ESCUELA SUPERIOR POLITECNICA DEL LITORAL

Facultad de Ingeniería en Electricidad

TOPICO ESPECIAL DE GRADUACION

"SISTEMAS DE CONTROL, SUPERVISION Y ADQUISICION DE DATOS"

Profesor - Director:

Ing. Javier Urquizo C.

Realizado por:

JAVIER ALOVILLO
MARIA ARNES
MANUEL CHAW
CESAR ESPIN
VICENTE GARCIA
XAVIER GUZMAN
MIGUEL ORTIZ
MANUEL RIVADENEIRA

Guayaquil - Ecuador
1.991

AGRADECIMIENTO

La realización de este proyecto, no solo constituye fruto de nuestro esfuerzo, sino de la colaboración desinteresada de muchas personas, a quien debemos extender nuestro más sincero agradecimiento.

Ing. Xavier Urquizo

Ing. Gustavo Bermúdez

Ing. Hernán Gutierrez

Además debemos agradecer :

A la ESPOL, cuna de nuestra formación.

A nuestros familiares, que han contribuído, directa o indirectamente en la realización de este proyecto.

Para ellos reiteramos nuestro agradecimiento y respeto por su valiosa colaboración.

INDICE GENERAL

CAPITULO 3 : COMUNICACION ENTRE MICROS .

CAPITULO 4 : COMUNICACION SERIAL ASINCRONICA .

CAPITULO 5 : CONTROL DE LA UNIDAD TERMINAL REMOTA

CAPITULO 6 : UNIDAD TERMINAL REMOTA .

CAPITULO 7 : VISUALIZACION .

CAPITULO 8 : ADQUISICION DE DATOS .

CAPITULO 9 : MUESTRA Y OBTENCION DE DATOS POR MEDIO DE PROCESOS ESTADISTICOS Y NUMERICOS

INTRODUCCION.

En la industria moderna el control de procesos productivos a distancia se está convirtiendo cada vez más en un requisito necesario tanto para reducir costos de producción como para optimización de los mismos.

El sistema de Supervision Control Adquision de Datos (SCADA) proporciona una respuesta a los sistemas de control a distancia. Dado que en nuestro medio estos sistemas no son producidos en nuestro pais, se hace necesario que exista personal que esté en capacidad de diseñar,implementar y dar mantenimiento a estos sistemas, para poder dejar de importar estos sistemas que por otro lado son bastante costosos.

Nuestra institución educativa a querido llenar este vacio para lo cual se estan capacitando técnicos en esta área. El presente libro muestra la forma en la cual se diseño un sistema con estas caracteristicas y la forma en que fué implementada cada una de sus partes.

El problema a resolver consiste en diseñar e implemnetar un sistema de supervisión control y adquisición de datos (SCADA) el cual deberá ser capás de cargar un programa de proceso a crearse en una estación de control, hacia una unidad terminal remota ubicada lejos de la estación de control, y desarrollar

este proceso sobre una planta industrial que en este caso será una caldera la cual está constituida por algunos dispositivos eléctricos los cuales serán controlados por el programa creado y cargado desde la estación maestra. Dada la gran distancia existente entre el centro de control y la planta, la comunicación entre estas deberá efectuarse a través de lineas telefónicas por medio de Modems. El sistema también debe luego de cargado el programa de proceso en la unidad terminal remota, debe visualizar por medio de un monitor en la estación de control como se está desarrollando el proceso en la planta, para que un operador pueda supervisar la planta desde la estación de control y si se produce una falla poder actuar inmediatamente.

El sistema también debe poseer un centro de estadistico en el cual van a realizar los calculos estadisticos del proceso, por parte de personal mas técnico los cuales deberán optimizar el proceso productivo de la planta. Los datos de temperatura son transmitidos cada cierto tiempo desde la estación de control hacia el centro de estadística y deberá ser en forma automática.

La presentación del proyecto se a dividido en bloques los cuales dan una descripción de cada una de las partes constitutivas del mismo. Los capítulos siguientes explican en detalle cada una de las partes que forman en conjunto el proyecto del Sistema de Supervisión Control y Adquisición de Datos aplicado a una industria.

CAPITULO 1

EDITOR DE COMANDOS

1.- INTRODUCCION.

En el proceso de Control una parte escencial del sistema es la estructuración de comandos de control que manejarán todo un proceso durante el día.

El Editor permite la creación de un grupo de instrucciones que luego pasarán por un proceso de compilación o acondicionamiento, a fin de que el microprocesador o controlador de planta entienda lo que el operador desea hacer con ella.

El Editor está estructurado en un formato de fácil utilización y previamente presentará menus de ayuda a fin de poder comprender mejor el manejo del mismo. El lenguaje de formación es una mezcla de PASCAL y ASSEMBLER lo que permite un mejor manejo del mismo.

El programa se subdivide en subprogramas llamados procedimientos, estos son externos e internos, con traspaso de variables que permiten una mejor interrelación entre ambos lenguajes. Además permitirá, a una nueva generación, la mejora del programa y de su utilización.

2.- ESTRUCTURACION DEL PROGRAMA.

El programa Editor se estructura de la siguiente manera, procedimientos externos que permiten el control de la pantalla, acceso a teclado y movimiento, y los procedimientos internos que permiten la creación de los comando y las características del sistema.

El Editor permite al operador escribir hasta 100 (cien) comandos numerados desde el 1 hasta el 100, la longitud de cada comando no debe exceder los 20 (veinte) caracteres.

Es posible borrar líneas, insertar líneas, reemplazar líneas y buscar comando por línea. Una vez terminada la estructuración de los comando se procederá a la grabación en un archivo cuyo nombre lo escoje el operador; con este archivo se podrá realizar la compilación, la extensión del archivo debe ser (.SCP) que significa: 'Sistema de Control de Proceso'. La salida del programa se da mediante la tecla < ESC >, lo que permite salir sin grabar o salir una vez grabado.

Como mencionamos en la introducción el programa tiene procedimientos externos e internos, estos procedimientos se detallan a continuación.

2.1 PROCEDIMIENTOS EXTERNOS.

Estos procedimientos fueron escritos en lenguaje

ASSEMBLER, utilizan como dispositivo principal subrutinas o llamados a interrupciones del BIOS de la máquina de trabajo, los procedimientos permiten crear una ventana de presentación y el menu de comandos, movimiento del cursor, busqueda de teclado presionado y contabilización de lineas por pantalla (20). Los procedimientos son los siguientes:

*** Ventana:**

Permite separar en dos ventanas la pantalla de trabajo, la primera , de fondo negro, que va desde la linea 1 hasta la 20, la segunda, de fondo violeta, desde la 21 hasta la 25, adicionalmente muestra en la pantalla inferior parte del menú de los comandos de operación, las letras de los comandos es blanca lo que permite una mejor visualización.

Para este procedimiento se utiliza la interrupción 09, 02 del bios, que permiten posicionar el cursor y mostrar en pantalla los caracteres deseados.

*** Menu - Menu1:**

Siguen escribiendo los comandos de operación del editor, estos comandos van identificados por las teclas de función (F1 hasta F9) y la de ESC. La forma de presentación se podrá observar en la figura 1.1

*** PosCur:**

Es un procedimiento que llama a la interrupción 02 del BIOS y permite posicionar el cursor en la posición deseada

por el usuario, los valores se pasan desde PASCAL y el orden es Columna,Fila; si deseamos el cursor en la columna 5 fila 2 se llama al procedimiento de la siguiente manera:

```
Poscur(5,2)
```

* **Page:**

Este procedimiento permite dar a Pascal la ubicación del cursor para monitorear la primera pantalla, es decir la escritura debe ser hasta la línea 20, una vez llegada a esta linea, se debe cambiar la página, el valor que devuelve a Pascal es de tipo word.

Su llamda es de la siguiente manera:

```
page(a);
```

* **Explore:**

Este procedimiento verifica y explora el teclado, lo que permite observar las teclas de función desde F1 hasta F9 y la tecla de ESC, mediante este procedimiento podemos controlar las funciones que deseamos en el programa de edición de comandos. La forma de llamado es la siguiente:

```
Explore(scan,ascii);
```

Scan es el código de exploración de funciones especiales y Asciidevuelve el valor de la tecla presionada

* **Subir,Bajar,Derecha, Izquierd:**

Estos procedimientos permiten el movimiento del cursor a través de la pantalla de edición. Los limites del movimiento son desde (0,0) hasta (0,20), desde (0,0) hasta (80,0), entre esos limites se da el movimiento del cursor.

2.2.- PROCEDIMIENTOS INTERNOS.

Los procedimientos internos están estructurados en lenguaje Pascal y permiten la interrelación entre el operador, sus funciones y los procedimientos externos, los procedimientos son los siguientes:

* **Pintando:**

Es la unión de los procedimientos externos ventana,menu,menu1, lo que hace es subdividir la pantalla en dos, la pantalla de edición y la pantalla de ayuda y comandos.

* **Ayuda:**

Este procedimiento se activa al presionar la tecla de función que llame a la pantalla que muestra el mensaje de ayuda, este mensaje explica al operador los comandos y las opciones de cada comando, así como las funciones principales del programa.

* **Editor:**

Este procedimiento realiza la edición de los comandos, está estructurado por medio de arreglos de registros, estos registros guardan el numeral de cada línea y los comandos (hasta 20 caracteres), cada pantalla permite hasta 20 líneas de comandos y las páginas o pantallas de edición son cinco(5), en total se pueden escribir hasta 100 comandos diferentes.

Antes de ingresar al editar, se visualiza un menú auxiliar de ayuda para asegurar que el operador escriba correctamente los comandos, en ésta opción, sólo puede retroceder borrando, el movimiento del cursor se pierde.

* **Grabar:**

Este procedimiento permite grabar el texto escrito, el usuario puede dar un nombre cualquiera, pero la extensión del archivo a grabar siempre debe ser .SCP, para no haber error en el momento de compilación.

* **Mostrar:**

Permite visualizar los comandos escritos, puede visualizar el operador, el texto original o el texto modificado.

* **Reemplazar:**

Ubica el comando, en la línea y luego pide al usuario el comando nuevo que desea colocar, antes verifica que el operador esté seguro de cambiar el comando.

* **Buscar:**

Si el operador desea encontrar determinados comandos y su ubicación mediante la búsqueda es posible determinar la ubicación del comando.

* **Borrar:**

Borra la línea indicada por el operador y verifica que su deseo sea correcto, si una línea que no ha sido editada es mandada a borrar, le indicará que dicha línea no existe.

* **Insertar:**

Introduce nuevas líneas de comandos en el texto previamente editado, lo puede hacer entre líneas o al final de la línea, si es al final, le pedirá el número de líneas a insertarse, desade 0 para una sola en adelante.

* **Reset:**

Limpia la pantalla, borrando lo que se ha escrito previamente.

* **Cargar:**

Es un procedimiento auxiliar, que carga un archivo grabado anteriormente, de acuerdo al nombre dado, se lo utiliza dentro de la compilación.

Cada uno de estos procedimientos llaman a su vez a los procedimientos de assembler para su configuración total.

2.3.- MENU PRINCIPAL.

El menú principal utiliza una sentencia de control While que espera salir de ella si se digita ESC, internamente contien un repeat, para verificar que tecla de función se ha digitado, esto lo hace con un Case, dentro de cada Case hay llamados a sus procedimientos respectivos.

3.- UTILIZACION DEL PROGRAMA.

Para utilizar al programa se debe llamar al módulo ejecutable, una vez ingresado lo primero en visualizarse es la pantalla de presentación, el usuario deberá oprimir la función deseada.

La introducción de las líneas es automática y luego espera la introducción del comando:

```
1 comando1;

2 comando2;

3 etc...
```

Si el usuario desea salir de la edición simplemente digita un espacio en blanco, luego si lo desea puede grabar lo editado, o puede realizar las modificaciones del caso. Si el texto se modifica, al grabar se grabará lo modificado al último.

3.1.- COMANDOS DEL SISTEMA.

Los comandos de control del sistema son los que permiten el control total de la planta , estos comandos son los siguientes:

Apagar, Prender, Mantener, Esperar.

Estos comandos tienen opciones o argumentos; en el caso del comando Apagar y Prender los argumentos son:

"a": agitador

"b": bomba de recirculación

"c": calentador

"v": válvula solenoide

Para mantener y esperar los argumentos son:

"T": temperatura del agua

"t": tiempo

Cada argumento representa el elemento físico de control en la planta, en el caso del tiempo y la temperatura, sirven para la verificación del estado de la planta.

3.2.- ESCRITURA DE LOS COMANDOS

Los comandos deben escribirse con una lógica adecuada, ésta permitirá una buena compilación del mismo, en el caso de una mala edición de los comandos, la compilación será negativa y será necesario repetir todo lo anterior.

La gramática de escritura de los comandos es la siguiente:

* digitar el comando deseado
* sin espacio en blanco colocar un paréntesis de apertura
* colocar la opción deseada, con minúscula
* cerrar el parèntesis
* colocar como cierre de comando el punto y coma (;)
* Toda la sentencia debe ser escrita en minúscula
* Para salir de la edición, simplemente dejar un espacio en blanco
* después de cada comando, se avanza a la siguiente linea con <ENTER>

```
 1 apagar(a);
 2 apagar(b);
 3 apagar(c);
 4 apagar(v);
 5 mantener(t=10);
 6 esperar(t=20);
 7 mantener(T=30);
 8 esperar(T=90);
 9 prender(a);
10 prender(b);
11 prender(c);
12 prender(v);
```

Como podemos apreciar para tiempo y temperatura la lógica es una sentencia de igualdad, con ésta lógica es posible realizar una comparación y proceder a un mejor control.

4.- CONDICIONES ESPECIALES

El programa original, fue modificado por cinco ocaciones, la versión del editor para nuestro caso es la 5.0.

Es posible ampliar más características, tal es el caso de imprimir el archivo que se editó, pero lo más principal es el aumento del número de lineas a editarse, nuestro caso fue limitado, pero para una planta más grande es posible crear un mayor número.

Para el perfecto funcionamiento del programa, se lo insertó dentro del programa Master, el cual incluye el editor, el compilador, y la carga al sistema; ésto se hace debido a la interrelación de ambos programas.

La versión de PASCAL que se puede utilizar es la 3.0; la 4.0; la 5.0 y la 6.0, todas son de fácil adquisición y manejo, el lenguaje assembler es de la IBM PC (8086); se lo creo cada programa independientemente, si segmento de pila, para el caso del traspaso de variables, éstas necesitan una dirección offset adicional para una mejor comunicación entre PASCAL y ASSEMBLER.

El programa final fue escrito y compilado en PASCAL versión 6.0 la forma de llamar a los procedimientos externos es la siguiente:

* los procedimientos externo se deben compilar como .OBJ
* luego del nombre del programa va el llamado a ellos
* se los llama con la directiva $I

```
Program editor;

  {$I BAJAR.OBJ}
```

Mediante esta directiva, al compilar el programa se carga automáticamente cada procedimiento externo llamado, y lo compierte en un sólo paquete executable, sin nesecidad de tener los mismos dentro de la unidad de trabajo.

Para el manejo de los archivos se debe cambiar la configuración del sistema usado, en el archivo Config.SYS, para esto se lo hace mediante un editor o el EDLIN, se debe aumentar la siguiente linea:

```
files=50
```

Con esto permitimos a la máquina el manejo libre de un número grande de archivos, sin error de apertura.

5.- CONCLUSIONES

El editor es un program de fácil manejo, permite al operador

una ágil manera de escritura de comandos y su presentación es vistosa, para no crear fatiga en el operador.

Es posible cambiar la estructuración, y no utilizar arreglos de registros, en su caso podemos usar punteros a manera de listas enlazadas, cuya agilidad es mucho mayor y de más flexibilidad.

En las nuevas versiones del lenguaje PASCAL, vienen incorporadas funciones que realizan el mismo trabajo que los procedimientos externos realizados en ASSEMBLER, por eso se recomienda mejorar la calidad creando un programa en un sólo lenguaje, lo cual permite mayor agilidad de trabajo.

No olvide el operario que sus archivos a grabar deben ser de extensión . SCP para que al realizar la compilación, ésta no se caiga.

Para la utilización de la diversas funciones, siga cada paso que se le indica en el momento de llamarlas; en el caso de inhibirse la máquina la única forma es resetear la máquina y empezar otra vez; si tiene dudas sobre el funcionamiento o problemas con el mismo, consulte el Manual de operación del CYCIT o con el creador del mismo.

CAPITULO 2

COMPILADOR

1.- INTRODUCCION

Se requiere tener el control de una planta externa desde un cuarto de control, a partir de un computador, desde el cual se desea controlar un proceso que para nuestro caso, se trata de un control de temperatura.

Para el efecto se ha implementado algunas sentencias de control, todo gobernado por medio de un lenguaje de procesos, en el cual existen algunos comandos que permiten tener el control de la planta.

Para que sea efectivo todo este proceso se ha diseñado un compilador de lenguaje de procesos, el cual transforma o traduce un lenguaje fuente, en un codigo objeto.

Durante el proceso de compilación se tiene que ir descubriendo cualquier tipo de error que se pueda haber pasado en la etapa de edición.

Sólo si existen cero errores en el editado de la sentencias de control, se procederá a la creación de un archivo con la extensión **.ASC** , el cual será un archivo en código decimal para luego ser transmitido y ser entendido por el controlador , que para nuestro caso es un microprocesador 8080.

2.- CONTENIDO

Dentro del Proyecto Final del TOPICO ESPECIAL DE GRADUACION se ha implementado en lo que es el diseño del compilador, lo que corresponde al editor de pantalla y la parte detectora de todos los errores encontrados , en el proceso de compilación.

Estos errores encontrados en las sentencias de control, son analizados linea por linea, y a medida que se los encuentra, se irán publicando, con el detalle adecuado de cada uno de ellos. El proceso de compilación resultará exitoso, siempre y cuando no se encuentre ningun error al digitar las lineas correspondientes en lenguaje de procesos, es decir se creará un archivo entendible solo por el controlador en este caso el 8080.

2.1.- DIAGRAMA DEL PROCESO DE COMPILACION

La manera de manejar todo el proceso de compilación está implementada en la figura 2.1.

2.2.- DETECCION DE ERRORES

El programa de compilación está hecho de la siguiente manera:

Como se tiene que detectar todo tipo de error en las sentencias se procedió ha, determinar como deberian ir las sentencias.

Por ejemplo los comandos que se han implememntado para verificar, la existencia o no de errores fueron :

prender

apagar

esperar

mantener

Estas sentencias de control serán analizadas , y verificadas por el programa , para evitar que se sucedan errores de léxico.

El programa comenzará a analizar , e irá anunciando los errores respectivos, estos errores, al irse encontrando se irán directamente a grabar en un archivo cuyo nombre será el mismo que el archivo que se procedió a compilar, pero tendrá la característica de tener la extensión ".lst" , archivo en el cual se observarán los respectivos errores, correcciones, número de línea del error encontrado, cada error deberá ir con su respectiva corrección.

