 OR H.M>59 OR H.S>59 THEN
291   2         DO;
```

```
292  3            CALL ERROR;
293  3            RETURN;
294  3            END;
295  2            CALL RQSEND(.PONHOR,.TEMDES);
296  2            END PONER$HORA;
297  1    VALTEM.     PROCEDURE (AP) BYTE;
298  2            DECLARE AP BYTE;
299  2            DO WHILE (BUFFER(AP)<30H OR BUFFER(AP)>39H) AND AP<
                  (ENTRADA.ACTUAL-1);
300  3            AP=AP+1;
301  3            END;
302  2            IF BUFFER(AP)>2FH AND BUFFER(AP)<3AH THEN DO;
304  3            IF BUFFER(AP+1)>2FH AND BUFFER(AP+1)<3AH THEN DO;
306  4            IF BUFFER(AP+2)>2FH AND BUFFER(AP+2)<3AH THEN DO;
308  5            REF=ASCBIN(.BUFFER(AP),3);
309  5            IF REF>150 THEN DO;
311  6            REF=0;
312  6            RETURN 0;
313  6            END;
314  5            ELSE DO;
315  6            RETURN 0FFH;
316  6            END;
317  5            END;
318  4            ELSE DO;
319  5            REF=ASCBIN(.BUFFER(AP),2);
320  5            RETURN 0FFH;
321  5            END;
322  4            END;
323  3            ELSE DO;
324  4            REF=BUFFER(AP)-30H;
325  4            RETURN 0FFH;
326  4            END;
327  3            END;
328  2            ELSE RETURN 0;
329  2            END VALTEM;
330  1    PROGRAMA:   PROCEDURE;
331  2            DECLARE (SW1,I,J) BYTE;
332  2            DECLARE DATO(2) STRUCTURE(HORA BYTE,MIN BYTE);
333  2            DECLARE POSICION(2) BYTE;
334  2            DECLARE BUFFER1(*) BYTE DATA(CR,LF,'INTRODUZCA EL PROGRAMA',
                  ' DESEADO PARA EL DIA DE HOY (S PARA TERMINAR):',CR,LF);
335  2            CALL SACA(.BUFFER1,LENGTH(BUFFER1));
336  2            ENTRADA.BUFFER$ADR=.BUFFER;
337  2            ENTRADA.COUNT=72;
338  2            DO WHILE 1;
339  3            CALL RQSEND (.RQINFX,.ENTRADA);
340  3            RESPUESTA=RQWAIT(.SALEX,0);
341  3            IF BUFFER(0)=53H THEN RETURN;
343  3            I,SW1=0;
344  3            DO J=0 TO 1;
345  4            DO WHILE (BUFFER(I)<>3AH)AND(I<(ENTRADA.ACTUAL-1));
346  5            I=I+1;
347  5            END;
348  4            IF (BUFFER(I)<>3AH) AND J=0 THEN DO;
350  5            SW1=VALTEM(0);
351  5            IF SW1=0 THEN
352  5            DO;
353  6            CALL ERROR1;
354  6            RETURN;
355  6            END;
```

```
356   5           ELSE DO;
357   6           DO K=0 TO 47;
358   7           REFERENCIA(K)=REF;
359   7           END;
360   6           RETURN;
361   6           END;
362   5           END;
363   4           ELSE DO;
364   5           IF BUFFER(I)<>3AH AND J=1 THEN DO;
366   6           CALL ERROR;
367   6           RETURN;
368   6           END;
369   5           ELSE DO;
370   6           IF (BUFFER(I-1)>2FH AND BUFFER(I-1)<3AH) AND I>0 THEN DO;
372   7           DATO(J).HORA=BUFFER(I-1)-30H;
373   7           IF (BUFFER(I-2)>2FH AND BUFFER(I-2)<33H) AND I>1 THEN DO;
375   8           DATO(J).HORA=DATO(J).HORA+(BUFFER(I-2)-30H)*10;
376   8           IF DATO(J).HORA>23 THEN DO;
378   9           CALL ERROR;
379   9           RETURN;
380   9           END;
381   8           END;
382   7           ELSE DO;
383   8           CALL ERROR;
384   8           RETURN;
385   8           END;
386   7           IF (BUFFER(I+2)=30H) AND I<128 THEN DO;
388   8           IF BUFFER(I+1)=33H THEN DO;
390   9           DATO(J).MIN=30;
391   9           IF J=1 THEN SW1=VALTEM(I+3);
393   9           END;
394   8           ELSE DO;
395   9           IF BUFFER(I+1)=30H THEN DO;
397  10           DATO(J).MIN=0;
398  10           IF J=1 THEN SW1=VALTEM(I+3);
400  10           END;
401   9           ELSE DO;
402  10           CALL ERROR;
403  10           RETURN;
404  10           END;
405   9           END;
406   8           END;
407   7           ELSE DO;
408   8           CALL ERROR;
409   8           RETURN;
410   8           END;
411   7           END;
412   6           ELSE DO;
413   7           CALL ERROR;
414   7           RETURN;
415   7           END;
416   6           END;
417   5           END;
418   4           I=I+5;
419   4           END;
420   3           IF SW1=00 THEN DO;
422   4           CALL ERROR1;
423   4           RETURN;
424   4           END;
425   3           POSICION(0)=(SHL(DATO(0).HORA,1)+DATO(0).MIN/30);
```

```
 426  3            POSICION(1)=(SHL(DATO(1).HORA,1)+DATO(1).MIN/30);
 427  3            IF POSICION(1)>POSICION(0) THEN DO;
 429  4            DO K=POSICION(0) TO (POSICION(1)-1);
 430  5            REFERENCIA(K)=REF;
 431  5            END;
 432  4            END;
 433  3            ELSE DO;
 434  4            IF POSICION(1)<POSICION(0) THEN DO;
 436  5            DO K=1 TO (POSICION(1));
 437  6            REFERENCIA(K-1)=REF;
 438  6            END;
 439  5            DO K=POSICION(0) TO 47;
 440  6            REFERENCIA(K)=REF;
 441  6            END;
 442  5            END;
 443  4            END;
 444  3            END;
 445  2            END PROGRAMA;
 446  1  MONCRT:      PROCEDURE PUBLIC;
 447  2            R,H,M,H,M,H,S=0;
 448  2            ENTRADA.LENGTH,SALIDA.LENGTH=17;
 449  2            SALIDA.RESPONSE$EXCHANGE,ENTRADA.RESPONSE$EXCHANGE=SALEX;
 450  2            ENTRADA.TYPE=8;
 451  2            SALIDA.TYPE=12;
 452  2            DO K=0 TO 47;
 453  3            GRAFICA(K),REFERENCIA(K)=0;
 454  3            END;
 455  2            CALL ENS$MENU;
 456  2            CALL PONER$HORA;
 457  2            CALL PROGRAMA;
 458  2            DO WHILE 1;
 459  3            DO CASE MENU;
 460  4            CALL PONER$HORA;
 461  4            CALL PROGRAMA;
 462  4            CALL ENSENA;
 463  4            CALL INT$PROCESO;
 464  4            CALL ARR$PROCESO;
 465  4            CALL ENS$MENU;
 466  4            CALL GRAFBU(.REFERENCIA);
 467  4            CALL SACA(.BUFFER5,LENGTH(BUFFER5));
 468  4            CALL SACA(.BUFFER6,LENGTH(BUFFER6));
 469  4            CALL GRAFBU(.GRAFICA);
 470  4            CALL ERROR2;
 471  4            END;
 472  3            END;
 473  2            END MONCRT;
 474  1            END;


MODULE INFORMATION:

    CODE AREA SIZE     = 00C9H   3529D
    VARIABLE AREA SIZE = 0142H    322D
    MAXIMUM STACK SIZE = 000CH     12D
    461 LINES READ
    0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
```

```
ISIS-II PL/M-80 V2.0 COMPILATION OF MODULE TERMO3
OBJECT MODULE PLACED IN :F1:TERMO3.OBJ
COMPILER INVOKED BY:  PLM80 :F1:TERMO3.PLM