La forma en que se analizan los errores, se la implementó de la siguiente manera:

Se grabó en un arreglo de caracteres todas las sentencias posibles con los respectivos argumentos, es decir solo se permitirá como sentencias correctas lo siguiente:

prender(a); apagar(a); esperar(T>xxx);

prender(b); apagar(b); esperar(t=xx);

prender(c); apagar(c); mantener(T,t=xx);

prender(v); apagar(v);

Deberán ir en cada línea, ya que así se va leyendo las sentencias y deberá ir al final el " ; " como delimitador de instrucción.

2.3.- ELEMENTOS A CONTROLAR

Dentro de los elementos a controlar en la planta tenemos, que:

a, b, c y v representan:

a: AGITADOR

b: BOMBA DE RECIRCULACION

c: CALENTADOR

v: VALVULA SOLENOIDE

2.4.- COMANDOS IMPLEMENTADOS

```
prender(a);
```

Con esta sentencia se procede a prender el **agitador**, el cual tiene como misión, tener homogénea la temperatura en el agua que se encuentra en el interior de un tanque, se debe notar que al final de cada sentencia debe ir el punto y coma como delimitador de instrucción. El arguemento de prender debe ir entre paéntesis.

```
prender(b);
```

Con esta sentencia se procede a prender la **bomba de recirculación** la misma que tiene por función hacer circular agua a través de la camisa para mantener la temperatura constante por un cierto tiempo.

```
prender(c);
```

Con esta sentencia se procede a prender el calentador, el cual tiene como misión calentar el agua que se mantiene en el interior del tanque.

prender(v);

Con esta sentencia se procede a prender la válvula solenoide , la misma que sirve para activar la válvula que abrirá el compartimiento , para así poder activar la bomba de recirculación que hará circular el agua a través de la camisa que rodea al tanque.

Para apagar todos estos dispositivos se procede con el comando **apagar** , por ejemplo:

apagar(a); apagar(b); apagar(c); apagar(v);

las mismas que procederían a apagar el agitador, bomba de recirculación , calentador y válvula solenoide respectivamente.

esperar(t=xx);

Donde el argumento de esperar es el tiempo y solo es permitido digitar entre el tiempo y su valor el operador " = " , y solo se permite escribir 2 digitos del 0 al 9; de no ser asi se publicarán los mensajes respectivos, asi mismo deberá ir el respectivo " ; ".

Ademas otra sentencia puede ser :

esperar(T>xxx);

Donde T representa la temperatura , y es permitido como operador el signo " mayor que " , " > " , **xxx** representa el valor que toma la temperatura, dicho valor será escrito en decimal, y es posible poner 3 digitos del 0 al 9.

Asi mismo deberá ir el punto y coma como delimitador de instrucción.

mantener(T,t=xx);

Cuya función es la de mantener un cierto nivel de temperatura por un tiempo determinado por **xx** , el mismo que representa un tiempo dado en segundos.

La eficiencia de mantener(T,t=xx); resulta solo si se ha llegado hasta una temperatura, dada por T, por ejemplo, sea la secuencia de control de la siguiente manera:

```
prender(c);
    .
    .
    .
esperar(T>xxx);
mantener(T,t=yy);
    .
    .
```

Luego de haber esperado por que la temperatura se mayor a xx, y luego de haberse sensado esto, si la siguiente sentencia de control es mantener(T,t=yy);, se consigue que se mantenga la temperatura dada por un cierto tiempo, determinado por t=yy , donde **yy** son dígitos del 0 al 9 y representan un tiempo dado en segundos.

Es importante saber que todos los argumentos deberán ir entre paréntesis.

Todas estas instrucciones estan grabadas en arreglos de caracteres, ya que en el momento de proceder a detectar los errores , se procederá a comparar simplemente caracter por caracter, las sentencias estan grabadas en un arreglo de

records , dicho record tiene una dimensión de soportar 100 sentencias, es decir 100 lineas a ser compiladas, y cada linea tiene una capacidad de 20 caracteres.

Al ser digitadas las sentencias , no se podrá comenzar con un espacio en blanco, antes de las 100 lineas , pues al escribir un espacio en blanco como inicio de sentencia, lo que resultará es que se romperá la secuencia y ya no seguirá pidiendo más sentencias.

2.5.- CONVERSION A CODIGO OBJETO

El proceso de compilación está hecho de tal manera , que sólo se consigue crear un archivo con la extensión 'ASC ' si y solo si , se ha conseguido editar un archivo de sentencias de control , con cero errores .

Si todo el proceso es correcto se crearian tres archivos los cuales serian por ejemplo:

SENTENCIAS.SCP : en el cual se edita un archivo que contenga sentencias de control.

SENTENCIAS.LST : en el cual se listan todos los errores cometidos en el archivo en el que constan las sentencias de control.

SENTENCIAS.ASC : el cual contiene la información de las sentencias editadas, pero en código entendible unicamente por

el controlador el 8080.

Este código se ha establecido , que sea código DECIMAL. El diagrama de flujo del **PROCESO DE COMPILACION** , se muestra en la fig. 2.2

CAPITULO 3

COMUNICACION ENTRE MICROS.

1.- INTRODUCCION.

La **Comunicación entre Micros** , la vamos a realizar con la ayuda de un SDK - 8080 , una P.C. y una P.S.60 . Todos ellos son micros a los cuales se les creara **Programas de Comunicación** , con el fin de hacer acsecible la comunicación entre ellos.

El micro SDK - 8080 , se encuentra dentro de la **Unidad Terminal Remota** , que a su vez esta pertenece a la **Planta.** En cambio la P.C. representa el **Control Maestro** , que supervisa el buen funcionamiento de la planta .

Por ultimo la P.S.60 es el que archiva los resultados de los estados de la planta para hacer análisis de **ESTADISTICA** de la producción de la planta.
La Comunicación estará comprendida en :

1.- **Comunicación PLANTA - CONTROL .**

2.- **Comunicación CONTROL - ESTADISTICA .**

Para realizar la comunicación PLANTA - CONTROL, los micros a utilizar son el SDK - 8080 y la P.C.
Y para la comunicación CONTROL - ESTADISTICA se utiliza la P.C. y laP.S.60.

En la comunicación Planta - Control , nosotros vamos a transferir **ORDENES e INFORMACIONES** . En cambio en la comunicación **Control - Estadistica** , se va a enviar **RESULTADOS** actuales de la planta .

Las ordenes son programas de control de la planta ,que son creadas por el Operador de la P.C. con el fin que la planta realice un trabajo sistematizado durante el dia de operación . Para que ello se cumpla deberemos enviar este programa de control en forma de un archivo desde la P.C. hacia el SDK - 8080 . Las **Informaciones** , son datos actuales de la planta que llegan continuamente al Control Maestro ,para ser mostrados en la pantalla mimica del monitor de la P.C. , con el fin que el operador controle la planta .

En cambio los **resultados** , son el envio desde la P.C. hacia la P.S.60 de los datos actuales de la planta recabados durante un cierto tiempo , para un análisis de Estadistica de la producción de la planta .

Con la ayuda de esta breve introducción al tema de la comunicación entre Micros , el lector podrá darse una imagen de como el Control Maestro controlará a su voluntad el desarrollo diario de la planta .

2.- **CONTENIDO**

2.1.- Comunicación PLANTA - CONTROL :

Como se dijo anteriormente, deberemos hacer Programas de Comunicación , tal que nos permitan hacer acsecible la comunicación entre los micros (SDK - 8080 y la P.C.) .

La comunicación entre **la Planta** y **el Control** es vía Telefónica con la ayuda de los Modems , en el cual es necesario dejar enlazado la comunicación , utilizando un **Programa de comunicación** el cual me permita de un modo Automatico hacer la llamada desde el Control Maestro.

Aqui el Programa que utilizamos se llama **BITCOM** , Luego de ello el operador procede a crear el Programa de Control de la Planta para almacenarlo en un Archivo , previo al envio a la Planta.

Entonces el Programa de comunicación , deberá ser capáz de enviar el Archivo cuyo contenido es el Programa de Control de la Planta , hacia las localidades de Memoria del SDK-8080.

El **programa de comunicación** lo vamos hacer en Pascal, el cual me permitira llamar a Subrutinas hechas en Ensamblador, siendo este un lenguaje de mas bajo nivel que el Pascal dandonos la facilidad de manejar la comunicación a

través del puerto serial 2 .

La subrutina externa utilizada tiene como nombre **COMUNICA** y esta me permite realizar previamente la configuración del puerto serial 2 e ir transmitiendo el caracter tomado del archivo, hacia la Planta.

De acuerdo al Diagrama de flujo de este Programa (Fig. 3.1) el cual abrimos el archivo cuyo contenido es el Programa de Control en modo de lectura e iremos sacando caracter por caracter y lo enviamos a la planta hasta que el Operador le mande un código de parada para que el 8080 deje de recibir y transmitir al control maestro un código de éxito , en la comunicación.

Al enviar el caracter a la planta, estamos utilizando la subrutina externa hecha en Ensamblador el cual nos permite conducir la comunicación a tráves del puerto serial 2 , por medio de los parametros de comunicación que utilizamos. Para facilidad , el programa de comunicación se lo inserto en el programa principal el cual puede editar,compilar y cargar el programa de control de la Planta. En caso de falla en la comunicación el operador del Control Maestro no podrá recibir de la Planta el código de exito de la comunicación, y deberá otra vez enviar el programa de control hacia la planta, hasta que el control maestro reciba de la Planta el código de éxito de la comunicación.

Cuando el Programa de Control este ya almacenado en la memoria del SDK - 8080 , este procedera a activar la planta para despues ir Sensando continuamente los estados de la Planta , para de imediato enviar las Informaciones de los Estados Actuales de Planta hacia el Control Maestro (P.C.) donde seran mostrados al operador en el monitor de la P.C. en forma de grafico Mimico de la Planta, con el fin de controlar los estados de Planta en forma visual por intermedio del operador.

Para poder recibir la información de la Planta ,deberemos hacer uso de **otro Programa de comunicación** que asimismo estara hecho en Pascal, el cual llamara a Subrutinas externas en Ensamblador , que nos permitiran ver el **Status** del puerto serial 2 , **Recibir** y **Transmitir** , informaciones y ordenes respectivamente.

Entre las Subrutinas Externas utilizadas para este Programa de comunicación son:

- **RECIBA.COM**
- **TRANSMITA.COM**
- **ESTATUS.COM**

RECIBA:

Esta subrutina externa , recibe un caracter del puerto

serial 2 , en el cual podemos ingresar las informaciones de la Planta.

TRANSMITA:

Esta subrutina externa es usada para transmitir un caracter a travez del puerto serial 2 , especificado desde el programa principal en pascal.

ESTATUS:

Esta subrutina externa es usada para ingresar valores significativos del estatus del puerto serial 2 , especificado desde pascal en Dato Preparado y Error de Alcance .

De acuerdo a los Diagramas de flujos mostradas en las figuras 3.2a , 3.2b , 3.2c y 3.2d , el lector se dara una idea mas clara del diseño de este programa de comunicación.

Para facilidad del operador , se tuvo que insertar en el programa de la Visualización al programa de comunicación con el fin de que al recibir la informaciones desde la Planta no se interrumpa el proceso de Graficación de los Estados Actuales de la Planta. De acuerdo al Diagrama de Flujo (Fig. 3.2a) el programa de comunicación inicia un rastreo en el puerto serial sobre la presencia de un código inicial que me permita asegurar que la información de la Planta esta presente, esto me asegurara que la visualización no se me interrumpa.

El SDK - 8080 nos mandara continuamente hacia el control maestro lo siguiente:

- Codigo Inicial.
- Paquete de Información.
- Codigo Final.

Una vez que la P.C. logre la captura del código Inicial procedera a mostrar en la pantalla mimica , el Paquete de la Información de la planta, que son los Estados Actuales de la Planta para luego ser almacenados en el archivo Planta.dta ,a continuación recibe el código Final que significa que el paquete de la información ha sido enviado , por lo que el programa hará que la P.C.envie al SDK-8080 un código de Exito en la comunicación .Si el operador detecta en la Pantalla Mímica que los Estados Actuales de Planta son incorrectos , este mandara una Orden de Apagar la Planta ,mediante el Uso de la Tecla ESC , El cual Corresponde a un código que es reconocido por el 8080 , para cambiarse al Control Manual y asi apagar la Planta.

El proceso de la recepción de la información de la Planta sera de forma continua, hasta que haya transcurrido un tiempo previamente determinado por el operador para que deje de recibir mas informacion de la Planta y mande a hacerla transferencia del Archivo Planta.dta a la P.S.60 en un Archivo Temphora.dat .

Una vez hecha la transferencia la P.C.volvera a Almacenar nuevos Datos de los Estados actuales de la Planta.

2.2.- **Comunicación CONTROL - ESTADISTICA :**

La comunicación a realizarse, utiliza **2 Computadoras** ,la P.C. y la P.S.60, su conexion es a través del cable fisico RS-232C en los conectores DB-25.

Es menester mencionar los pines del conector DB-25 que se utilizaron son los siguientes:

PIN 2 (TRANSMISOR DE DATOS).
PIN 3 (RECEPTOR DE DATOS).
PIN 4 (REQUEST TO SEND).
PIN 5 (CLEAR TO SEND).
PIN 6 (DATA SET READY).
PIN 7 (SENAL DE TIERRA).
PIN 8 (CARRIER DETECT).
PIN 20 (DATA TERMINAL READY).

P . C .			P . S . 6 0	
P I N	2	========>	P I N	3.
P I N	3	========>	P I N	2.

P I N	4	========>	P I N	5.
P I N	5	========>	P I N	4.
P I N	6	========>	P I N	20.
P I N	7	========>	P I N	7.
P I N	20	========>	P I N	6.

Los pines **6** y **8** de cada lado deberan estar **puenteados.**

Los parametros de comunicación de las 2 computadoras deberan ser identicos , pero pueden ser seleccionados por ellos, la selección que puede escojer son los siguientes:

- 1 bit o 2 bit de **PARADA.**
- **Paridad** PAR o IMPAR.
- **VELOCIDAD:**
  - 300 bauds.
  - 1200 bauds.
  - 2400 bauds.
  - 4800 bauds.

Todos los parametros de comunicación para ambas Computadoras están referidos al Puerto Serial 1.

El programa de comunicación lo vamos a hacer en lenguaje **TURBO C de MYCROSOFT V 6.00,** siendo este un lenguaje de mayor nivel que el Ensamblador, el cual para este programa no sera utilizado .

El programa de comunicación de la P.C. nos permite hacer la transmisión del Archivo Planta.dta, en cambio el programa de comunicación de la P.S.60, nos permite ir recibiendo caracter por caracter e ir Almacenandolo en un Archivo Temphora.dat.

De acuerdo a los diagramas de flujo (Fig.3.3 y 3.4) de los programas de comunicación , el programa emisor o receptor podra esperar al otro hasta que el enlace de la comunicación sea establecido , de alli el programa emisor abrira el archivo en modo de lectura, y enviara caracter por caractera la P.S.60 , entretanto el programa receptor de la P.S.60 hará que los datos que llegan sean almacenados en un archivo donde estaran los datos actuales de la Planta recabados en un tiempo determinado por el operador.
Cuando el programa emisor haya enviado el archivo Planta.dta , enviara un código de parada el cual el programa receptor de la P.S.60, lo reconocera como fin de archivo y dejara de recibir la P.S.60 mas datos actuales de la Planta , y a su vez la P.C. dejara de transmitir y volvera a recibir nuevos paquetes de informaciones de la Planta para mostrarlos en la Pantalla mimica.

Cuando Ocurre la comunicación entre la P.C. y la P.S.60 las pantallas de ambas computadoras quedaran interrumpidas ,en la P.C. , la pantalla mimica de la P.C. quedará fijado en el último paquete de información de la Planta.

Cuando la comunicación haya terminado las 2 Computadoras volveran ser habilitadas, procediendo en el trabajo de otras tareas.

Y asi continuamente, luego de otro tiempo determinado, volver a ocurrir la comunicación entre ambas computadoras. Si no existió falla en la comunicación , el programa receptor mostrara un mensaje comunicación buena .

3.- CONCLUSIONES :

El proceso de la comunicación entre micros , nos permite lograr una comunicación eficaz entre la Planta y el Cuarto de Control , en la que el Control Maestro manejara a voluntad de un Programa de Control , creados por un Personal de ingenieria capacitados para el manejo de la Planta.

En base a este prototipo, se puede pensar en el control de varias Plantas, a la cual el control deberia ser en un orden sequencial determinado por un seleccionador de llamadas.

Las Interrupciones del BIOS, son las más utilizadas en los Programas de comunicación tanto en Turbo C , como en los Programas Pascal-Ensamblador.

CAPITULO 4.

COMUNICACION SERIAL ASINCRONICA.

1.- INTRODUCCION

Para que la estación maestra controle el proceso diario de producción de una planta necesita un método de enlace de comunicación por el cual se establesca el diálogo entre estas y así intercambiar información de estado proveniente de la planta y comandos que envía la estación maestra.

Dicha información deberá ser manejada de manera correcta, precisa y coordinada para que no se produzcan prdidas de información o errores en los datos que viajan y que podrían conducir a interpretaciones incorrectas de un estado actual o a la mala ejecución del proceso en la planta.

El método en base al cual se implementará la comunicación es half - duplex de comunicación serial asincrona acondicionado para el caso específico a tratarse.

Half - Duplex se justifica debido a que la comunicación solo se produce en un sentido a la vez. El hecho de escoger el método asíncrono es porque podrían haber largar pausas entre caracteres de información u ocacionalmente seguir rapidamente uno tras otro.

Para el enlace entre la Unidad Terminal Remota y la Estación Maestra se ha elegido la comunicación vía telefónica a través de Modems. Más adelante se justifica esta elección y se

presentan posibles alternativas.

Este método como cualquier otro a implementarse podría en determinado momento presentar inconvenientes de operación. Este hecho no debería ser causa para interrumpir el proceso en la planta por lo que debería existir una alternativa de control emergente.

Para casos así se posee la operación de controlar manualmente la Planta desde el mismo lugar en que esta se encuentra.Se posee un programa de control manual grabado en la cinta de un casette por lo que también hay que establecer comunicación entre un tocacintas y el procesador que maneja la Unidad Terminal Remota o Planta.

Los programas que controlan la comunicación están hechos de una manera básica y siguiendo una secuencia de flujo sencilla para adaptarse a cualquier método de enlace.

Los circuitos de acondicionamientos necesarios son también suministrados para adaptar correctamente los dispositivos a operar.

2.-COMUNICACION ENTRE ESTACION MAESTRA Y UTR

Existen algunos métodos por los cuales se puede establecer este enlace pudiendo ser via radio,por medio de cable coaxial

o vía telefónica a través de modems.