  1          TERMO3:    DO;
  2     1          DECLARE ADD ADDRESS;
  3     1          DECLARE PRESION(90) BYTE EXTERNAL;
  4     1          DECLARE BLOQP BYTE EXTERNAL;
  5     1          DECLARE (RESPUESTA) ADDRESS;
  6     1          DECLARE (PRUEBA BASED RESPUESTA) BYTE;
  7     1          DECLARE (TEMVIS,SINCRO) STRUCTURE (MESSAGE$HEAD ADDRESS,
                   MESSAGE$TAIL ADDRESS,TASK$HEAD ADDRESS,TASK$TAIL ADDRESS,
                   EXCHANGE$LINK ADDRESS) EXTERNAL;
  8     1          DECLARE MENSAJE STRUCTURE (LINK ADDRESS,LENGTH ADDRESS,
                   TYPE BYTE,REMAINDER ADDRESS);
  9     1          DECLARE (RESULT,SETEAR,I) BYTE;
 10     1     PONDIS:    PROCEDURE (V,D) EXTERNAL;
 11     2          DECLARE (V,D) BYTE;
 12     2          END PONDIS;
 13     1     ROSEND:    PROCEDURE (E,M) EXTERNAL;
 14     2          DECLARE (E,M) ADDRESS;
 15     2          END ROSEND;
 16     1     ROWAIT:    PROCEDURE (EX,UT) ADDRESS EXTERNAL;
 17     2          DECLARE (EX,UT) ADDRESS;
 18     2          END ROWAIT;
 19     1     TERMO3:    PROCEDURE PUBLIC;
 20     2          DO WHILE 1;
 21     3          RESPUESTA=ROWAIT(.SINCRO,10) + 4;
 22     3          SETEAR=07FH;
 23     3          OUTPUT(0E4H)=SETEAR;
 24     3          DO I=0 TO 8;
 25     4          RESULT=INPUT(0E4H);
 26     4          IF (INPUT(0E6H) AND 4)=4 THEN
 27     4          RESULT=RESULT OR NOT(SETEAR);
 28     4          SETEAR=NOT(SETEAR);
 29     4          SETEAR=SHR(SETEAR,1);
 30     4          SETEAR=NOT(SETEAR);
 31     4          OUTPUT(0E4H)=RESULT AND SETEAR;
 32     4          END;
 33     3          RESULT=NOT(RESULT);
 34     3          ADD=((RESULT)*3)/4;
 35     3          RESULT=ADD;
 36     3          IF (PRUEBA<>3) THEN DO;
 38     4          MENSAJE.LENGTH=8;
 39     4          MENSAJE.TYPE=80;
 40     4          MENSAJE.REMAINDER=(RESULT);
 41     4          CALL ROSEND(.TEMVIS,.MENSAJE);
 42     4          END;
 43     3          ELSE DO;
 44     4          IF BLOQP=0H THEN DO;
 46     5          IF RESULT<=100 THEN DO;
 48     6          RESULT=0;
 49     6          END;
 50     5          ELSE RESULT=PRESION(RESULT -101);
 51     5          DO I=0 TO 2;
 52     6          SETEAR=RESULT MOD 10;
 53     6          CALL PONDIS(SETEAR,6-I);
 54     6          RESULT=RESULT/10;
```

```
 55  6           END;
 56  5           END;
 57  4           END;
 58  3           END;
 59  2           END TERMDC;
 60  1           END;
```

MODULE INFORMATION:

```
    CODE AREA SIZE     = 0150H    336D
    VARIABLE AREA SIZE = 000EH     14D
    MAXIMUM STACK SIZE = 0004H      4D
    60 LINES READ
    0 PROGRAM ERROR(S)
```

END OF PL/M-80 COMPILATION

ISIS-II PL/M-80 V2.0 COMPILATION OF MODULE ALGCON
OBJECT MODULE PLACED IN :F1:ALGCON.OBJ
COMPILER INVOKED BY: PLM80 :F1:ALGCON.PLM

```
 1        ALGCON:    DO;
 2   1            DECLARE B BYTE EXTERNAL;
 3   1            DECLARE (PRES$DES,PRES$MED) BYTE;
 4   1            DECLARE (BAN,R) BYTE PUBLIC;
 5   1            DECLARE GRAFICA(48) BYTE PUBLIC;
 6   1            DECLARE REFERENCIA(48) BYTE EXTERNAL;
 7   1            DECLARE PRESION(256) BYTE EXTERNAL;
 8   1            DECLARE ESTRUCTURA$EXCHANGE LITERALLY 'STRUCTURE (MESSAGE$HEAD
                  ADDRESS,MESSAGE$TAIL ADDRESS,TASK$HEAD ADDRESS,TASK$TAIL
                  ADDRESS,EXCHANGE$LINK ADDRESS)';
 9   1            DECLARE FF BYTE EXTERNAL;
10   1            DECLARE CR LITERALLY '0DH';
11   1            DECLARE LF LITERALLY '0AH';
12   1            DECLARE BUFFER(*) BYTE DATA(CR,LF,'PROCESO INTERRUMPIDO',CR,LF);
13   1            DECLARE BUFFER1(*) BYTE DATA(CR,LF,'****** PELIGRO :
                  SOBREPRESION ******',CR,LF);
14   1            DECLARE BUFFER2(*) BYTE DATA(CR,LF,'****** PELIGRO :
                  NIVEL DE AGUA DEMASIADO ALTO ******',CR,LF);
15   1            DECLARE BUFFER3(*) BYTE DATA(CR,LF,'****** PELIGRO :
                  NIVEL DE AGUA DEMASIADO BAJO ******',CR,LF);
16   1            DECLARE AP$HORA ADDRESS;
17   1            DECLARE (HORA BASED AP$HORA) STRUCTURE (H BYTE,M BYTE);
18   1            DECLARE FALSO STRUCTURE(LINK ADDRESS,LENGTH ADDRESS,TYPE
                  BYTE);
19   1            DECLARE SAL1EX ESTRUCTURA$EXCHANGE PUBLIC;
20   1            DECLARE (SINCRO,CONTEM,TEMVIS,RQOUTX,SALEX,VERHOR,HORLIS)
                  ESTRUCTURA$EXCHANGE EXTERNAL;
21   1            DECLARE DELAY ESTRUCTURA$EXCHANGE PUBLIC;
22   1            DECLARE (DIR$DESEADA,DIR$MEDIDA,RESPUESTA,RETARDO) ADDRESS;
23   1            DECLARE MEDIDA BASED DIR$MEDIDA BYTE;
24   1            DECLARE SALIDA STRUCTURE (LINK ADDRESS,LENGTH ADDRESS,TYPE
                  BYTE,HOME$EXCHANGE ADDRESS,RESPONSE$EXCHANGE ADDRESS,STATUS
                  ADDRESS,BUFFER$ADR ADDRESS,COUNT ADDRESS,ACTUAL ADDRESS);
25   1    DIS:       PROCEDURE;
26   2            DISABLE;
27   2            B=0FFH;
28   2            OUTPUT(0E8H)=(INPUT(0E8H) AND 0DFH);
29   2            ENABLE;
30   2            END DIS;
31   1    EN:     PROCEDURE;
32   2            DISABLE;
33   2            B=0;
34   2            OUTPUT(0E8H)=(INPUT(0E8H) OR 20H);
35   2            ENABLE;
36   2            END EN;
37   1    ROWAIT:    PROCEDURE (EXCHANGE,TIEMPO) ADDRESS EXTERNAL;
38   2            DECLARE (EXCHANGE,TIEMPO) ADDRESS;
39   2            END ROWAIT;
40   1    ROSEND:    PROCEDURE (EXCHANGE,MENSAJE) EXTERNAL;
41   2            DECLARE (EXCHANGE,MENSAJE) ADDRESS ;
42   2            END ROSEND;
43   1    SACA:      PROCEDURE (DIR,K);
44   2            DECLARE (DIR,K) ADDRESS;
45   2            SALIDA.BUFFER$ADR=DIR;
```