El enlace vía radio tiene como desventaja el elevado índice de ruido que puede introducirse y el alto nivel y costo de la instalación y mantenimiento, posee también limitaciones en la banda de frecuencia usada.

El cable coaxial es más económico que el enlace vía radio , introduce menos ruido en la transmisión pero tiene como inconveniente la necesidad de hacer un tendido del cable para lo cual se requiere permiso en las tierras o derechos de vía a lo largo de toda la ruta, así como acceso posterior recurrente para el mantenimiento.

Se ha elegido entonces el enlace telefónico a través de modems principalmente debido a la disponibilidad de líneas de coneccionado telefónico y bajo costo de mantenimiento de equipos.

El principal inconveniente que podría producirse es el grado de servicio que preste la central telefónica que maneja los circuitos de abonado.

La implementación de este método requiere de dos modems, uno en la estación maestra y otro en la UTR, y dos líneas o pares telefónicos por lo que se establecerá la comunicación.
El esquema general de coneccionado se encuentra en la figura

4.1.

La Estación Maestra necesita de una interface serial standard RS-232C para conectarse a su modem mientras que en la UTR existe un cableado adecuado para conección con el modem. Un computador IBM - PC es el que va conectado al modem en la Estación Maestra y es el KIT 8080 el que está en la UTR.

2.1.- CONECTANDO EL MODEM AL COMPUTADOR IBM - PC

EI modem irá conectado al computador de la forma DTE (Data Terminal Equipment), a DCE (DATA COMUNICATIONS EQUIPMENT).

DTE del lado del computador IBM - PC y DCE del lado del modem como lo indica la figura 4.2.

2.2.- CONECTANDO EL MODEM AL KIT 8080

Se efectúa un cableado sencillo entre el KIT 8080 y el modem La salida desde el programa de escritura, debe ser en el PORTOCO (en bit menos significativo de la dirección del puerto 02) y la entrada es recibida en PORTOBO (el bit menos significativo de la dirección del puerto 01).

En la UTR el KIT 8080 maneja niveles TTL que se adaptan a

niveles RS-232C a través de los integrados 1488 (QUAD LINE DRIVER) y 1489 (QUAD LINE RECEIVER).

Un diagrama apropiado de coneccionado es la figura 4.3.

3.- COMUNICACION ENTRE UTR (KIT 8080) Y TOCACINTAS

Este enlace es necesario en caso de emergencias en que se requiera cargar el programa del control manual de la planta . Para esto se hace uso de la misma entrada y salida indicada anteriormente pero pasando a través de un circuito de acondicionamiento de señal que posee el KIT 8080 .

La instalación para su operación se encuentra en la figura 4.4.

La conección OUT-AUX se usa cuando se desea grabar información almacenada en el KIT 8080 al cassette.

La conección IN-EAR es usada para cargar un programa, control manual en nuestro caso, grabado en le casesette a la memoria volátil del KIT 8080.

El operador tendrá que ejecutar una breve secuencia coordinada de instrucciones para realizar cualquier acción

antes mencionada.

4.- DEFINICION DE LA TRANSMISION ASINCRONA

En la transmisión asincrona cada caracter es independiente del otro y arastra su propia palabra de marcación.

El convenio adoptado es que cada caracter dato sea precedido por un cero seguido por uno o más intervalos de señal uno.

Después de ciertos periodos de tiempo sin recibir datos, por ejemplo una señal de unos constantes, el receptor lee una trasición a cero .

Esto señala el comienzo de un caracter , y el receptor sincroniza su reloj.

El dato es transmitido dentro de un caracter, siendo enviado el menos significativo primero, asi la secuencia es :

bit inicio

bit cero

.

.

.

.

bit siete

bit parada

(es opcional el uso de más bits de parada)

Este formato para comunicación asíncrona ha sido adoptado por el AMERICAN NATIONAL STANDARDS INSTITUTE y por la CCITT.

5.- PROGRAMA DE TRANSMISION - RECEPCION

Antes de entrar en detalle de manera como se afrontó la realización del programa de transmisión-recepción es de anotar, que el esquema aqui utilizado es el mismo tanto para la comunicación con el modem hacia la Estación Maestra o para la comunicación con el tocacintas.

Por supuesto que los programas deberán ser adecuados para cada efecto tanto en su estructura final como en la ejecución de los mismos .

Las variables especificas del programa como son los parámetros de comunicación serán también los correspondientes para cada caso.

Los diagramas de flujo de los programas de comunicación son básicos para ambos tipos de enlace y se encuentran en las figuras 4.5-a, 4.5-b, 4.5-c, 4.5-d, 4.5-e.

5.1.- TRANSMISION SERIAL

Esta rutina de servicio transmite cada bit a medida que va siendo ingresado al programa . Este programa llamará una subrutina para generar un tiempo de retardo, y usará un llamado para dar paso al servicio de interrupción al final de cada periodo de retardo.

El programa principal carga necesariamente bytes desde la memoria o algun dispositivo de lectura y pasa a través de la rutina de servicio de interrupción .

Puesto que esto es un sistema real de interrupción, el programa principal no tiene concocimiento de cuando ocurre esto, el dato debe pasar a través de un arreglo de localidades de memoria y el servicio de interrupción debe indicar cuando ha terminado con un byte y necesita otro .

La rutina de servicio de interrupción debe enviar tanto el bit de inicio como el de parada al igual que los bits de datos.

Una manera simple de manejar esto es guardando un modelo de doce bits, estos serán desplazados un bit a la vez por el servicio de interrupción .

Cuando el modelo este vacío la rutina de interrupción y el

programa principal sabrán que el caracter ha sido recibido. No se enviarán más bits hasta que el programa principal guarde los nuevos modelos .

MODELO DEL CARACTER DATO.

El dato modelo es grabado en dos bytes tal como sigue:

El programa principal carga y almacena estos a través de las tres siguientes instrucciones :

```
Mov l,a     Dato
Mvi h,---   Bit de parada
Dad h       Ingrese el bit de inicio
```

El servicio de interrupción desplazara el dato hacia la derecha y sacará sucesivamente cada bit hasta dejar el modelo vacio (Todo cero).

5.1.1.- **RUTINA DE SERVICIO DE INTERRUPCION**

En la rutina de servicio de interrupción se observa que la salida esta definida en el portoco.Ademas el bit dato es copiado en el portoc2, el mismo que controla el indicador de

acarreo .

```
Sbb a          Copia todos los bits en C1
Ani 05         Enmascara los bits 0,2
Ori 02         Fija en uno el bit 1
Out portoc     Habilita el monitor
```

Dado que el monitor automáticamente almacena la dirección 8228 como una dirección de salto para RST 5,es necesario como paso prewvio que el programa traslade los comandos del servicio de interrupción hacia esa localidad y evitar el uso de las instrucciones antes mencionadas.

6.- RECEPCION SERIAL

6.1.- RECEPCION ASINCRONA

Este programa es un poco más complicado que el de transmisión en el formato serial, ya que dos funciones de entrada son necesarias:

a) Esperar por el bit de inicio
b) Recibir sucesivamente los bits de datos

El bit de inicio es inicialmente detectado cuando el bit de dato recibido cambia de uno a cero (De una parada o condición de no dato a bit de inicio).

Este debe ser reconocido prontamente para obtener la

sincronización, entre el dato entrante y el dispositivo de tiempo del receptor o el lazo de retardo.

Al usar interrupciones para la recepción de datos debemos generar una interrupción al comienzo del bit de inicio para obtener sincronización, retardada un tiempo igual a medio bit, en la mitad del bit de inicio y luego retardar un tiempo igual a un bit en la mitad de cada sucesivo bit hasta que todo el caracter ha sido recibido.

6.2.- ESPERA DEL BIT DE INICIO

La subrutina mostrada lee respectivamente la entrada del portobo y espera por el bit de inicio:

```
* IN    portobo
  RAR
  JC  *
```

Las tres instrucciones señaladas toman alrededor de 24 periodos de reloj, o aproximadamente doce microsegundos.

Esto proporciona la precisión necesaria para detectar el flanco del bit de inicio en ratas bajas de datos.

Luego la detección consiste en retardar medio periodo de bit y asegurar que la entrada está aún baja .

Esto evita sincronizar la recepción con un alto momentáneo. Como una confirmación de que el bit receptado es legítimamente el de inicio, la señal debe permanecer baja medio periodo de bit .

Nuevamente se utiliza la subrutina de retardo DELYT para retardar un tiempo correspondiente a un bit y retornar en el acarreo el bit ingresante.

Una entrada alternada retarda medio tiempo de bit .

Usando DELYT se evita la necesidad de cambiar el valor del registro C_3 para retardar un periodo completo o medio periodo del bit.

6.3.- RECEPCION DE LOS BITS DE DATOS

Luego de que el bit de inicio ha sido reconocido, el paso siguiente consiste en la recepción y almacenamiento de los ocho sucesivos bits de datos.

El registro A es utilizado tanto para el almacenamiento de los datos recibidos como para el conteo de los bits.

El bit de inicio es señalizado por el desplazamiento de un

uno en el bit siete .

Cuando este bit es desplazado hacia afuera del bit o sabemos que los ocho bits de datos han sido ingresados.

Es posible que se efectúe erróneamente la lectura del bit de inicio y por lo tanto, se sincronice inapropiadamente , de ser así, los datos recibidos serán basura.
Como una manera de protección se asegura la correcta recepción del bit de parada.

Cuando una cadena continua de caracteres está siendo recibida es posible arrastrar un error de tiempo que se propagará a los siguientes caracteres .

La sincronización podrá efectuarse, no al detectar el bit de inicio sino en una transición de uno a cero por parte del dato .

Sin embargo esto es prontamente detectado al aparecer un cero (parte del dato) en lugar del bit de parada .

Al realizar la detección del bit de parada se percata este tipo de error.

La subrutina retorna el dato en el acumulador y el bit de parada en el acarreo. así si el acarreo no está en alto, un

error ha sido detectado. La bandera cero estará en alto gracias a DELYT y así se retornará a la subrutina de recepción .

7.- LAZO PRINCIPAL

El programa de comunicación entre Estación Maestra y UTR se ejecuta de manera "automática" a medida que corre el programa monitor de control de la UTR.

En realidad no existe un lazo principal que controle este mencionado enlace .

Transmisión y recepción son subprogramas o subrutinas cuyos llamados son hechos en forma dinámicas y solo en el instante en que uno de ellos es requerida.

Lo que si hay que anotar es que antes de efectuar el llamado a cualquiera de las dos subrutinas, transmisión o recepción hay que asignar los parámetros adecuados para la comunicación especificada.

El lazo principal del programa de comunicación con el tocacintas permite al operario elegir entre transmisión y recepción mediante la entrada de la dirección de inicio y fin

de los datos a ser transmitidos o el ingreso de la dirección donde serán almacenados durante la recepción .

Para el ingreso de datos utilizamos la subrutina del monitor ENTWD. También existe una subrutina de retardo compartida entre los programas de transmisión y recepción denominada DELYT.

Estos módulos que se han desarrollados son efectuados a través del monitor utilizando las funciones de grabación de cinta propias del 8080 . (ver figura 4.6).

8.- SUBRUTINAS EMPLEADAS

Las subrutinas comunes implementadas para ámbos tipos de comunicación son :

TRASLADO.- Efectua el traslado del bloque de servicio de interrupción ubicado en la ROM a la memoria RAM para su posterior ejecución .

DELYT .- Retarda un tiempo igual al fijado en el registro C.

DELYC .- Retarda un tiempo igual al fijado en el registro C, equivalente a la mitad del ejecutado por DELYT.

RCV .- Ingresa o lee un dato desde el puerto especificado.

La comunicación con el modem emplea las siguientes subrutinas

TRANS.EST.MODEM .- Envia el caracter (alfa) como inicio de trasnmisión, lee puertos para el envio del estado de la UTR, envia el caracter (omega) que es el fin de transmisión e inmediatamente llama a WAITS MODEM en espera del caracter (*) de respuesta.

Si se recibe un caracter distinto al (*) se efectúa inmediatamente una **parada de emergencia.**

TRANSMISION .- Transmite el byte almacenado en el acumulador, adecuando previamente los parámetros de comunicación .

WAITS MODEM .- Espera a que esté presente un dato proveniente del modem .

MENAB .- Rehabilita el sistema al pasar de transmisión a recepción .

El enlace con el tocacintas, emplea las siguientes subrutinas:

WAITS CASETERA.-Espera a que esté presente un dato proveniente del tocacintas.

ENTWD.-Ingresa dos bytes de datos como una dirección de memoria.

DELYT,DELYC,MENAB Y ENTWD,son subrutinas propias del monitor del KIT 8080.

En ambos casos de comunicación el programa de transmición hace uso del servicio de interrupción RST 5 al final de cada período de retardo.

9.- PARAMETROS DE COMUNICACION

Los parametros de comunicación son un punto importante a ser manejados en los programas.Deberán ser los mismos tanto en el programa de transmisión como en los de recepción respectivamente y tendrán que ser elegidos cuidadosamente atendiendo condiciones como distancias,facilidad de adecuación,tipo de enlace,etc.

Los parámetros de comunicación usados son los siguientes:

Para el enlace telefónico tenemos

Velocidad	1200 bps
Bits Datos	8
Bits Parada	2

Paridad	ninguna

Para el enlacen con el tocacintas

Velocidad	110 bps
Bits Datos	8
Bits Parada	3
Paridad	ninguna

10.- MODEM DE COMUNICACION

El modem empleado es el denominado EVEREX el cual posee algunas opciones de funcionamiento y su manejo se torna sencillo una vez conocido su alcance.

EVEREX posee dos programas utilitarios llamados CINSTALL y BITCOM. Ambos pueden establecer el enlace de comunicación de forma automática entre dos computadores que los ejecuten. Con el primero de ellos, CINSTALL, se puede verificar la correcta conexión del modem al computador o también comprobar si la conexión telefónica está bien realizada. Mediante el uso de los denominados comandos AT se pueden variar la forma de trabajo u otras funciones del modem.

BITCOM trae opciones de crear registros de enlace telefónico donde se fijan los parámetros de comunicación, puerto serial

usado y si el modem residente va a "llamar", "responder" o simplemente conectarse en "forma directa" (modem nulo).

En nuestro caso el modem conectado a la Estación Maestra es el que está en modo "call" de llamado y el que se encuentra en la UTR está en modo AA (auto answer) de respuesta automática.

El modem no necesita de un aparato telefónico para su funcionamiento. La llamada y respuesta se efectúa de manera automática y el enlace queda listo para establecer la comunicación de datos. La conexión al modem es por medio de un cable DB-25 RS-232C del cual solo son usadas las señales indicadas en la sección anterior.

11.- OPERACION Y EJECUCION DE PROGRAMAS

El enlace entre Estación Maestra y UTR se establece primeramente mediante el llamado del modem que se encuentra en la Estación Maestra. BITCOM es el que se encarga de efectuar dicho llamado al presionar D (dial) en el registro asignado para tal efecto.

Una vez establecido el enlace se pueden ejecutar comandos de DOS por medio de una de las opciones que posee BITCOM.
Del lado de la UTR su modem respectivo responde de forma

automática e inmediatamente contestada la llamada el operador debe echar a correr el programa de "control automático" almacenado en la ROM a partir de la localidad 0400, de la siguiente manera:

ADDR 0400

RUN

En este instante el procesador que controla la UTR ejecuta primeramente los comandos para recepción del modem.

Si se recepto correctamente o fue exitosa la comunicación se seguirá ejecutando el programa de "control automático" y el programa de transmisión a través del modem será invocado a medida que se produzca una variación del estado en la Planta.

Si en cualquier instante se interrumpe el enlace telefónico o si el programa de comunicación detecta algún error en la transmisión se mostrará en pantalla la letra C, para el primero de los casos, o ERR (error) en el segundo de aquellos. Cualquiera de las dos situaciones producidas son una sugerencia para que el operario ejecute el programa de enlace con el tocacintas para cargar el programa de "control manual" de la UTR.

El programa de enlace con el tocacintas permite elegir entre transmisión y recepción ejecutando una breve secuencia de

comandos específicos para cada caso.

La secuencia de ejecución, utilizando la subrutina normal del monitor para la transmisión, es como sigue:

Conecte la entrada AUX del tocacintas a la salida del modulador-demodulador del 8080. Encienda el tocacintas para grabar.

RESET
ADDR (dirección de inicio) MEM
ADDR (dirección de parada) BRK
ADDR 0371 RUN

El programa de transmisión utiliza el sistema de "puntos de ruptura" para finalizar la grabación.

Mientras el programa está siendo ejecutado, la pantalla quedará deshabilitada. Cuando los datos han sido grabados, la pantalla muestra 0382 CD. Se esperan dos segundos antes de apagar el tocacintas. El contenido de la dirección de parada no se graba en la cinta. En su lugar, un caracter de verificación de error es grabado. Por lo tanto, la dirección de parada debe ser la siguiente a la del fin del programa que se quiere almacenar.

Para leer el programa y guardarlo en memoria (recepción), se

debe escuchar primero la grabación hasta escuchar el tono previo al programa grabado. Se para la cinta y se conecta la salida EAR del tocacintas a la entrada del modulador-demodulador, luego

ADDR (dirección de inicio para grabar) MEM

ADDR O3AE

Prenda el tocacintas para transmitir

RUN

Es importante que hallan unos pocos segundos de tono constante antes de los datos. Mientras la cinta es leída la pantalla se deshabilitará, al final mostrará O3CF C5 (esto lo envía el monitor al finalizar la grabación).

12.- OTRAS CONSIDERACIONES

De acuerdo a las características específicas en cada caso, es posible usar cualquiera de los otros mencionados enlaces de comunicación. Los programas aquí empleados son fácilmente adaptables a ellos.

Las líneas telefónicas presentan inconvenientes en cuanto a

posibles interrupciones o bajo grado de servicio, estas no solo podrán transmitir datos sino también cualquier otra señal de voz por lo que pueden verse afectadas en cualquier momento. Se recomienda el empleo de una "linea dedicada" exclusiva para la transmisión de datos que puede ser solicitada a la institución local que presta el servicio telefónico.

La Estación Maestra puede requerir controlar no solo una UTR sino varias al mismo tiempo. Para ello puede seguirse usando el sistema telefónico a través de modems pero adaptando un circuito distribuidor de canales al modem ubicado en la Estación Maestra. Se necesitará también un programa que vaya verificando periódicamente qué UTR requiere o solicita servicio y asi establecer el enlace especifico.

Un esquema ilustrativo se da en la figura 4.7.
Claramente se observa que en este caso cada Unidad Terminal Remota deberá poseer su propio modem de comunicación.

Se podria evitar quizá el empleo de cassette y tocacintas para grabar el programa de control manual, este podria encontrarse ya alojado en una extensión de la memoria ROM del KIT 8080.

CAPITULO 5

CONTROL DE LA UNIDAD TERMINAL REMOTA

1.- INTRODUCCION.

El control de la Unidad terminal Remota debe ser realizado a través de un mecanismo inteligente no independiente que sea el encargado de captar, asimilar y procesar los comandos ejecutados desde la estación maestra, así como también de informar el estado de la planta y activar las señales de alarma.

La operación desde el centro de control es fundamentalmente dependiente de la adquisición de datos desde el sistema bajo control. Es solamente a través de evaluación manual o automática de los datos calculados que decisiones inteligentes pueden ser tomadas con respecto al sistema.

La UTR debe proveer seguridad en las operaciones y ser programable. Esto significa que una minicomputadora o microprocesador debe ser usado en vez de lógica de circuitos.

Las unidades programables ofrecen muchas ventajas sobre los tipos no programables: mejoras en costo/desempeño, reducción en el número de componentes (por ejemplo, tarjetas de circuitos impresos), capacidades mejoradas de diagnóstico, etc. Por otro lado y lo más importante, proveen la capacidad de control y proceso local.

Para este objetivo, se ha utilizado un microprocesador 8080 dado el portencial que ofrece para la grabción secuencial de eventos, control de lazo cerrado, colección local de datos, grabación de mantenimineto del equipo y operaciones locales bajo un costo poco significativo.

El hecho de ser una unidad programable permite el control a distancia y en caso de fallar esta, el control local de la planta, garantizando la continuidad del proceso. Para ello debe existir un programa base de control alojado en la estación remota que haga posible estos dos aspectos de funcionamiento.

Tanto el programa principal de control, como los de comunicación y subrutinas de control de la estación remota se encuentran alojados en la memoria ROM del microprocesador 8080.(ver figura 5.1).

2.- DEFINICION DEL SISTEMA.

Los aspectos de funcionamiento que deben ser manejados por la UTR son dictados de acuerdo a las necesidades del proceso que se va a controlar y son además las que determinarán las características del programa.

El proceso en cuestión cubre dos tipos de puntos de adquisición y control:

-Puntos analógicos: temperatura

-Puntos de estado/indicadores: nivel

-Puntos de estado/control: calentador

agitador

bomba

válvula p73

Los datos de temperatura y nivel son puntos de información para la ejecución del proceso, mientras que los cuatro dispositivos restantes serán además puntos de control.

(ver figura 5.2)

2.1.- PROGRAMACION DE LOS PUERTOS ENTRADAÑSALIDA .

Existen varias técnicas y dispositivos periféricos que pueden ser utilizados por el microprocesador 8080 para cubrir los aspectos de manejo de entrada/salida.

En el caso de los puertos de entrada/salida, cualquier dispositivo con las características eléctricas apropiadas puede ser manejado por la barra de control. En general, este tipo de dispositivos debe tener una alta impedancia de entrada y salida de tres estados.

El microprocesador 8080 tiene la capacidad de manejar una tarjeta adaptadora interface periférica programable para 8255. Contiene 24 conecciones externas que pueden ser programables como entradas o salidas en varias combinaciones.

Dicha interface es conectada al microprocesador, siendo

controlado el sistema a través del bus de datos, lectura entrada/salida, escritura entrada/salida, reengaste y, selección del dispositivo por el decodificador de direcciones.

El 8255 acepta datos de la barra de datos cuando el integrado es seleccionado y se encuentra en el modo de escritura. A su vez, este entrega señales al bus de datos cuando el integrado es seleccionado y está en modo de lectura.

El 8255 dispone de tres puertos (PA, PB, PC) y la señal de control (CNT) que es usada para programar los puertos externos como entrada o salida y seleccionar el modo de operación.

La configuración de los puertos es igual tanto para el modo manual de operación como para el automático. Se ha utilizado el 8255 #1 de la tarjeta interfase accesoria, configurándose sus puertos de la siguiente manera:

PA	entrada
PB	entrada
PC0-C3	entrada
PC4-C5	salida

en donde,

PB0--> sensa nivel,

PB1--> lee temperatura (una palabra de 8 bits),

PC0--> sensa estado del calentador,

PC1--> sensa estado del agitador.

PC2--> sensa estado de la bomba.

PC3--> sensa estado de la válvula.

PC4--> activa o desactiva el calentador.

PC5--> activa o desactiva el agitador.

PC6--> activa o desactiva la bomba.

PC7--> activa o desactiva la válvula.

p73

Además se habilita el 8255 #2 para enviar y recibir las señales de conversión del convertidor analógico digital para la temperatura.

2.1.1.- **PROGRAMACION DE LOS PUERTOS.**

Para configurar la tarjeta adaptadora interfase de acuerdo a lo ya señalado se utilizan las instrucciones

```
MVI A, 93
OUT 07
MVI A, 82
OUT 0F
```

Una vez programados los puertos, se 'limpia' una localidad de memoria que contendrá la palabra de control que será enviada a la salida para el control de los distintos componentes. De esta manera con operaciones lógicas se puede cambiar el estado de un bit, es decir que se podrá cambiar el estado de

cualquier dispositivo sin alterar a los demás.

3.- PROGRAMA DE CONTROL PRINCIPAL.

La UTR ejecutará las órdenes recibidas desde la estación maestra vía telefónica, ejecutando esta rutina de control durante el día sin intervención del operador (modo automático de operación).

En caso de fallar la recepción del programa proveniente del centro de mando, se dará paso al programa de recepción casetera- microprocesador 8080 para el control manual.
(ver figura 5.3).

3.1.- SEQUENCIA DEL PROGRAMA.

Una vez establecida la comunicación entre la estación maestra y la UTR, el operador activa el programa principal de control.

Si se ha establecido correctamente el enlace, el programa recibe la secuencia de ejecución del día alojándola en la memoria volátil (RAM) del microprocesador 8080.

Recibido el programa del día, continua sensando el nivel de la planta. Cuando este último alcanza su nivel mínimo necesario para la ejecución del proceso, 'salta' a las

localidades de la RAM que contienen los llamados de subrutinas recibidos que son los que representarán las actividades del día.

Al efecturase algún cambio en la planta, el programa de control residente debe comunicarlo a la estación maestra que dará permiso o no para que continue el proceso.

3.2.- CONFIGURACION INTERNA.

El programa se encuentra configurado de la siguiente manera:

Al inicio ejecuta un 'salto' hacia las rutinas de recepción de datos (vía modem) para recibir el proceso del dia, luego realiza un 'traslado' desde la EPROM hacia la memoria volátil con las rutinas que serán llamadas como una interrupción, siendo esta una característica propia de las subrutinas y comandos ubicados en el programa monitor del microprocesador 8080. Estas interrupciones son las utilizadas por los programas de comunicación, ya sea modem - 8080 o casetera - 8080. Una vez ejecutada esta sección del programa, la planta ha recibido la secuencia de ejecución del día y se ha preparado la comunicación 'half-duplex' con la estación maestra.

A continuación, los puertos de entrada/salida 8255 deben ser configurados de acuerdo a los requerimientos del proceso, tal como se anotó en la sección anterior.

Configurados estos, la pantalla del 8080 mostrará si se ha alcanzado el nivel adecuado para el proceso o no. Para esto, en la sección izquierda del despliegue se mostrará la etiqueta N recordando que lo que se señala es el nivel. El lado derecho del despliegue muestra si se ha alcanzado el nivel mínimo necesario o no (00 indica que aún no se cumple la norma, 01 indica que el proceso puede iniciarse). En caso de no cumplirse con este requerimiento básico, el programa esperará hasta que se alcance el nivel adecuado.

Terminado esto, se efectúa un salto a la dirección de memoria seleccionada en la RAM a partir de la cual se almacenó el programa del dia (localidad 8400), dando comienzo a la ejecución formal del proceso.

4.- SUBRUTINAS DE CONTROL.

Establecido el enlace telefónico y preparada la recepción del proceso, se envian las actividades del dia.

La manera escogida para ello, es con el uso del llamado de subrutinas que ejecuten las acciones típicas de control de unidad terminal remota (UTR).

Las acciones de control se ejercen sobre cuatro dispositivos: calentador, agitador, válvula y bomba, y son las de encendido y apagado. Además deberán existir tres actividades adicionales: esperar t minutos, fijar punto T de temperatura

(en grados centigrados) y mantener la temperatura T en un intervalo t de tiempo (en minutos).

El hecho de utilizar este método evita la práctica de particiones de memoria RAM para la ejecución del programa, además reduce la probabilidad de error en la recepción ya que la información enviada se encuentra en un paquete compacto.

Dentro de estas subrutinas de control se maneja la subrutina de transmisión de estado de la planta, la misma que informará los cambios ejecutados dentro del proceso.

A través de esta forma de 'conversación' en la que se maneja información concentrada se reducen costos de comunicación al p73 dirigir un número de terminales remotos a través de otro mayor. De la misma manera se puede obtener igual beneficio al compartir un circuito de comunicaciónentre varios terminales.

Es importante comprender que los requerimientos de los programas de aplicación tienen un impacto fundamental en el diseño de la unidad terminal remota. A pesar de ello, aunque hayan muchos posibles diseños y configuraciones para un subsistema de adquisición de datos, existen equipos básicos en común.

Por otro lado, la frecuencia de adquisición de datos de la estación maestra, es decir, la máxima edad permitida entre puntos de datos, depende en este caso, directamente de las subrutinas de control. De esta manera la frecuencia será de

acuerdo a los cambios detectados en la planta.

4.1.- SUBRUTINAS DE APAGADO Y ENCENDIDO.

Las subrutinas de apagado y encendido están formadas bajo una misma estructura:
un 'uno' lógico equivale a encender,
un 'cero' lógico significa apagar.

De acuerdo a esto, se cambia el estado del dispositivo solicitado y se llama a la subrutina encargada de enviar el estado actualizado de la planta.

Es necesario enfatizar que estas subrutinas no pueden afectar el estado de los demás dispositivos.

4.1.1.- SEQUENCIA DE LAS SUBRUTINAS.

Puesto que el estado de los demás elementos debe permanecer invariable durante la ejecución de una subrutina de control, se maneja una localidad de memoria que guarda el último estado de todos los dispositivos. Así, se puede variar tan solo un dispositivo mientras esta localidad 'recuerda' el estado delos demás,a la vez que 'recibe' el cambio en determinado elemento.

Esta 'palabra de control' es enviada hacia los periféricos a través del puerto de salida previamente programado en el programa principal de control.

Por último, se envía el estado de la planta con un llamado a la subrutina de transmisión de estado.

A este grupo pertenecen las siguientes subrutinas:

apagar calentador	CD 65 05
apagar agitador	CD 75 05
apagar válvula	CD 85 05
apagar bomba	CD 98 05
prender calentador	CD C0 05
prender agitador	CD D0 05
prender válvula	CD F0 05
prender bomba	CD E0 05

(ver figura 5.4)

4.2.- SUBRUTINA DE ESPERA.

Dentro del proceso, de acuerdo al tipo de mezcla que se esté efectuando o a ciertas condiciones predeterminadas es necesaria la presencia de intervalos de tiempo entre cada actividad.

Para ello se ha diseñado la subrutina de espera, en que las condiciones de la planta sufren los cambios naturales ya que no mantiene las condiciones del proceso sino que en un intervalo de tiempo predeterminado desde la estación maestra, se informa a cada minuto el estado de los

dispositivos, del nivel y de la temperatura.

Asi, para el llamado de esta subrutina es necesario 'cargar' el registro E con el número de minutos y luego solicitar la subrutina, de la siguiente manera:

1E #minutos CD 00 06

De esta manera, transcurrido un minuto se llama a la subrutina de transmisión de estado de la planta y se decrementa el registro E que hace las veces de contador.

(ver figura 5.5)

4.3.- SUBRUTINA DE JIFACION DE TEMPERATURA.

La temperatura representa un papel fundamental dentro del proceso. Ciertas mezclas deben alcanzar un nivel de temperatura especifico para la secuencia del proceso.

Es asi como se ha implementado una subrutina que eleve la temperatura al nivel señalado desde la estación maestra.

Para ello el operador debe almacenar en el registro B el valor deseado de temperatura en grados centigrados.

La subrutina se encarga de leer a través del puerto correspondiente el valor de temperatura manteniendo el calentador encendido (lo cual ebió ser efectuado previamente desde el centro de control). Asi, compara el valor actual de temperatura con el almacenado en el registro B hasta que sea

el requerido en donde 'sale' del lazo.

Vale mencencionar que dentro de este lazo se envía constantemente el estado de la planta a través de la subrutina correspondiente.

La ejecución del llamado debe ser hecha como sigue:

06 temperatura C5 CD 15 06

(ver figura 5.6)

4.4.- SUBRUTINA DE SOSTENIMIENTO DE CONDICIONES.

Como parte del mecanismo de proceso de la planta las condiciones del sistema deben mantenerse en periodos determinados de tiempo.

Debido a ello, se ha implementado una subrutina que satisface dicho requerimiento. Aquí, la palabra de control es mantenida en el puerto correspondiente variándose solo el estado del calentador teniendo en cuenta si la temperatura es mayor o menor a la requerida.

Esto ocurre dentro de un lazo que contendrá el número de minutos en el registro L que hará las veces de contador. Cada vez que se ejecuta el lazo se hace la transmisión del estado de la planta llamandose a la subrutina correspondiente.

La codificación para ejecutarla es la siguiente:

2E Nminutos CD 35 06

(ver figura 5.7)

4.5.- SUBRUTINA DE TRANSMISION DE ESTADO DE LA PLANTA.

La estación remota, como unidad programable, debe ser capaz de informar al centro de control si se están ejecutando o no las instrucciones dictadas al inicio del proceso.

Tres puertos han sido programados como entrada para recibir el estado de la planta. De esta manera, después de leer un puerto con la información en el acumulador se llama a la subrutina de transmisión vía modem. Así se envían en orden, el estado de la temperatura, luego nivel y por último el del calentador, agitador, bomba y válvula.

A continuación se pasa a un estado de espera en recepción, en donde el centro de mando informa que ha recibido correctamente los datos. Para esto se ha utilizado como código un asterisco (2A en su valor hexadecimal). Cualquier respuesta diferente será detectada como una señal de alarma que apagará el proceso y dará lugar al control manual de la planta.

Código: CD 70 06

(ver figura 5.8)

5.- PROGRAMA DE CONTROL MANUAL DE LA PLANTA.

Tal como se ha anotado, el control del proceso de la planta se lo realiza en dos posibles modos: manual o automático.

Cualquier falla en la transmisión o recepción estación maestra - estación remota, será detectada por el microprocesador 8080 que interrumpirá la secuencia y dará paso al control manual de la planta.

Este programa además, es utilizado para el proceso de mantenimiento de la UTR.

Dicho control se hará a través de un programa dispuesto en una cinta magnética. La recepción y transmisión casetera-8080 es controlada por una rutina almacenada en la ROM de microprocesador, de esta manera el operador 'llama' esta secuencia, almacenando así el programa de control manual.

Dicho programa deberá permitir al operador controlar los distintos dispositivos de la planta como son el calentador, el agitador, la bomba y la electroválvula, así como conocer el estado de cada uno de ellos. Además, podrá solicitar la temperatura del sistema, manteniendo la característica explicada anteriormente de alcanzar el nivel de liquido necesario para iniciar el proceso.

(ver figura 5.9)

5.1.- ESPECIFICACIONES DEL SISTEMA.

De acuerdo a los requerimientos del proceso, el programa ha sido diseñado de manera que al inicio se muestre el estado del nivel y permita al operador el control una vez que alcance el mínimo necesario.

Así, lleno el tanque, el operador podrá elegir el dispositivo a controlar o la visualizacióóón de la temperatura presionando la tecla respectiva destinada para cada uno de ellos.

Las rutinas de cada uno de los dispositivos realizan una acción similar, el operador lo elige, verifica el estado y tiene opción a cambiarlo o dejarlo igual.

Por ejemplo, al iniciar el proceso aparecerá en el display izquierdo la letra N indicando que el display derecho muestra el estado del nivel (00 indicará que aún no se ha llegado al mínimo necesario, 01 mostrará que el tanque ya está lleno listo para trabajar).

Cuando se llegue al nivel necesario se mostrará por unos segundos (aprox. 3 seg.) 01 en el display derecho. Luego se "limpiará" la pantalla indicando que el programa está listo para dar paso al control de cualquiera de los servicios mencionados.

De esta manera, si el operador decide controlar el calentador deberá presionar la tecla asignada para solicitarlo. En

seguida aparecerá en el display izquierdo la letra C (calentador) y en el display derecho el estado del mismo (00 apagado, 01 prendido). El estado se muestra en un intervalo de tiempo suficiente para la visualización, luego desaparece (pero la etiqueta C se conserva indicando el dispositivo) señalando que el operador puede prenderlo, apagarlo o salir sin variar el estado. Para los cuatro dispositivos que se manejan existe una tecla para el encendido y otra de apagado (general para todos).

Si no se desea cambiar el estado se presionará la misma tecla que solicitó dicho elemento.

Si se cambia el estado, aparecerá en el display derecho el nuevo estado del dispositivo.

Cada vez que se termine un servicio se limpiará la pantalla a la espera de una nueva solicitud.

En el caso de la temperatura, al solicitarla presionando la tecla respectiva aparecerá en el display izquierdo el valor en grados centigrados y en el derecho la etiqueta 't' (temperatura). Este valor mostrado cambia de acuerdo a las variaciones sensadas.

Con la tecla de apagado se abandona este servicio, a la espera de otra solicitud.

Por último, existe una tecla adicional de parada general que desabilita todos los dispositivos y que puede ser utilizada para el fin del proceso o en caso de emergencia.

(ver figura 5.10)

5.2.- PROGRAMA PRINCIPAL DE CONTROL MANUAL.

Al inicio del programa se configuran los puertos de entrada/salida de acuerdo a lo señalado arriba con las instrucciones

```
MVI A,93
OUT 07
```

Se "limpia" a continuación una localidad de memoria que contendrá la palabra de control que será enviada a la salida para el control de los distintos componentes. De esta manera con operaciones lógicas se puede cambiar el estado de un bit , es decir que podremos variar el estado de cualquier dispositivo sin alterar a los demás.

Después de esperar que la planta alcance el nivel necesario para el inicio del proceso, el lazo principal de control configurado a base de saltos condicionales, espera la solicitud de uno de los servicios para luego dar paso a la rutina correspondiente.

Las subrutinas utilizadas son las proporcionadas por el monitor, y otras adicionales de retardo y conversión binario-decimal.

5.2.1.- SUBRUTINAS DEL MONITOR UTILIZADAS.

CLEAR (0287)

Limpia todo el display. No necesita ninguna entrada.

CLRGT (0282)

Limpia los cuatro digitos derechos del display. No necesita ninguna entrada.

GETKY (023D)

Esta subrutina espera indefinidamente que se presione una tecla, y luego espera que ninguna tecla esté presionada por 26 mseg.