```
 46  2            SALIDA.COUNT=K;
 47  2            CALL RQSEND(.RQOUTX,.SALIDA);
 48  2            RESPUESTA=RQWAIT(.SAL1EX,0);
 49  2            END SACA;
 50  1    ERROR:      PROCEDURE;
 51  2            DECLARE BUFFER1(*) BYTE DATA(CR,LF,'ERROR EN INSTALACION DE
                  GAS',CR,LF);
 52  2            CALL SACA(.BUFFER1,LENGTH(BUFFER1));
 53  2            END ERROR;
 54  1    ENCIEN:     PROCEDURE (V);
 55  2            DECLARE V BYTE;
 56  2            CALL DIS;
 57  2            IF V=1 THEN DO;
 59  3            OUTPUT(0E5H)=INPUT(0E5H) AND 0F7H;
 60  3            RETARDO=RQWAIT(.DELAY,80);
 61  3            END;
 62  2            OUTPUT(0E5H)=INPUT(0E5H) OR 20H;
 63  2            OUTPUT(0E5H)=INPUT(0E5H) OR NOT(SHL(0FFH,V));
 64  2            RETARDO=RQWAIT(.DELAY,160);
 65  2            IF (INPUT(0E6H) AND 80H)<>0 THEN DO;
 67  3            CALL ERROR;
 68  3            FF=0FFH;
 69  3            OUTPUT(0E5H)=INPUT(0E5H) AND 0F8H;
 70  3            END;
 71  2            OUTPUT(0E5H)=(INPUT(0E5H) AND 00FH);
 72  2            IF V=1 THEN DO;
 74  3            OUTPUT(0E5H)=INPUT(0E5H) OR 8;
 75  3            END;
 76  2            CALL EN;
 77  2            END ENCIEN;
 78  1    RLGCON:     PROCEDURE PUBLIC;
 79  2            DECLARE K BYTE;
 80  2            FF=0FFH;
 81  2            SALIDA.TYPE=12;
 82  2            SALIDA.RESPONSE$EXCHANGE=.SAL1EX;
 83  2            SALIDA.LENGTH=17;
 84  2            DO WHILE 1;
 85  3            BAN,R=0;
 86  3            DIR$DESEADA=RQWAIT(.CONTEM,0);
 87  3            FF=0;
 88  3            OUTPUT(0E8H)=(INPUT(0E8H) AND 0FEH);
 89  3            OUTPUT(0E5H)=INPUT(0E5H) OR 8;
 90  3            CALL RQSEND(.SINCRO,.FALSO);
 91  3            DIR$MEDIDA=RQWAIT(.TEMVIS,0) + 5;
 92  3            IF MEDIDA<100 THEN DO;
 94  4            OUTPUT(0E5H)=INPUT(0E5H) OR 10H;
 95  4            RETARDO=RQWAIT(.DELAY,160);
 96  4            OUTPUT(0E5H)=INPUT(0E5H) AND 0EFH;
 97  4            END;
 98  3            DO WHILE FF=0;
 99  4            IF (INPUT(0E6H) AND 1)=0 THEN DO;
101  5            K=0;
102  5            CALL DIS;
103  5            DO WHILE ((INPUT(0E6H) AND 20H)=0) AND (FF=0);
104  6            IF (INPUT(0E6H) AND 40H)<>0 THEN DO;
106  7            CALL TIME(23);
107  7            IF (INPUT(0E6H) AND 40H)<>0 THEN DO;
109  8            FF=0FFH;
110  8            BAN=1;
111  8            END;
```

```
112  7          END;
113  6          ELSE DO;
114  7          OUTPUT(0E5H)=INPUT(0E5H) OR 80H;
115  7          RETARDO=ROWAIT(.DELAY,32);
116  7          K=K+1;
117  7          IF K>120 THEN DO;
119  8          FF=0FFH;
120  8          BAN=2;
121  8          END;
122  7          END;
123  6          END;
124  5          OUTPUT(0E5H)=INPUT(0E5H) AND 7FH;
125  5          OUTPUT(0E5H)=INPUT(0E5H) OR 10H;
126  5          RETARDO=ROWAIT(.DELAY,96);
127  5          OUTPUT(0E5H)=INPUT(0E5H) AND 0EFH;
128  5          CALL EN;
129  5          END;
130  4          CALL ROSEND(.SINCRO,.FALSO);
131  4          DIR$MEDIDA=ROWAIT(.TEMVIS,0)+5;
132  4          IF MEDIDA>186 THEN DO;
134  5          FF=0FFH;
135  5          CALL SACA(.BUFFER1,LENGTH(BUFFER1));
136  5          END;
137  4          ELSE DO;
138  5          IF MEDIDA<=100 THEN
139  5          DO;
140  6          PRES$MED=0;
141  6          END;
142  5          ELSE DO;
143  6          PRES$MED=PRESION(MEDIDA-101);
144  6          END;
145  5          CALL ROSEND(.VERHOR,.FALSO);
146  5          AP$HORA=ROWAIT(.HORLIS,0) + 5;
147  5          PRES$DES=REFERENCIA(HORA.H*2 + HORA.M/30);
148  5          IF (HORA.M = 0) OR (HORA.M = 30) THEN DO;
150  6          GRAFICA(HORA.H*2 + HORA.M/30) = PRES$MED;
151  6          END;
152  5          IF PRES$DES>PRES$MED THEN DO;
154  6          OUTPUT(0E5H)=INPUT(0E5H) AND 0EFH;
155  6          IF (PRES$DES-PRES$MED)>4 THEN DO;
157  7          IF (INPUT(0E6H) AND 80H)<> 0 THEN
158  7          DO;
159  8          CALL ENCIEN(3);
160  8          END;
161  7          ELSE DO;
162  8          OUTPUT(0E5H)=INPUT(0E5H) OR 7H;
163  8          END;
164  7          END;
165  6          ELSE DO;
166  7          IF ((PRES$DES - PRES$MED)>3) AND ((INPUT(0E5H) AND  2)=0) THEN
167  7          DO;
168  8          IF (INPUT(0E6H) AND 80H) <> 0 THEN
169  8          DO;
170  9          CALL ENCIEN(2);
171  9          END;
172  8          ELSE DO;
173  9          OUTPUT(0E5H)=(INPUT(0E5H) OR 3) AND 0FBH;
174  9          END;
175  8          END;
176  7          ELSE DO;
```