GETKY llama a SCAN para leer el teclado. La primera tecla presionada es detectada y retornada por GETKY en el registro A, siendo (B)=00. Todos los demás registros son conservados.

Carry es cero para las teclas de comandos y uno para las teclas hexadecimales.

Zero es uno al presionar MEM , cero para los demás.

Todas las interrupciones de salida del monitor, display, y todas las filas del teclado están habilitadas. No se necesita ninguna entrada.

SCAN (0257)

Desabilita las interrupciones del monitor. Prueba cada fila del teclado por turno, hasta que todas han sido probadas o hasta que se encuentre una tecla activada. Si ninguna tecla es presionada, todas las filas del teclado se desabilitarán. Si una tecla es presionada, su fila es habilitada y las otras

desabilitadas. Si sensa las teclas de comando primero, luego lo p73 hace con 8-F y finalmente con 0-7.

Si dos teclas de diferente fila son presionadas, SCAN detectará la tecla de la fila de más valor. Si dos teclas en la misma fila son presionadas, la subrutina retorna la de menor valor.

Si más de dos teclas son presionadas, un valor erróneo puede ser retornado.

Si ninguna tecla es presionada, SCAN retorna (A)=00, zero=1, carry=0.

Si una tecla es presionada, SCAN retorna (A)= valor de la tecla, carry=1. Zero=1 si la tecla es la del cero.

Todos los registros excepto A son preservados.

DBYTE (0295)

Muestra el contenido del registro A, después de cargar DE con la dirección del digito del display derecho, de esta manera se muestra el valor en dos posiciones del display derecho.

DWORD (02D1)

Muestra el contenido del registro par HL como cuatro digitos a la izquierda, cargando primero el par DE con la dirección 83FB del cuarto digito.

5.2.2.- **SUBRUTINA DE RETARDO.**

Función: Retardo= N x 100ms

Criterio: El parámetro N debe pasarse al programa de retardo vía el acumulador.

(ver figuras 5.11 y 5.12)

5.2.3.- SUBRUTINA DE CONVERSION BINARIO-DECIMAL.

Función: Convierte un byte en binario a decimal.

Criterio: El byte a convertir deberá estar en el acumulador previamente, la salida estará en los registros H y L.

4.3.-Ejecución del programa.-

Una vez almacenado el programa se ejecutará de la siguiente manera:

RESET

ADDR (dirección de inicio: 8400) RUN

CAPITULO 6

UNIDAD TERMINAL REMOTA.

1.- INTRODUCCION.

La unidad terminal remota o UTR. es la encargada de ejecutar todas las operaciones tanto manuales como automáticas que son dirigidas a controlar la planta. Es el brazo ejecutor de las acciones que sobre la planta se hagan con el fin de poder controlarla. Está constituido por dos bloques, el uno destinado a actuar sobre los circuitos de fuerza de la planta , y el otro destinado al control y a la comunicación lo cual es realizado por el KIT 8080.

2.- DESCRIPCION DE LA UNIDAD.

Este capitulo está destinado fundamentalmente al analisis de lo que constituye el hardware del proyecto es decir lo que constituye los transductores, sensores, fuente de voltaje,es decir la parte fisica del proyecto.

Los transductores son utilizados para sensar el correcto funcionamiento del sistema a ser controlado con la ayuda del microprocesador 8080 a través de sus puertos de entrada salida cuya información es pasada al interior del microprocesador para ser manejada de acuerdo a las condiciones establecidas por el programa ejecutado en ese instante.

La información de carácter analógico debe ser pasada por una etapa de conversión para ser transformada en una información digital que sea comprensible por el microprocesador que supervisa el funcionamiento del sistema controlado,lo mismo se aplica a las señales que son provocadas por el microprocesador de carácter digital para controlar al sistema.

2.1.- FUENTE DE ALIMENTACION.

La UTR debe ser alimentada desde una fuente alterna de 220 voltios ac la cual está compuesta de dos fases de 120 voltios cada una. La fuente de 220v es utilizada para energizar el calentador de la planta. Mientras que la fuente de 120 es usada para alimentar los demás dispositivos y para poder generar una fuente de voltaje DC cuya funcion y caracteristicas son descritas a continuación.

El sistema posee una alimentación de +12, -12 y +5 voltios que es proporcionado por tres reguladores de voltaje como indica la figura 6.1.

Las fuentes de +12 y -12 voltios es implementada usando uno de los secundarios que posee un tab central de tal forma que podemos implementar usando dos integrados que regulan el voltaje, en este caso el 7812 y 7912 respectivamente, para la fuente de +5 voltios se utilizó un regulador 7805 que posee

también la misma capacidad,los cuales necesitan un voltaje mínimo de entrada para que su salida se mantenga regulada, este debe ser por lo menos 2 voltios mayor al voltaje de salida regulado,así:

$$|Vent| > |Vsal| + 2 \text{ voltios}$$

De esto podemos concluir que necesitamos como mínimo un voltaje de entrada de 14 y 7 voltios en los reguladores correspondientes,pero éstos necesitan trabajar a un voltaje mayor,para lo cual se eligió 20 y 10 voltios respectivamente,los capacitores que filtran la señal rectificada son de 470 uF 50 voltios,puente rectificador de onda completa 200 Vac 1 Aac y un transformador de 120/2*15-15-10 Vac 60 VA.

2.2.- ETAPA DE TRANSDUCTORES .

La etapa de transductores posee las siguientes partes :

- Control-sensor de la electroválvula de la camisa
- Control-sensor de la bomba de recirculación
- Control-sensor del calentador
- Control-sensor del motor agitador
- Sensor de nivel
- Sensor de temperatura

2.2.1.- CONTROL-SENSOR DE LA ELECTROVALVULA,BOMBA DE RECIRCULACION CALENTADOR Y AGITADOR .

El control de la electroválvula, bomba, calentador y agitador se realiza a través de cuatro transistores que manejan cuatro relés que conectan estos mecanismos a sus respectivos voltajes de operación como indica la figura 6.2.1

Las señales de los puertos de entrada-salida del circuito integrado 8255 que pertenecen al control de estos elementos es amplificada con la ayuda de cuatro transistores que trabajan en corte y saturación,así,cuando esté un uno lógico a la salida de un puerto,esto hará que el relé se energice y conecte al elemento seleccionado,su desconección depende entonces de la colocación de un cero lógico a la salida de ese puerto.

El sensor de operación de estos elementos es realizado a través de los mismos relés que por medio de uno de sus contactos normalmente abierto se conecta un voltaje de +5 voltios que significa que el elemento está energizado figura 6.2.

Los datos de placas de los elementos enunciados son los siguientes:

- Electroválvula	220 Vac	50 VA
- Bomba de recirculación	110 Vac	40 W
- Calentador	220 Vac	3750 W
- Agitador	110 Vac	70 W

2.2.2.- SENSOR DE NIVEL

Como el tanque de operación se va ha llenar de forma manual el proceso de reconocer que el tanque está lleno,se lo realiza a través de un sensor de nivel el cual activa a un micro-interruptor con el cual se aplica +5 y cero voltios para que el microprocesador interprete que el tanque se encuentra lleno o con bajo nivel de agua.

2.2.3.- SENSOR DE TEMPERATURA

,Existen varios tipos de sensores que se pueden utilizar para medir la temperatura de un sistema determinado,los elementos más apropiados para este tipo de mediciones son los termistores y termocuplas.

EL TERMISTOR .-Es como su nombre lo implica una resistencia sensible a temperaturas,esto es su resistencia terminal está relacionada con la temperatura.Tiene un coeficiente negativo de temperatura,indicando que su resistencia disminuye con un aumento de temperatura de su cuerpo.No es un dispositivo de unión y se construye de germanio,silicio o una mezcla de oxido de cobalto,niquel,estroncio o manganeso,el elemento no es lineal sino aproximadamente exponencial,por lo que existe un rango de temperatura para el cual el valor de resistencia decrece rápidamente.

LA TERMOCUPLA .- Una termocupla consiste en la unión de dos o más hilos de diferentes metales en uno de sus extremos.Cuando la unión se calienta un voltaje es producido,la medida de este voltaje o la diferencia de potencial entre los terminales de dos hilos depende de las características de los metales y del valor de la temperatura de la unión.

El voltaje producido es usualmente de carácter lineal respecto al incremento o disminución de la temperatura en la unión.

Un detalle muy importante que debe tomarse en cuenta en el desarrollo de este mètodo para sensar temperatura es que la diferencia de potencial presente en los terminales de una termocupla es del orden de los milivoltios,por lo que es preciso pasar dicho voltaje por una etapa de amplificación previa a su análisis para un sistema de control.

Vemos que la termocupla presenta grandes ventajas con respecto al termistor en un circuito diseñado para sensar temperatura,entre las más importantes se tiene:

-.Suministra un voltaje en sus terminales que se puede usar para el sistema de control

-.La variación de su característica es lineal

—.No se necesitan fuentes adicionales para generar dicha variación de voltaje

2.3.- CALCULO DE CARGA

De acuerdo a la ecuación de potencia:

$$P=S*FP$$

donde P es la potencia activa en vatios,S la potencia aparente en voltios-amperios y FP el factor de potencia.

Asumiendo un factor de potencia de 0,8 tenemos:
La potencia total es de 3900 vatios,lo que representa un corriente de 17,7 amperios para un voltaje de 220 Vac;considerando un factor de 1,25 para el cálculo del disyuntor principal se tiene una corriente de 22,1 amperios,por lo que seleccionamos un disyuntor de 30 amperios.

Para cada elemento tenemos una protección de sobrecarga o cortocircuito:

ELEMENTO	TERMICO	FUSIBLE
Electroválvula		0,3 A 250 V
Bomba de recir.	0,5 A 250 V	
Calentador		20 A 250 V
Agitador	0,8 A 250 V	

La capacidad de los contactos para los relés seleccionados con una bobina de +12 voltios es:

- Electroválvula	1 A	250 V
- Bomba de recir.	1 A	250 V
- Calentador	20 A	250 V
- Agitador	1,5 A	250 V

5.4.- IMPLEMENTACION DE LOS CIRCUITOS ELECTRICOS Y ELECTRONICOS

Se construyó panel eléctrico de 40 centímetros de base,50 de altura y 18 de ancho,color amarillo. En este panel se instaló dos ventiladores para panel de 12 voltios 0.12 y 0.2 amperios respectivamente,uno para que entre aire del exterior y el otro para que extraiga el aire caliente del interior del panel. Dos disyuntores, General Electric de 30 amperios 500 voltios para protección general de los circuitos eléctricos. Un contactor siemens con tres contactos normalmente abiertos (N.A) y uno normalmente cerrado (N.C),los cuales tienen una capacidad de 5 kilovatios 500 voltios,dos contactos N.A son para energizar el calentador de 220 voltios 15 amperios y el otro N.A para señal de estado para que el microprocesador 8080 entienda si se encuentra encendido o apagado,este contactor tiene una bobina de 220 voltios,la cual es accionada por un relay de +5 voltios con un contacto N.A.

Tres relés squar de 5 amperios 250 voltios, con cuatro

pares de contactos,con bobina de 120 voltios,estos relay manejan dos motores de 110 voltios 40 y 70 vatios respectivamente y una electroválvula de 220 voltios 50 voltios-amperios,usando cada uno dos contactos N.A para su respectivo accionamiento y otro N.A para la señal de estado que es dirigida hacia el microprocesador 8080 el cual sensa su estado de conexión y desconexión.

Tres bases para relés de 15 pines las cuales tienen los terminales de conección de los contactos y bobina de los relay usados para los motores y electroválvula.

Cuatro bases de baquelita para fusibles de cartucho de vidrio 250 voltios 2 amperios para protección de los motores y electroválvula.

Dos conectores hembras de 25 pines para recepción y transmición de los bits hacia el microprocesador 8080,uno es para transmision y recepcion via MODEM entre un computador personal y el 8080,el otro conector usa cuatro pines para encender y apagar dos motores,un calentador y una electroválvula,cinco pines para sensar el estado de encendido y apagado de los motores,calentador,electroválvula y sensor de nivel de agua del tanque de calentamiento,ocho pines para el valor digital de la temperatura del tanque de calentamiento,tres para la multiplexación de las ocho entradas del convertidor digital-analógico 0808 que realiza

el microprocesador ajustando de cero a siete,dos pines para las señales de inicio y fin de conversión para el convertidor 0808,un pin para la señal de referencia del circuito impreso para el retorno de corriente de las diferentes señales de voltaje,este conector se acopla a un conector macho de 25 pines,el cual se comunica con los puertos de entrada y salida de los puertos de entrada y salida del circuito integrado 8255.

Ocho conectores hembras para las diferentes señales analógicas,para que sean convertidas por el 0808 en palabras de ocho bits.

Dos conectores NPT de 3/4 pulgadas, uno para entrada de los conductores de alimentación del circuito de 220 voltios y el otro para los conductores de encendido de los motores,calentador y electroválvula.

Se instaló cinco focos de señal 120 voltios 5 vatios para observar el estado de funcionamiento de los motores, calentador,electroválvula y sensor de nivel.

2.5.- DISEÑO DE LOS CIRCUITOS IMPRESOS

Se diseñó los circuitos impresos donde se montarán los circuitos integrados 8255 para los puertos de entrada y salida del 8080,uno de ellos es usado para el inicio y fin de

observa en la figura 1,una de +12 voltios 1 amperio para los ventiladores,dos de +12 y -12 voltios para polarización de los amplificadores operacionales y una de +5 voltios para el convertidor analógico digital y señales de acondicionamiento.

CAPITULO 7

VISUALIZACION.

1.- INTRODUCCION.

Luego que el programa a ejecutar durante el dia ha sido cargado en la unidad terminal remota. El paso siguiente es monitorear la planta, a fin de observar la evolución del programa previamente cargado, este seguimiento es realizado por medio de los paneles de visualización del operador, en los cuales vamos a representar la evolución del proceso en la planta.

La visualización de la planta es realizada en el monitor de un computador PC. Esta consiste de dos pantallas o paneles los cuales están dedicados a observar el proceso que se está desarrollando en la planta. Estas pantallas son activadas a voluntad por el operador por medio de dos teclas de función principales, adicionalmente existe otra tecla encaminada a cumplir el papel de tecla de emergencia con la cual el operador puede apagar la planta completamente desde el cuarto de control. Las teclas utilizadas para cumplir las funciones previamente descritas son F1 para activar la primera pantalla, F2 para tener acceso a la segunda pantalla, y ESC la cual se desempeña como tecla de emergencia. Todas las demás teclas no están habilitadas por lo cual si son presionadas no causan ningún efecto en el proceso de

visualización. Dentro del proceso de visualización se incluye una Alarma audible la cual entra en funcionamiento cuando la temperatura de la planta sobrepasa un valor máximo.
Adicionalmente la visualización va a tomar datos de tiempo y temperatura, a intervalos de tiempo iguales los cuales van a ser transmitidos cada cierto tiempo determinado, al centro de estadística donde se va ha procesar la información del proceso de la planta.

2.- DESCRIPCION DEL SISTEMA

En el puesto de control el operador tiene a su disposición en el monitor del computador PC, dos pantallas por medio de las cuales el operador puede conocer el estado de los contactos de la planta, la temperatura del agua de la planta, la fecha hora actuales y una curva de temperatura del agua versus tiempo. También el operador dispone de una tercera opción de emergencia mediante la tecla ESC estas funciones son descritas en los parrafos siguientes.

2.1.- PRIMERA PANTALLA DEL OPERADOR.

Cada vez que el operador cambia de pantalla pulsando la tecla F1 o cuando se inicia el programa de visualización, se tiene acceso a la primera pantalla del operador, la cual está formada por dos divisiones horizontales, en la división

inferior se muestra el panel mimico de la planta, y en la división superior la fecha la hora y temperatura de la planta actuales. El panel mimico de la planta está formado por gráficos de los diferentes dispositivos con que contamos en la planta para el control. El operador va a observar los estados de los contactos de los componentes de la planta que pueden estar encendidos o apagados, lo cual es representado por medio del color del dispositivo respectivo, los componentes con que contamos para el control y que son representados en esta pantalla están: el calentador, agitador, nivel de agua, transductor de temperatura, bomba de recirculación y electroválvula. El calentador es utilizado para calentar el agua dentro de la caldera, el agitador es usado para mezclar el agua de la caldera para conseguir una temperatura uniforme en todo el líquido, el transductor de temperatura es usado para convertir el valor de la temperatura en un valor digital, la bomba de recirculación es usada para hacer circular el líquido refrigerante para enfriar el agua en el interior de la caldera, finalmente la electroválvula es usada para permitir el paso del líquido refrigerante a la camisa para el enfriamiento de la caldera. Además del panel mimico previamente descrito también esta pantalla, muestra en la división superior izquierda la fecha actual de la máquina, en la parte central de la misma división tenemos que se muestra la hora, compuesta por hora minutos y segundos actuales, finalmente en la parte superior derecha se muestra la temperatura actual de la planta, la

cual es ingresada junto con el estado de los contactos desde el puerto serial via modem. Cabe señalar que los valores mostrados de hora y temperatura son continuamente actualizados en la pantalla de acuerdo a sus valores nuevos respectivos.

2.2.- SEGUNDA PANTALLA DEL OPERADOR.

Cuando el operador desea cambiar de pantalla pulsando la tecla F2, esta acción le permite poder visualizar la segunda pantalla, la cual está implementada en modo de alta resolución lo cual permite realizar gráficos con una mayor definición. Esta pantalla está compuesta por una barra vertical cuya funcion es indicar la presencia de nivel de liquido dentro de la planta, es decir que esta barra se grafica o no, dependiendo de si hay nivel o no hay nivel en la planta. Junto a la barra de nivel está una segunda barra vertical cuya magnitud va a variar en forma directa con el valor de temperatura que se registre en la planta. También está formando parte de esta pantalla una gráfica de Temperatura de la planta versus tiempo, en la cual el operador puede observar como a variado la temperatura de la planta en los últimos minutos. Incluído dentro de la visualización está implementada una Alarma Sonora la cual es activada cuando la temperatura de la planta a sobrepasado un valor máximo que es de 90°C. Valores de temperatura y tiempo son tomados a intervalos de tiempo iguales a fin de crear un

en su mayoria servicios tanto del BIOS como de servicios del DOS los cuales nos permiten un amplio control de las funciones primarias del computador. Los servicios del BIOS están almacenadas en la memoria de solo lectura del computador por lo cual generalmente se los conoce como los servicios de la ROM-BIOS.

Todos los servicios de la ROM-BIOS se invocan mediante interrupciones. Las instrucciones de interrupción apuntan a una posición en particular en la tabla de vectores de interrupción en las posiciones bajas de la memoria, que contienen un vector de interrupción: la dirección de la rutina de servicio almacenada en la ROM. Esto diseño hace posible que cualquier programa demande un servicio sin conocer la localización de memoria especifica de la rutina de servicio de la ROM-BIOS. Permite tambiéin que los servicios sean trasladados, expandidos o adaptados, sin afectar a los programas que utilizan dichos servicios. La forma estándar, preferida y más fiable de invocar un servicio de la ROM-BIOS, es utilizar su interrupción que su posición absoluta. En cambio los programas de los servicios del DOS son cargados en la memoria RAM del computador al momento en que este es inicializado. están guardados en archivos ocultos los cuales no son mostrados al usuario. Los servicios que puede prestar el DOS depende de la versión de sistema operativo con que se haya cargado el computador al momento de inicializar el mismo. Las interrupciones del DOS se llaman mediante códigos

de interrupción individuales con la instrucción INT.

3.1.- PROCEDIMIENTOS EN ENSAMBLADOR.

Hay dos tipos de programas en lenguaje Ensamblador: la subrutina o procedimiento en lenguaje Ensamblador, que es llamada por otros programas que pueden estar escritos en un lenguaje de alto nivel, y el programa en lenguaje ensamblador autosuficiente. Las subrutinas dependen en gran medida del programa que las llame para proporcionar su estructura y apoyo, mientras que los programas autosuficientes en lenguaje ensamblador deben proporcionarse su propia estructura y apoyo, y deben arreglárselas con los problemas operativos con los que se encuentran los programas que se soportan solos. Las subrutinas en ensamblador son relativamente fáciles de construir, mientras que los programas en ensamblador pueden ser bastante complicados.

Los procedimientos en lenguaje ensamblador deben de cumplir ciertas condiciones con el fin de que puedan ser llamados desde cualquier programa en Pascal. Estos requerimientos se basan fundamentalmente en que un procedimiento en lenguaje Ensamblador no puede tener segmento de datos. Otra característica importante de los procedimientos en lenguaje Ensamblador es que deben tener un nombre el cual deber ser único y estar declarado como público para que pueda ser incorporado dentro del programa ejecutable cuando se realice la compilación del programa principal. Un procedimiento en

Ensamblador es declarado público mediante la pseudoperación PUBLIC con lo cual el compilador sabe que este procedimiento ser llamado por otro programa. Cabe señalar que el encabezado y el proceso de inicialización de todos los procedimientos externos tienen la misma forma básica.

3.2.- ENLACE PASCAL ENSAMBLADOR.

Cuando un programa en Pascal llama a un procedimiento escrito en lenguaje Ensamblador el principal ingresa en la pila la dirección de retorno la cual luego de ejecutado el procedimiento es cargado nuevamente en el puntero de instrucción para ejecutar la siguiente instrucción del programa principal. El procedimiento en lenguaje ensamblador debe seguir una secuencia determinada con el propósito de retornar correctamente al programa principal. La pila es el medio atravez del cual se realizan las operaciones de paso de datos, de paso de direcciones de variables, asi como también direcciones de retorno, entre el programa principal y del procedimiento en ensamblador. En el procedimiento también se debe tomar en cuenta el tipo de parámetros que se pasan desde el principal, ya que de esto depende el número de bytes que Pascal va ha dejar en la pila antes de dar el control al procedimiento. Otro factor de importancia es el orden en el cual los parámetros o variables están declaradas en el procedimiento en Pascal ya que de esto depende en que

posición van a estar las variables en la pila. Finalmente se debe declarar el procedimiento como externo en el programa en Pascal por medio de la palabra clave EXTERNAL la cual le indica al compilador que debe buscar un procedimiento fuera del programa principal en un archivo ejecutable cuyo nombre también se lo coloca con la declaración.

Los pasos necesarios para que un procedimiento externo pueda ser llamado desde un programa escrito en Pascal se enumeran a continuación:

-Ensamblar el programa Fuente (Usando el MASM)
-Enlazar el programa objeto (Usando el LINK)
-Obtener en formato Binario (Usando el EXE2BIN)

El primer paso, es el paso normal para ensamblar un código fuente usando el Macro-Ensamblador, convierte el código fuente en instrucciones de lenguaje de máquina, pero no las dejan en una forma que están listas para ser ejecutadas. En vez de ello, los ensambladores ponen sus resultados en una forma que se conoce como código objeto y es depositado en archivo con la extensión .OBJ.

El segundo paso es realizar el enlace o montaje del programa previo de objeto por medio del programa LINK.EXE el cual nos da como resultado un archivo ejecutable con la extensión .EXE. En esta parte se completan direcciones de saltos las cuales no fueron puestas durante el primer paso.

Los programas que están almacenados en un disco en formato EXE no están completamente listos para ser ejecutados. Cuando

son cargados desde el disco en la memoria el DOS realiza unas pocas operaciones de último momento para preparar el programa para su ejecución. Esta preparación de carga puede ser hecha de antemano, convirtiendo el fichero a un formato de fichero .COM. Se usa el programa EXE2BIN para convertir un fichero .EXE en un formato de fichero .COM. No todos los programas pueden ser convertidos al formato .COM. Aquellos programas que están cualificados podemos libremente convertirlos, o dejarlos con el formato EXE.

4.- ESTRUCTURA DEL PROGRAMA DE VISUALIZACION.

El programa implementado para la visualización consta de tres bloques principales que son: procedimientos externos escritos en lenguaje Ensamblador, procedimientos en Pascal, y el programa principal también escrito lenguaje Pascal. Los procedimientos son llamados tanto del programa principal como de otros procedimientos en Pascal.

4.1.- PROCEDIMINETOS EXTERNOS EN LENGUAJE ENSAMBLADOR.

Los procedimientos externos en ensamblador cumplen varias funciones dentro de la visualización, estos procedimientos no están disponibles en Pascal Estandar ya que es requisito del proyecto no poder utilizar funciones que no están definidas dentro del Pascal Estandar. Por lo cual se hizo necesario implementar varias subrutinas o procedimientos externos cuyo

contenido y función se describe en los siguientes párrafos.

Las subrutinas externas en ensamblador implementadas para la visualización son las siguientes:

CURSOR.COM

LIMPIAR.COM

SUBG.COM

CAR1.COM

TIEMPOE.COM

FECHAE.COM

EXPLORAR.COM

MODO1.COM

PALETAFON.COM

PIXEL.COM

TIMBRE.COM

RECIBA.COM

TRANSMITA.COM

ESTATUS.COM

CURSOR:

La subrutina externa en ensamblador cursor tiene como función controlar el encendido o apagado del cursor en la pantalla en modo de texto cuando se grafica la primera pantalla del operador, con el argumento 0 desaparece el cursor, con el argumento 1 aparece el cursor en la pantalla.

LIMPIAR:

Esta subrutina externa divide horizontalmente la pantalla del operador colocando un color diferente para cada división, los

argumentos de la subrutina determinan los colores de cada división. Estos argumentos son pasados al procedimiento desde el programa en Pascal.

SUBG:

Esta subrutina externa nos permite imprimir una fila de un solo caracter con atributo, desde una posición fila columna. Todos estos valores son pasados como parámetros desde el procedimiento subgra que está en Pascal. Esta subrutina es utilizada para hacer los gráficos del panel mímico de la planta en la primera pantalla del operador.

CAR1:

Esta subrutina externa en ensamblador nos permite imprimir un solo caracter con atributo en la pantalla seleccionada en una posición de fila y columna, los valores que son utilizados por esta subrutina son pasados desde el programa en Pascal.

TIEMPOE:

Esta subrutina externa escrita en ensamblador nos permite extraer la hora actual del sistema y devolver estos valores al programa en Pascal por medio de los argumentos variables las cuales son alterados por la subrutina en ensamblador. Y luego procesados en el procedimiento que llama a esta subrutina.

FECHAE:

Esta subrutina externa en ensamblador nos permite extraer la fecha actual del sistema y devolverlo como el caso anterior en los valores de las variables que son pasadas como parámetros desde el procedimiento del programa en Pascal.

EXPLORAR:

Esta subrutina nos permite explorar el teclado y cambiar el valor del argumento del procedimiento cuando una de las teclas de función implementadas a sido pulsada. En caso de que el teclado no haya sido presionado o que una tecla diferente a las tres de función habilitadas haya sido presionada, la subrutina no altera el valor del argumento y regresa al programa que lo llamó.

MODO:

Esta subrutina externa cumple el papel de cambiar el modo de operación del monitor del operador, es decir que puede cambiar de baja resolución a alta resolución o viceversa, el modo de operación es controlado por medio del parámetro especificado desde el Pascal. Cabe señalar que el modo de alta resolución depende del tipo de monitor con que se disponga para la graficación.

PALETAFONDO:

Esta subrutina externa la utilizamos para seleccionar ya sea la paleta de colores o el color del fondo y el marco de la pantalla, lo cual es realizado solamente para el modo de alta resolución. La selección escogida depende de los valores de los parámetros especificados desde Pascal. En cada paleta de colores se puede seleccionar un grupo de tres colores que dependiendo de cuales se requiera son especificados desde el programa principal en Pascal.

PIXEL:

Esta subrutina externa es utilizada para graficar un pixel cuyo color, y posición son especificados desde el programa en Pascal. Este procedimiento es el corazón de toda la graficación en alta resolución ya que por medio de este procedimiento se realiza todos los gráficos y detalles en alta resolución en la segunda pantalla de operador.

TIMBRE:

Esta subrutina externa tiene como función sonar el parlante del computador cada vez que es llamada desde el programa en Pascal es usada como alarma para el operador de la planta. En este caso no es necesario pasar valores de variables desde el programa principal.

RECIBA:

Esta subrutina externa en ensamblador cada vez que es invocada recibe un caracter del puerto serial especificado desde Pascal. Es necesario antes que el puerto haya sido configurado a la velocidad adecuada de transmisión y recepción. Es usada para ingresar los datos de temperatura nivel y estado de contactos desde la planta.

TRANSMITA:

Esta subrutina externa es usada para transmitir un caracter a travez del puerto serial. El caracter y el puerto son pasados desde el programa en Pascal. Como en el caso anterior se debe configurar el puerto por el cual se va ha enviar el caracter antes de invocar a este procedimiento.

VERSTATUS:

Finalmente la subrutina externa verstatus es usada para

ingresar valores significativos del estatus del puerto serial como son dato preparado y error de alcance. Estos valores son devueltos a travez de la variable especificada cuando se llama al procedimiento.

4.2 PROCEDIMIENTOS EN LENGUAJE PASCAL.

La visualización también consta de procedimientos los cuales son en lenguaje Pascal y forman parte del programa Principal. Estos procedimientos son en su mayoría parte complementaria de los procedimientos en lenguaje ensamblador.

Los procedimientos utilizados para el programa de visualización son los siguientes:

SUBGRAF

CALDERA

AGITADOR

CALENTADOR

TERMOMETRO

LLAVE

BOMBA

NIVEL

IMPRIMAFECHA

IMPRIMAHORA

IMPRIMATEMP

PIXGRAF

CERO

BARRA

NIVELBAR

GLINEA

ACTUALICE

LINEATIEMPO

HAGARCHIVO

ACTARCHIVO

CONVERSIONDATOS

INGRESARDATOS

GRAFESTADOPLANTA

FALLA

SUBGRAF:

Este procedimiento es usado para imprimir el caracter con atributo n filas y m columnas desde la posición fila y columna especificada atravez de los parámetros de las variables.

CALDERA:

Por medio de este procedimiento podemos graficar los caracteres que forman la caldera cuyo color puede variar pero va a ser fijo luego de compilado el programa.

AGITADOR:

El procedimiento agitador nos sirve para graficar los caracteres que constituyen el agitador en la primera pantalla del operador. El valor del argumento determina el color mediante el cual el agitador es impreso en la pantalla.

CALENTADOR:

El procedimiento calentador imprime en la primera pantalla del operador el calentador el cual como en el caso anterior es un dispositivo que puede estar prendido o apagado lo cual se representa mediante el color en la pantalla, el color a imprimir es controlado por medio del argumento del procedimiento.

TERMOMETRO:

Este procedimiento lo utilizamos para graficar el termometro en la pantalla del operador su argumento va a ser constante lo cual nos indica que no va a variar de color en nuestro caso tiene el mismo color del agitador.

LLAVE:

La electroválvula es graficada mediante este procedimiento, también tiene un argumento el cual como en los anteriores va a determinar el color en la pantalla y por ende si el dispositivo está prendido o apagado.

BOMBA:

Por medio del procedimiento bomba se imprime en la primera pantalla el gráfico de lo que representa la bomba de recirculación, es un dispositivo que puede ser encendido o apagado lo cual hace que su color deba también variar, esto se hace mediante el argumento del procedimiento.

NIVEL:

El nivel representa el nivel de liquido de la planta lo cual también es graficado mediante el procedimiento nivel en este caso el color del nivel del liquido es graficado con un solo

color.

IMPRIMAFECHA:

Por medio de este procedimiento imprimimos los titulos de fecha, hora, y temperatura, junto con ello también imprimimos la fecha actual en la primera pantalla del operador. Este procedimiento extrae la fecha actual de la máquina por medio del procedimiento externo fecha e imprime este valor en la parte izquierda de la división superior.

IMPRIMAHORA:

Mediante este procedimiento imprimimos el valor de hora, minutos y segundos actuales los cuales son los valores extraidos del reloj interno del computador por medio del procedimiento externo tiempo, el cual devuelve en las variables el valor actual de la hora, minutos y segundos. Este procedimiento imprime estos valores, dependiendo si los valores que fueron imprimidos antes han variado en comparación con los actuales, en caso contrario no los imprime.

IMPRIMATEMP:

Este procedimiento nos permite imprimir el valor de temperatura ingresado de la planta atravez del modem. Si no se ingresan datos o el valor de temperatura es igual al anterior, el valor de la temperatura no es alterado. El valor de temperatura es impreso en la parte derecha de la división superior de la primera pantalla del operador.

PIXGRAF:

El procedimiento en Pascal pixgraf tiene como función,

imprimir en la pantalla en modo de alta resolución un grupo de pixeles cuyo atributo, número de filas, número de columnas, posición fila y columna son especificados por medio de los parámetros del procedimiento. Este procedimiento es usado para imprimir o borrar barras, lineas que se imprimen en la pantalla en modo de alta resolución.

CERO:

El procedimiento cero tiene como tarea, el escribir el número cero en la pantalla de alta resolución, con un color y posición de fila y columna especificados atravez de los valores de los parámetros del procedimiento. Se usa para graficar los ceros que forman parte del gráfico de alta resolución.

EJES:

Por medio de este procedimiento se dibujan todos los detalles que forman parte de la segunda pantalla del operador, estos gráficos son los ejes de las barras, divisiones, números y letras, todos ellos no cambian mientras la segunda pantalla del operador está activa.

BARRA:

Este procedimiento es utilizado para que imprima la barra de temperatura en la segunda pantalla del operador, también está encargado de borrar parte de la barra si es que la temperatura que es pasada como argumento, a disminuido. El valor de temperatura es pasado como argumento del procedimiento. Por ser en la segunda pantalla, para graficar

se utilizan procedimientos que grafican en alta resolución.

NIVELBAR:

Se basa en el mismo algoritmo del procedimiento anterior, salvo que en este caso en valor de la barra es uno o cero, si es uno se imprime la barra completa, si es cero no se imprime la barra. Este procedimiento puede ser modificado para que el valor del nivel, también pueda variar como en el caso de la temperatura. La barra es graficada mientras la segunda pantalla del operador está activa.

GLINEA:

Este procedimiento tiene como propósito graficar una linea desde una fila y columna especificada hasta una segunda fila y columna, con un color dado. Todos estos valores son conocidos por el procedimiento atravez de los valores de los argumentos que son pasados al procedimiento cuando es llamado. Es usado principalmente para imprimir o borrar la curva de temperatura versus tiempo en la segunda pantalla del operador.

ACTUALICE:

El procedimiento actualice es utilizado como su nombre sugiere para actualizar los valores de los puntos de la curva de temperatura versus tiempo asi como también actualiza las posiciones de los puntos a borrar de la curva durante el scroll que realiza cuando a alcanzado la linea vertical derecha. Estos valores son luego graficados en la segunda pantalla del operador. Este procedimiento se realiza continuamente mientras dure la visualización.

LINEATIEMPO:

Este procedimiento tiene como función graficar en la segunda pantalla del operador, la curva de Temperatura-Tiempo, así como borrar la curva anterior cuando el diagrama realiza el scroll o desplazamiento lateral, luego que la curva llega al margen derecho del gráfico. Esta curva es realizada utilizando otros procedimientos y es realizada en alta resolución.

HAGARCHIVO:

Este procedimiento tiene como función cuando es llamado, el crear el archivo de datos de temperatura y tiempo, el cual está formado por muestras de datos de temperatura y tiempo, tomadas cada cinco minutos. Este archivo creado es luego transmitido al centro de estadistica mediante el programa de comunicación.

ACTARCHIVO:

La función del procedimiento actarchivo es tomar los datos de temperatura y tiempo a intervalos de tiempo iguales y estos los ingresa en arreglos, los cuales cada cierto tiempo determinado son grabados en un archivo por medio del procedimiento anterior. Este procedimiento es llamado continuamente desde el programa principal.

CONVERSIONDATOS:

Mediante el procedimiento conversiondatos, los valores ingresados desde la planta son decodificados, con el fin de obtener los valores de los estados de los contactos de la

planta, también asigna los valores de temperatura y nivel. Este procedimiento es ejecutado cada vez que los datos llegan desde la planta.

INGRESARDATOS:

El procedimiento ingresardatos es llamado continuamente a fin de ingresar los datos de la planta si es que han sido transmitidos. En caso contrario si los datos no han sido enviados desde la planta, el procedimiento no realiza nada y regresa al programa que lo llamó.

GRAFESTADOPLANTA:

El procedimiento grafestadoplanta tiene como función la de graficar en la primera pantalla del operador los dispositivos cuyo color varia dependiendo de si están prendidos o apagados, los dispositivos que son graficados son calentador, agitador, bomba, llave, y nivel, los cuales son graficados por los procedimientos previamente descritos de cada componente.

FALLA

Finalmente el procedimiento falla es ejecutado cuando el operador ha pulsado la tecla de emergencia. Mediante este se apaga la planta completamente y se informa al operador mensajes que le informan de las acciones tomadas en la planta.

4.3.- PROGRAMA PRINCIPAL.

El programa principal en Pascal puede ser dividido en dos

bloques principales dentro de cada uno de los cuales se tiene un sub-bloque, este último dentro de un lazo, lo cual hace que se repita siempre que se mantenga una condición. El programa principal en Pascal primero asigna los valores iniciales a las variables del programa luego de lo cual entra en el bloque de procedimientos de la tecla F1. El diagrama de flujo se muestra en la figura 7.1.

4.3.1.- BLOQUE DE PROCEDIMIENTOS DE LA TECLA F1.