```
177   8           IF ((PRES$DES - PRES$MED)>2) AND ((INPUT(0E5H) AND 7)=0) THEN
178   8           DO;
179   9           IF (INPUT(0E6H) AND 80H)<>0 THEN
180   9           DO;
181  10           CALL ENCIEN(1);
182  10           END;
183   9           ELSE DO;
184  10           OUTPUT(0E5H)=(INPUT(0E5H) OR 1) AND 0F9H;
185  10           END;
186   9           END;
187   8           END;
188   7           END;
189   6           END;
190   5           ELSE DO;
191   6           IF (PRES$MED - PRES$DES) > 2 THEN DO;
193   7           OUTPUT(0E5H)=INPUT(0E5H) AND 0F8H;
194   7           END;
195   6           IF (PRES$MED-PRES$DES)>5 THEN DO;
197   7           OUTPUT(0E5H)=INPUT(0E5H) OR 10H;
198   7           END;
199   6           ELSE DO;
200   7           OUTPUT(0E5H)=INPUT(0E5H) AND 0EFH;
201   7           END;
202   6           END;
203   5           IF (INPUT(0E5H) AND 7)<>0 THEN DO;
205   6           IF (INPUT(0E6H) AND 2)=0 THEN DO;
207   7           R=0FFH;
208   7           END;
209   6           ELSE DO;
210   7           IF (INPUT(0E6H) AND 80H)<>0 THEN DO;
212   8           FF=0FFH;
213   8           CALL ERROR;
214   8           END;
215   7           ELSE DO;
216   8           R=100;
217   8           END;
218   7           END;
219   6           END;
220   5           ELSE DO;
221   6           R=0;
222   6           END;
223   5           END;
224   4           END;
225   3           IF BAN=1 THEN DO;
227   4           CALL SACA(.BUFFER2,LENGTH(BUFFER2));
228   4           END;
229   3           ELSE DO;
230   4           IF BAN=2 THEN DO;
232   5           CALL SACA(.BUFFER3,LENGTH(BUFFER3));
233   5           END;
234   4           ELSE DO;
235   5           IF BAN=3 THEN DO;
237   6           CALL SACA(.BUFFER1,LENGTH(BUFFER1));
238   6           END;
239   5           END;
240   4           END;
241   3           OUTPUT(0E5H)=0;
242   3           CALL SACA(.BUFFER,LENGTH(BUFFER));
243   3           END;
244   2           END ALGCON;
```

```
245  1          END;
```

MODULE INFORMATION:

```
    CODE AREA SIZE     = 0507H  1287D
    VARIABLE AREA SIZE = 006EH   110D
    MAXIMUM STACK SIZE = 0008H     8D
    235 LINES READ
    0 PROGRAM ERROR(S)
```

END OF PL/M-80 COMPILATION

ISIS-II PL/M-80 V2.0 COMPILATION OF MODULE TABLA
OBJECT MODULE PLACED IN :F1:TABLA.OBJ
COMPILER INVOKED BY: PLM80 :F1:TABLA.PLM