Dentro de este bloque primero ponemos a la pantalla en modo de texto esto se consigue usando la subrutina externa MODO y colocando el valor correspondiente como argumento, luego llegamos al procedimiento externo CURSOR con cuyo argumento podemos encender o apagar cursor en modo de texto,luego llama al procedimiento externo LIMPIAR con el cual se va a limpiar la pantalla de un color y divide la parte superior de la pantalla con un color diferente, estos colores pueden ser cambiados por el argumento al ser llamado el procedimiento. Luego el programa llama al procedimiento en Pascal CALDERA con el cual se grafica la caldera en la pantalla este procedimiento tiene un argumento que determina también su color de encendido o apagado, continua el bloque llamando al procedimiento en Pascal TERMOMETRO con el cual graficamos este dispositivo en la pantalla, a continuación el programa llama al procedimiento IMPRIMAFECHA el cual extrae e imprime

la fecha en la parte superior de la pantalla. Luego de ejecutado este último procedimiento se ingresa en un sub-bloque compuesto por procedimientos los cuales van a repetirse continuamente mientras la tecla de función siga siendo F1.

4.3.2.- SUB-BLOQUE DE PROCEDIMIENTOS DE LA TECLA F1

En este sub-bloque se llama al procedimiento en Pascal IMPRIMAHORA dentro de este procediminento se llama a la subrutina externa tiempo, la cual extrae el tiempo real de la máquina y por ende del dia y muestra hora, minutos, y segundos, y los imprime en en la mitad de la división superior de la pantalla. En seguida el sub-bloque llama al procedimiento en Pascal INGRESARDATOS por medio del cual los datos de la planta son ingresados desde el puerto serial por medio de tres subrutinas escritas en lenguaje ensamblador. Luego el sub-bloque llama al procedimiento en Pascal IMPRIMATEMP con el cual se imprime el valor de la temperatura ingresado antes, en la división superior de la pantalla. El procedimiento GRAFESTADOPLANTA es luego ejecutado, con este se grafican los componentes variables de la planta dentro de este procedimiento se llaman a otros procedimientos los cuales son: el Procedimiento llamado AGITADOR con el cual se grafica el agitador en la pantalla y se pasa un atributo, el cual va ha depender si está prendido o apagado el agitador lo que har que varie su color, indicando con esto al operador de su estado, al igual que el procedimiento anterior los

procedimientos CALENTADOR, TERMOMETRO, LLAVE, BOMBA, y NIVEL siguen los mismos pasos. Luego el sub-bloque llama al Procedimiento EXPLORA el cual tiene como función explorar el teclado y analizar si una de las teclas de función a sido presionada con lo cual el valor de la tecla varia lo cual hace que se ejecute el bloque correspondiente a la tecla pulsada. Luego el sub-bloque llama al procedimiento en Pascal ACTUALICE en el cual se actualizan los valores de la curva de temperatura versus tiempo que es graficada en la segunda pantalla del operador. El procedimiento que se ejecuta luego es ACTARCHIVO en el cual se actualizan los valores de temperatura en intervalo de tiempo dado y luego es creado el archivo a ser transmitido a la otra computadora para realizar las estadísticas del proceso.

Este sub-bloque de procedimientos se repiten mientras la tecla de función sea igual a F1, si se pulsa F2 y luego F1 el bloque se ejecuta desde la parte inicial, es decir desde el procedimiento MODO y se repite la secuencia nuevamente. Si la tecla de función F2 es presionada el programa ejecuta el bloque de procedimientos correspondientes a la tecla F2. El diagrama de flujo se muestra en la figura 7.2

4.3.3.- BLOQUE DE PROCEDIMIENTOS DE LA TECLA F2

Al igual que en el bloque anterior tenemos procedimientos que se repiten solamente una vez cuando la tecla F2 a sido pulsada y otro grupo los cuales se repiten mientras la tecla

de función sea igual a F2.

El procedimiento externo PALETAFONDO es el primero en ser llamado en este bloque, con este seleccionamos los colores de fondo y marco para nuestro gráfico en alta resolución. Luego con este mismo procedimiento seleccionamos la paleta de colores que vamos a usar en este modo de pantalla. Luego cambiamos la pantalla a alta resolución por medio del procedimiento externo MODO . Luego se usa el procedimineto en Pascal EJES con el cual graficamos todos los ejes del gráfico de alta resolución asi como también las letras y números. Luego de realizado el procedimiento anterior entramos en el sub-bloque de procedimientos como en el caso de la tecla anterior.

4.3.4.- SUB-BLOQUE DE PROCEDIMIENTOS DE LA TECLA F2

Este sub-bloque de procedimientos está dentro de un lazo los cuales se repetiran mientras la tecla de función sea igual a F2. El procedimineto externo TIEMPO es primero ejecutado para extraer la hora de la máquina y asignar las variables para luego actualizar curva de temperatura y los datos a ingresar al archivo. Luego el procedimiento INGRESARDATOS es nuevamente ejecutado en esta parte para ingresar los datos de la planta atravez del puerto serial. BARRA es luego ejecutado para graficar la barra de temperatura actual en la pantalla. Como en el procedimiento anterior se llama a NIVELBAR para graficar la barra de nivel de la planta en la pantalla. Una

vez terminado el procedimiento anterior el sub-bloque llama al procedimento EXPLORA el cual es el mismo procedimiento explicado anteriormente tiene como función explorar el teclado. Luego se ejecuta el procedimiento en Pascal ACTUALICE con el cual actualizamos los valores nuevos de temperatura y tiempo y también los valores de las lineas que hay que borrar cuando hace el desplazamiento la curva de temperatura-tiempo. ACTARCHIVO es luego ejecutada para actualizar los valores de temperatura y tiempo que es luego transmitido al centro de estadistica. Finalmente se ejecuta el procedimineto en Pascal LINEATIEMPO la cual nos imprime la curva de temperatura versus tiempo. El lazo se repite desde el procedimiento TIEMPO si es que no es pulsada ninguna tecla diferente de F2. Si es que la tecla ESCAPE es pulsada, el valor de tecla se altera al valor correspondiente y hace que se ejecuta el procedimiento llamado FALLA en el cual se manda a apagar la planta y se informa al operador del proceso realizado. El diagrama de flujo correspondiente se muestra en la figura 7.2

5.- CONCLUSIONES.

El proceso de visualización es una de las partes principales del proyecto devido a que aqui llega y sale toda la información del proceso. El programa a sido diseñado de tal forma que pueda ser instalado en cualquier tipo de computadora lo que lo hace particularmente útil. Se usaron procedimientos en lenguaje Ensamblador por lo cual para poder

entenderlo hace necesario tener conocimientos de lenguaje Ensamblador.

La visualización es también utilizada en la industria ya que muchos procesos automáticos requieren del monitoreo de estos, por parte de una persona calificada la cuál opera desde un puesto de control por medio de una computadora. Para cada aplicación hace necesario crear su propio programa de visualización ya que va a depender de cuantos componentes conste la planta y de con que tipo de resolución desee implementar. También juega un papel importante dentro de la calidad de los gráficos, el tipo de monitor que se tenga para la visualización ya que de este va a depender la resolución y el numero de colores disponibles para los gráficos.

Las interrupciones del BIOS y las del DOS son las más utilizadas para la graficación y en general para trabajar programas Pascal-Ensamblador como el nuestro.

Podemos encontrar en la industria actual muchas aplicaciones para la visualización por lo que es muy utilizada.

CAPITULO 8

ADQUISICION DE DATOS

1.- INTRODUCCION.

El proceso de control de planta tiene un control de investigación y análisis que permite la comparación de datos por dia y el resultado de cada comparación para optimizar o realizar cambios a la planta. Esto se lo realiza en un centro de control de proceso y adquisición de datos, en la cual llegan datos pedidos o deseado y luego se realizan cálculos estadisticos que permiten la relación general entre planta y proceso.

La adquisición de datos es la que permite la llegada de los mismos al centro de computo, sin ésta parte, seria ineficiente el trabajo; los datos llegan desde la planta de control mediante una comunicación rápida de tipo serial, con un protocolo sencillo que permite la agilidad y rapidez de operación.

El programa de adquisición de datos está escrito en lenguaje C, dividido en dos partes, la primera de llamado de sentencias executables y la segunda la aparición de comandos de control de transmisión- recepción.

Dentro del programa se llama al programa de cálculos y gráficos; es decir es un complejo sistema que permite a

cualquier ingeniero de control la operación de los mismos.

2. ESTRUCTURACION DEL PROGRAMA.

El programa contiene funciones de lenguaje C internas, y el programa principal; las funciones internas son para el llamdo de las opciones que se desea realizar. La presentación del programa es el siguiente:

1. Insertar datos en Achivo
2. Mostrar datos llegados
3. Realizar cálculos de proceso
4. Reporte final
5. salida al sistema principal

* Insertar datos:

Permite al operador una vez recibidos los datos incorporarlos a un archivo que simula una base de datos, este archivo recibirá diariamente hasta 600 datos totales, el nombre general es de TABLA.DBF; los datos que se archivan llevan el siguiente formato:

08:00 50

08:05 55

y así sucecivamente.

Los datos que llegan son en intervalos de hora, cada muestra llegada son datos que van variando cada cinco (5) minutos, y se toman desde el inicio del proceso; es decir si se inició a

las ocho (8am) los primeros datos se reciben a las 9am en un total de 12 y asi sucecivamente.

* **Mostrar datos llegados:**

Esta opción permite enseñar al operador si los datos adquiridos en cada hora no presentan errores o basuras y se lo visualiza en el monitor de la siguiente manera:

HORA	TEMPERATURA
08:00	56
08:05	65

hasta completar los doce (12) datos llegados

* **Cálculos de proceso:**

Este procedimiento es el que permite realizar el análisis estadistico de la planta y sus caracteristicas generales.

* **Reporte final:**

Me permite al final del dia tener un reporte gráfico de apreciasión de como funcionó la planta durante las horas de operación, el gráfico es dé tipo polar y además se podrá observar en un archivo los datos promedios por hora.

* **Salida al sistema principal:**

Me envia al sistema de control general escrito en lenguaje C y permite llamar comandos executables.

El sistema principal permite llamar comandos del sistema estos comandos son los siguientes:

* **Puerto1:**

Permite configurar el puerto serial para la recepción; la configuración es en los bit de control de recepción y la velocidad de la misma.

* **Recibe1:**

Este comando permite a la máquina entrar en un lazo de espera hasta recibir todos los datos necesarios que fueron transmitidos o serán transmitidos; mientras ocurre la espera, no se puede realizar ningún otro proceso, al final de la recepción sale un sello de verificación indicando una recepción correcta.

* **Centroes:**

Llama al programa expuesto anteriormente que permite insertar datos, visualizar y realizar los cálculos estadisticos de control.

* **Hora:**

Permite al operador visualizar la hora y llevar un control de los datos llegados y la hora de llegada de los mismos asi como también, mejor control de todo el proceso.

*** Salir:**

Con este comando salimos al sistema operativo para la ejecucuion de otro tipo de programas, dentro de la misma planta.

El programa fue escrito el TURBO C de BORLAND; su utilización es fácil y es posible incrementar más caracteristicas necesarias en el control de un proceso; las funciones que contiene son:

*** menu:**

Da la presentación principal y las opciones que presenta el programa.

*** Visualizar:**

Permite mediante la apertura de un archivo, la visualización de los datos llegados desde la planta.

*** Insertar datos:**

Adiciona datos en cada recepción al archivo de la base de

datos.

* **Reporte:**

Crea un reporte final del proceso, visualiza un archivo de datos promedios por hora y luego grafica temperatura vs hora en valores promedios; en donde; la hora son los ángulos de los arco (hora*15°) y el radio es la temperatura promedio del sistema (temperatura*2).

Como se puede apreciar es un programa sencillo que permite la rápida adquisición de datos.

3.- CARACTERISTICAS ESPECIALES.

El programa simula una base de datos, pero en realidad el programa final trabaja con una base de datos real bajo ORACLE; el problema encontrado fue la relación entre DOS y esta base de datos, para su perfecto uso se realiza el mismo proceso utilizando Microsoft C y Microsoft Assembler, para accesar a la base de datos utilizamos el Pro-C; una vez conseguido el ingreso a esta base de Datos, el proceso es similar, pero para este caso los datos ya no van a un archivo sino a una tabla de proceso que guarda los datos y es posible el manejo de esa tabla desde Lenguaje C. Para la adquisición de datos simplemente recibimos e insertamos; si deseamos visualizar leemos del archivo de llegada los datos incorporados y la opción adicional sería poder observar los

datos que contiene la tabla en el momento.

Se realizó un programa adicional de ejecución de comandos, debido a que el programa de transmisión recepción fue escrito en otro tipo de lenguaje no compatible con el TURBO C; para evitar problemas y errores se creó el adicional que se explicó anteriormente.

4.- **CONCLUSIONES.**

Este capítulo es muy sencillo y la estructuración del programa puede observarse en el listado ubicado en el anexo; para nuestro caso sólo se recibe e inserta datos; verificando que los datos sean correctos; en el caso de errores se procede a comunicar a la planta sobre el error para la correción del mismo.

Utilizar la base de datos fue al inicio muy difícil, por razones de tiempo se la simuló con archivos; pero las investigaciones que se realizarón con ayuda de Ingenieros expertos en Base de Datos y Programacion, ha permitido encontrar la solución y poder mejorar el programa.
Esta mejora queda como un adicional y no consta dentro de este informe general.

El manejo es sencillo y no necesita mayor información; una vez accesados los datos, se procede a los procesos de cálculos y graficación estadística.

CAPITULO 9

MUESTRA Y OBTENCION DE DATOS

POR MEDIO DE

PROCESOS ESTADISTICOS Y NUMERICOS

1.-INTRODUCCION.

Los datos del sistema estan almacenados en la base de datos del sistema (en nuestro caso se utilizan archivos que se renuevan dia a dia) y que son actualizados cada cierto tiempo.

Nos interesa tomar de los datos almacenados la mayor cantidad de información posible, a demás de la que por si nos muestran los datos, esto es una TEMPERATURA en un TIEMPO determinado de un dia en especial; pero de esta información agregado ciertos datos externos como consumo de energia, material, refrigerante, o producción podemos sacar concluciones a cerca del proceso, ya sea viendolo desde el punto de vista de producción, contról o mantenimiento.

Podemos incluso con cierto grado de confiabilidad, haciendo supociciones predecir el comportamiento del sistema en cualquier area, dependiendo de ciertos factores del area que se analice.

2.- CLASIFICACION DEL AREA DE ESTUDIO.

De los datos obtenidos de la planta, esto es temperatura

y hora, podemos observar el proceso que es continuo durante todo el dia y se puede concluir que el sistema estará sujeto a cambios de temperatura cada cierto tiempo.

Estos cambios de temperatura afectaran de algún modo, no solo al producto en si que es lo que nos interesa, sino también al sistema en si. Los elementos que conforman el sistema, el area de producción,los elementos que se utilizan para producir una determinada cantidad de producto son también afectados.

Como la temperatura es el factor más predominante en el sistema, es necesario hacer un analisis de la misma para un tiempo determinado o para todo un dia de producción.

El problema esta encaminado a resolver de alguna manera la forma en el cuál el ingeniero pueda darse cuenta de lo que sucede el sistema en un tiempo determinado, por lo que la primera area de estudio sera la variación de temperatura en un tiempo determinado y las condiciones del sistema para un dia de trabajo bajo ciertos parametros ya prefijados de operación.

Otro problema es el saber bajo que parametros podemos trabajar sin que se altere la producción o aun mejor saber los parametros que podemos alterar para aumentar la producción.

Considerando solo sistemas como la cantidad de producto

vs una determinada temperatura de proceso o producción vs tiempo de elaboración,sistemas en que una variable depende solo de otra, sistemas lineales en otras palabras, podemos mediante tablas de resultados u observaciones anteriores, crear funciones que se aproximen de alguna manera al comportamiento del sistema bajo esas condiciones.

El problema se dirige ahora a otra area que es la de aproximación, por lo que en base de datos se tratara de estabecer una relación entre estos y una función para la evaluación de datos ficticios para observar el comportamiento de la variable analizada, usando metodos de aproximación numérica.

Debido a esto el otra area de estudio seria entonces conocer de alguna forma como responde el sistema en si a las condiciones en que se esta elaborando un producto; dicho proceso es constante para una fecha determinada, por lo que el sistema estará constantemente sometido a una condición que puede ser beneficiosa o perjudicial para el sistema y por lo tanto sera parte de nuestro problema en el analisis del sistema de contról.

Podemos resumir lo anterior en los siguientes puntos:

- **ANALISIS ESTADISTICO DEL PROCESO.**

Se podran tomar muestras en un dia en diferentes horas para

el cálculo de la media,mediana y varianza de la temperatura.Para un dia y un intervalo de tiempo determinado se extrae una muestra y se saca un histograma y una ojiva.

- **APROXIMACION POR METODOS NUMERICOS.**

Con datos tomados previamente se podra aproximar una función la que se puede evaluar para estimar valores no tomados o no observados para hacer supociciones a cerca del comportamiento del sistema dependiendo de como varie una determinada variable.

- **EVALUACION DIARIA DEL PROCESO ANTERIOR.**

Con los datos almacenados se puede hacer una ejecución del programa para evaluar los valores medios de cada hora y almacenarlos en una tabla para su observación.

3.- METODOS UTILIZADOS PARA EL DESARROLLO DEL PROBLEMA.

3.1.- ANALISIS ESTADISTICO.

El agrupamiento de datos obtenidos de forma experimental o de un proceso determinado se deben de agrupar de forma adecuada para su analisis y presentación gráfica. ¡ver figura 9.1¿.

El comportamiento del sistema en si ya es predecible de la forma como se presentan los datos conforme transcurre el

proceso diario, pero para una observación más detallada en un periodo determinado son útiles algunos indicadores estadísticos.

La DISTRIBUCION DE FRECUENCIA es una tabla que divide un conjunto de datos en un número de clases o categorias apropiadas, mostrando a demás el número de elementos de cada clase; esta tabla a pesar de hacer perder identidad a los datos de la muestra, debido a que se presentan solo como pertenecientes a un grupo determinado, compensa dicha perdida porque resalta la caracterisiticas principales de los datos asi agrupados.

Las distribuciones de frecuencia cuyos datos se hayan agrupados por tamaño se denominan DISTRIBUCIONES NUMERICAS y las agrupadas por alguna cualidad o atributo se conocen como DISTRIBUCIONES CATEGORICAS, nuestros datos agrupados son entonces una DISTRIBUCION NUMERICA porque son agrupados de acuerdo a su peso (en nuestro caso es el valor de temperatura).

Para agrupar los datos se deben determinar las clases a utilizar y sus limites; el rango de las clases es la diferencia entre la observación mayor menos la menor, en nuestro caso el limite inferior esta entre los 20 y 25 grados y la mayor entre los 130 y 150 grados por lo que el rango abarcaria los 130 grados. Pero considerando que el refrigerante puede no solo enfriar sino congelar las

sustancias internas del recipiente el rango se determino desde los 0 grados hasta mayores 135 grados.