```
   1        TABLA:      DO;
   2  1             DECLARE PRESION(90) BYTE PUBLIC DATA
                    (1,1,1,1,2,3,4,5,5,6,7,8,8,9,9,10,11,12,13,14,15,16,17,
                    18,19,20,21,22,23,24,26,27,29,30,30,31,33,34,36,38,39,
                    41,43,45,45,46,48,50,52,54,56,59,61,63,64,65,68,70,72,75,
                    77,80,83,86,87,88,91,94,97,100,103,106,110,113,115,116,
                    120,123,127,131,134,138,142,146,148,150,154,159,163,167);
   3  1             END;
```

MODULE INFORMATION:

```
     CODE AREA SIZE     = 005AH     90D
     VARIABLE AREA SIZE = 0000H      0D
     MAXIMUM STACK SIZE = 0000H      0D
     8 LINES READ
     0 PROGRAM ERROR(S)
```

END OF PL/M-80 COMPILATION

```
NAME CONFIG
;
;
CSEG
;
$INCLUDE (:F1:STD.MAC)
$INCLUDE (:F1:XCHADR.MAC)
$INCLUDE (:F1:GENTD.MAC)
$INCLUDE (:F1:CRTAB.MAC)
;
;
NTASK SET 0
NEXCH SET 0
;
;
DSEG
FALSA:DS 2
RQL0EX EQU FALSA
RQL2EX EQU FALSA
RQL3EX EQU FALSA
RQL4EX EQU FALSA
;
PUBLIC RQL0EX,RQL2EX,RQL3EX,RQL4EX
;
;
CSEG
;
STD TECDIS,38,100
STD RQTHOI,36,112,RQOUTX
STD RELOJ,32,190
STD TERMO3,32,230
STD MONCRT,40,200
STD ALGCON,38,250
;
;
XCHADR ESP
XCHADR RQL5EX
XCHADR HORLIS
XCHADR VERHOR
XCHADR PONHOR
XCHADR RETARD
XCHADR RQINPX
XCHADR RQL7EX
XCHADR RQL6EX
XCHADR SAL1EX
XCHADR SALEX
XCHADR RQWAKE
XCHADR RQOUTX
XCHADR TEMVIS
XCHADR SINCRO
XCHADR CONTEN
XCHADR DELAY
;
;
GENTD
CRTAB
;
;
PUBLIC RQRATE,RQTCNT
RQTCNT:DW 1
RQRATE:DW 611
END
```

# R E F E R E N C I A S

(1).- "STEAM PLANT OPERATION", E. B. Woodruff, H. B. Lammers, Mc Graw-Hill, 1967.

(2).- "CONTROL AUTOMATICO DE CALDERAS", F. D'Alvano, V. Guzmán y L. Ravinovici; Proyecto de grado, Universidad Simón Bolívar, Venezuela, 1975.

(3).- "RMX/80 USER'S GUIDE", Appendix G, Intel Corp. 1978.

(4).- "RMX/80 USER'S GUIDE", CAP. 2, Intel Corp., 1978.

(5).- "RMX/80 USER'S GUIDE", CAP. 3, Intel Corp., 1978.

# B I B L I O G R A F I A

1.- "STEAM PLANT OPERATION", E. B. Woodruff, H.B., Lammers; McGraw-Hill, 1967.

2.- "CONTROL AUTOMATICO DE CALDERAS", F. D'Alvano, V. Guzmán y L. Ravinovici; Proyecto de Grado, Universidad Simón Bolívar, Venezuela, 1975.

3.- "RMX/80 USERS GUIDE", Intel Corp., 1978.

4.- "STEAM TABLES", J. Keenan, F. Keyes, P. Hill, J. - Moore, Wiley Interscience, 1969.

5.- "PL/M-80 PROGRAMMING MANUAL", Intel Corp. 1978.

6.- "SBC 80/10 SINGLE BORAD COMPUTER, HARDWARE REFERENCE MANUAL", Intel Corp., 1976.

7.- "USING INTEL'S INDUSTRIAL CONTROL SERIES IN CONTROL APPLICATIONS", Application note AP-52, Intel, Corp.