Las fronteras de clase que se escogen son valores de cierto modo imposibles porque ya se conoce que los datos no llegaran a más de 1 cifra decimal y por lo tanto se pueden traslapar. a pesar de esto se han colocado las siguientes fronteras para 10 clases ya definidas para el tipo de mediciones que tenemos (tabla1).

FRONTERAS DE CLASE				
>= 0.00	-	< 15.00	GRADOS	
>= 15.00	-	< 30.00	"	
>= 30.00	-	< 45.00	"	
>= 45.00	-	< 60.00	"	
>= 60.00	-	< 75.00	"	
>= 75.00	-	< 90.00	"	
>= 90.00	-	<105.00	"	
>=105.00	-	<120.00	"	
>=120.00	-	<135.00	"	
>=135.00			"	

TABLA 1.- CLASIFICACION DE LAS FRONTERAS DE CLASE UTILIZADAS

Los datos quedan representados por las MARCAS DE CLASE que son el promedio de los valores de las fronteras de clase, el valor entre cada marca de clase; distribuciones con clases de la misma longitud como nuestro caso se denomina INTERVALO DE CLASE y tiene un valor de 15. Nuestro problema tendría las marcas de clase presentadas en la tabla 2.

CLASE	MARCAS DE CLASE
1	7.5
2	22.5
3	37.5
4	52.5
5	67.5
6	82.5
7	97.5
8	112.5
9	127.5
10	---

TABLA 2.-CLASIFICACION DE LAS MARCAS DE CLASE UTILIZADAS

Si se desea comparar distribuciones de frecuencia es más ventajoso mostrar las DISTRIBUCIONES PORCENTUALES, esto se logra dividiendo la frecuencia de cada clase para el número

total de observaciones y multiplicarlo por 100; nuestro programa nos da la distribución de frecuencia y la distribución porcentual.

Las propiedades de estas distribuciones se pueden observar por medio de gráficas; la forma más comun es el HISTOGRAMA, que son rectangulos cuya altura representa la frecuencia de clase y sus bases se extienden entre dos fronteras de clase sucesivas.

De esto podemos observar cuál ha sido la mayor temperatura (representada por su marca de clase) a la que ha sido sometido el sistema durante el tiempo de observación, valores aberrantes que no coinciden con el patron de los datos o que exhiba dos o más maximos(MODAS), con lo que concluiriamos que existen errores en la medición o se altero el proceso en algún momento.

Otra forma de observar esto es el POLIGONO DE FRECUENCIAS que representa las marcas de clases con sus frecuencias de clases correspondientes unidas por lineas rectas agregando clases con frecuencia cero en los limites de la distribución.

Si realizamos distribuciones de frecuencia con la caracteristica de ser "menor que" un limite ya establecido (FRONTERAS DE CLASE) se obtienen distribuciones acumuladas

que mostrarian el número totál de observaciones menor que los valores dados. Este tipo de distribuciones se presenta graficamente en forma de OJIVAS, que es un gráfico parecido al poligono de frecuencia excepto que se grafican las frecuencias acumuladas sobre las fronteras de clase en vez de graficar las frecuencias sobre las marcas de clase.

Un ejemplo de este tipo de gráficos podemos observarlo en la figura 1a,b,c.,con la ayuda de la tabla 3.

HORA	TEMPERATURA (GRADOS C)	HORA	TEMPERATURA (GRADOS C)
08:00	25.6	10:00	82.3
08:05	32.5	10:05	82.1
08:10	53.8	10:10	80.3
08:15	79.5	10:15	78.2
08:20	80.1	10:20	81.2
08:25	82.3	10:25	46.1
08:30	78.5	10:30	25.1
08:35	81.5	10:35	22.0
08:40	61.2	10:40	25.3
08:45	22.3	10:45	33.7
08:50	21.5	10:50	55.4
08:55	25.6	10:55	80.2
09:00	60.2	11:00	82.4
09:05	78.5	11:05	80.3
09:10	80.6	11:10	78.6

09:15	82.1		11:15	81.6
09:20	79.5		11:20	25.6
09:25	80.2		11:25	32.5
09:30	64.2		11:30	53.8
09:35	30.2		11:35	79.5
09:40	25.2		11:40	80.1
09:45	24.6		11:45	82.3
09:50	47.5		11:50	78.5
09:55	71.6		11:55	81.5

TABLA 3.- MUESTRAS DE UN PROCESO FICTICIO

FIGURA 1a.-HISTOGRAMA PARA UNA MUESTRA DE TODO EL PROCESO

FIGURA 1b.-POLIGONO DE FRECUENCIAS PARA UNA MUESTRA DE TODO EL PROCESO.

FIGURA 1c.-OJIVA DE UNA MUESTRA DE TODO EL PROCESO

Los datos deben tener cierta medida que nos indique su punto medio o localización central(valor medio), esta medida es la MEDIA ARITMETICA que se define como la suma de todos los valores de las observaciones dividido para un número de n observaciones:

$$\bar{X} = \frac{\sum_{i=1}^{n} Xi}{n}$$

La MEDIANA es otra medida descriptiva del centro de un conjunto de datos y evita la influencia de datos extremos,muy grandes o muy pequenos, y nos dara el valor más cercano a la mitad una vez que ordenemos dichos datos. La mediana se obtiene de n datos ordenados de la siguiente forma :

Para n impar el valor de la mediana es el valor que aparece en la posición (n+1)/2; si n es par sera el promedio de los valores que aparecen en la posición n/2 y (n+2)/2.

En todo caso estos valores nos serviran para conocer cuál es el valor de temperatura que más ha predominado en el sistema durante el periodo de observación.

La VARIANZA de n observaciones mide escencialmente el promedio de los cuadrados de las desviaciones con respecto a su media. La varianza y la DESVIACION ESTANDAR son medidas de variación absoluta y miden la cantidad real de variación presente en un conjunto de datos y dependen de la escala de medición.

La variación estandar es la raiz cuadrada de la varianza.

$$s^2 = \frac{\sum_{i=1}^{n} (X_i - \bar{X})^2}{n-1}$$

Si se van a comparar variaciones de distintas muestras es preferible hacerlo con una medida relativa como es el COEFICIENTE DE VARIACION que da la desviación estandar como un porcentaje de la media; así a mayor sea el coeficiente de la muestra mayor sera la desviación de los datos con respecto a la media.

$$V = 100\%(s / \bar{X})$$

3.2.- APROXIMACION POLINOMICA.

Si para ciertas condiciones de operación del sistema se tiene que una variable varia con respecto a otra como por ejemplo la cantidad de producto y el número de procesos por día o la producción y la cantidad de refrigerante usado,se tienen una serie de datos que nos serian muy utiles si pudiermos aproximarla a una caracteristica o función que no solo refleje estos datos sino que nos permita predecir cuál sera el comportamiento de una de estas variables con respecto a la otra interpolando o extrapolando valores. ¡2¿

En base a estos datos crearemos una función (representada por un polinomio) que puede ser lineal o de

tipo curvilineo dependiendo del grado del polinomio que nosotros propongamos.

El metodo usado es la APROXIMACION POR MINIMOS CUADRADOS en el cuál se plantea el problema de obtener la mejor recta aproximante cuando el error involucrado es la suma de los cuadrados de las diferencias entre los valores de la recta aproximante y los valores dados.

Aunque existen otros metodos como la aplicación de minimos cuadrados pero asumiendo que los datos estan relacionados exponencialmente o utilizando polinomios formado por funciones ortogonales entre si que pertenecen a un mismo conjunto que es linealmente independiente para un rango ¡a,b¿ donde b>a.

Este tipo de metodos se descarto primero porque se supone que se aproximaran funciones de tipo lineal y por lo tanto el polinomio de aproximación sera de grado 1; segundo porque si los datos tuvieran una tendencia curvilinea el polinomio de aproximación puede ser de grado 2 o 3 y aproximar los datos con cierto margen de error y tercero porque los polinomios creados por conjuntos linelmente independientes involucran generalmente funciones trigonometricas que tienden a oscilar y por lo tanto hacen variar la cota de error en la aproximación a pesar de ser faciles de calcular,integrar o derivar dependiendo del caso.

Una razón adicional para considerar el enfoque de los

minimos cuadrados tiene que ver con el estudio de la distribución estadística del error y que para M datos se puede tener un polinomio de grado n (n<M) usando el metodo de minimos cuadrados y su solución queda expresada como un sistema de ecuaciones donde las variables son las constantes pertenecientes al polinomio de aproximación.(1)

Este sistema de ecuaciones que nos dá el metodo de los minimos cuadrados se reduce en el programa por el metodo de solución de ecuaciones lineales por ELIMINACION GAUSSIANA Y SUSTITUCION HACIA ATRAS,aunque existen otros metodos para la solución de este tipos de ecuaciones.(1)

****

NOTA 1 . La demostración para la obtención de una ecuación de tipo linal o un polinomio de grado n (si n=1 el polinomio se vuelve de tipo lineal) y los metodos de resolución de ecuaciones de distinto orden se puede encontrar en ANALISIS NUMERICO < BURDEN-FAIRES >.

****

Nuestro problema entonces queda expresado solo como una recta del tipo:

$$y = ax + b$$

donde a y b son las constantes que minimicen el error de los minimos cuadrados o como el resultado de un polinomio del tipo :

$$y = Ao + A1*x + A2*x^2 + \ldots\ldots + An*x^n$$

En nuestro caso solo quedaria un polinomio cuyo grado maximo es n=4,por lo que el polinomio aproximante para una variable que depende de otra nos quedaria de la siguiente forma:

$$y = Ao + A1*x + A2*x^2 + A3*x^3 + A4*x^4$$

4.- CARACTERISTICAS DEL PROGRAMA DE MANEJO DE DATOS.

El programa es parte de un menú principal el cuál permite realizar otras operaciones en el sistema, una de las cuales crea el archivo del que se tomaran datos para la ejecución de este programa.

Ambos programas, el principal y el de cálculos estadísticos y numericos estan hechos en un lenguaje de programación de alto nivel para una mayor facilidad en hacer modificaciones o mejoras posteriores; el lenguaje utilizado es el lenguaje de programación C debido a su facilidad para el manejo de gráficos y rutinas. (3)

4.1.- DESCRIPCION DEL PROGRAMA.

El programa esta hecho en el LENGUAJE DE PROGRAMACION C que es un lenguaje de alto nivel y permite el acceso a la base de datos relacionales (en nuestro caso se trata de un manejo de archivos que se renuevan todos los dias).

Los datos son almacenados en arreglos (ptemp y dato) y los resultados en una estructura (muestra), que permite incluso almacenar el intervalo de tiempo de la observación.

Presenta 4 opciones :

-Creación de una tabla de MEDIA,MEDIANA Y VARIANZA de distintas muestras para hacer un analisis comparativo.
(procedimiento adqmuestra)

-Creación de un HISTOGRAMA y una OJIVA con la presentación de la distribución de frecuencias y distribución porcentual de las muestras.
(procedimiento histojiv)

-Creación de un POLINOMIO DE APROXIMAION con la ayuda de los datos tomados y la posibilidad de evaluar datos para observación.
(procedimiento aproxpol)

El programa se aborta si existe un mal ingreso al sistema y concluye la adquicición de datos automaticamente;

permite salir del menú con una opción de salida hacia el menú principal.

4.1.1.- **PROCESAMIENTO DE DATOS.-**

Los datos pueden ser procesados de dos maneras:

-POR METODOS NUMERICOS

-POR CALCULOS ESTADISTICOS

4.1.2.- **PROCESAMIENTO DE DATOS POR METODOS NUMERICOS.**

Se ha creado un programa que por medio de aproximaciones numericas crea un POLINOMIO DE APROXIMACION de datos experimentales provenientes de mediciones en la cuál una variable depende de otra variable, el polinomio de aproximación puede ser hasta de grado 4 con lo cuál los datos que se evaluan posteriormente son una buena aproximación de una interpolación o extrapolación de datos del sistema anteriormente medido.

Esta parte fue creada considerando que el operador este interesado en conocer algo más que la temperatura del sistema, como por ejemplo el comportamiento de la producción con respecto al tiempo, o cantidad de refrigerante con respecto a la cantidad de producto, conociendo de antemano mediciones ya tomadas.

4.1.3.- PROCESAMIENTO DE DATOS CALCULOS ESTADISTICOS.

Los datos de la planta, esto es temperatura en un tiempo determinado del proceso, pueden ser observados de dos diferentes maneras:

-POR MEDIO DE ESTIMADORES

-POR MEDIO DE GRAFICOS REPRESENTATIVOS

4.1.4.- OBSERVACION DE DATOS POR MEDIO DE ESTIMADORES

Los datos pueden ser observados en el momento de la llegada, dependiendo de la hora en que se realice la muestra; esto es que el operador puede sacar una conclución de que el proceso esta marchando bien a partir de la primeras horas de la inicialización del sistema, esto se debe a que el archivo de llegada de los datos se llena cada hora con datos tomados cada 5 minutos (en la primera hora el archivo esta vacio); esta toma de muestras la puede realizar hasta 5 veces en una corrida del programa. (Las muestras pequenas no son una representación confiable de lo que sucede en el sistema).

Una vez completado el ciclo del proceso, esto es desde las 08:00 H hasta las 22:00 H, al siguiente dia el operador puede sacar una evaluación de los datos durante cada hora del proceso del dia anterior.

4.1.5.- OBSERVACION DE LOS DATOS POR MEDIO DE GRAFICOS REPRESENTATIVOS

Los gráficos representativos a los que se hacen referencia son los gráficos correspondientes al histograma,ojiva y polígono de frecuencia de los datos correspondientes al intervalo de tiempo que se quiera estimar.

Estos datos que son evaluados pueden ser observados de un dentro del programa hasta 2 veces (el programa permite hacer un cambio en la magnitud del gráfico), con esto el operario puede observar lo que sucede en el proceso conforme pasa el tiempo u observar los datos que se tomaron el día anterior, con esto podemos ver los valores que más concurrencia tuvieron dentro del proceso, en el tiempo de la muestra tomada.

Para realizar los gráficos se utilizo la opción de modo gráfico que se puede inicializar en cualquier momento dentro de un programa en C (normalmente se trabaja en modo texto), dentro del programa existen 4 rutinas que crean el modo gráfico, estas son HISTO,OJIV,POLFRC y GRAFAPROX.¡2¿.

****

NOTA 2. Los metodos y modos gráficos y los medios de inicialización de este modo se pueden encontrar el el capitulo 17 y 18 en TURBO C BIBLE <Barkakati>.

****

En HISTO se crea el histograma, en OJIV se crea la ojiva, en POLFRC se crea el polígono de frecuencias y en GRAFAPROX se realiza un gráfico de los puntos evaluados del polinomio de aproximación encontrado teniendo como única condición el de colocar los puntos a evaluar de forma ORDENADA DE MENOR A MAYOR.

Cada subrutina dispone de punteros que le permiten usar los datos que son ingresados y cada uno de ellos abre y cierra el modo gráfico para mayor comodidad, aunque puede mantenerse abierto mientras dura la presentación de los gráficos y cerrarla despues.

Como C permite la creación de programas en forma modular pueden usarse las rutinas independientemente tomando en cuenta solo las variables de entrada y de salida de dicho procedimiento; esta es la razón por la cuál se separaron ciertos procedimientos que podian ser usados de modo externo y que fueron colocados en la libreria INCLUDE.C del C.

Este archivo es el archivo FUNCTION.H que se adjunta con el programa y que necesita ser instalado en el INCLUDE.C para su correcto funcionamiento.

Despues de salir del programa vuelve al menú principal, con lo cuál el operario obtiene una serie de archivos etiquetados como *.R en el cuál se pueden observar las mediciones hechas.

5.- CONCLUSIONES.-

El programa permite al usuario sacar una conclución de lo que esta pasanado en el sistema en el mismo instante en que esta ocurriendo el proceso o el día despues,por medio de estimadores o gráficos representativos.

El usurario puede el dia posterior al proceso,antes de la inicialización colocar los datos en una copia para ser procesados posteriormente.

Todos los datos, resultados de los programas llamados, son dirigidos al DRIVE A: por lo que el usuario debera tener un disco para la salvaguardia de esta información y su posterior analisis.

Para la ejecución de este programa es necesario cargar en el directorio INCLUDE de la libreria estandar del lenguaje C el programa FUNCTION.C que acompaña a este programa ejecutable.

A P E N D I C E

A.- MANUAL DEL USUARIO.

M A N U A L D E L U S U A R I O .

INTRODUCCION.

Para poder operar el Sistema de Control SCADA diseñado, es necesario que sus componentes estén instalados en su totalidad.

Los dispositivos que conponen el sistema SCADA son los siguientes:

Unidad Terminal Remota.

Modems del tipo everex.

Lineas telefónicas.

Computador PC o Compatible.

Cable de comunicación RS232-C.

Computador PS-60 o Compatible.

DESCRIPCION DE LOS COMPONENTES.

Unidad Terminal Remota.

La unidad terminal remota (UTR), se encuentra ubicada junto a la planta a controlar. Está formada por el panel de control y el kit del microprocesador 8080. El panel de control es el encargado de actuar sobre los dispositivos eléctricos que forman parte de la planta, los cuales permiten al 8080 sensar los estados de los contactos de la planta.

El panel de control posee puertos por medio de los cuales se energizan a los componentes de la planta. Están etiquetados por lo que se debe proceder a contectar cada uno

respectivamente. Se debe energizar desde una fuente de 220 Vac. Además posee un conector del tipo DB25 el cuál nos servirá para conertar el modem y otro para conectar el kit de 8080.

El kit 8080 tiene como función realizar el control de todos los dispositivos y realizar la transmisión y recepción de información desde y hacia la planta a través del modem.

Se ha implementado en una memoria EPROM los programas necesarios para este proyecto, por lo que solamente es necesario conocer como proceder cuando se inicia la comunicación, y como proceder en caso de falla.

Modems.

Los modems tienen como función realizar el enlace y la comunicación entre la planta y el control maestro a través de dos lineas telefónicas una para cada lado de la comunicación. Los registros del modem han sido seteados de acuerdo a las necesidades de nuestro enlace, por lo que no es necesario alterar estos valores, ya que quedan grabados en los mismos permanentemente. Para la conección de los modems se requieren de cajetines telefónicos adecuados, para conectar el modem a la linea telefónica. Además requiere que estén conectados a una fuente de alimentación de 120 Vac a través de un transformador el cuál viene junto con el modem. En la planta se debe conectar el modem al panel de control a través de un conector DB25, mientras que en el centro de control se debe conectar el computador PC al modem a través de un cable DB25.

Lineas telefónicas.

Se requiere de dos lineas telefónicas las cuales deberán estar ubicadas una en la planta, y otra en el centro de control. Estas lineas no podrán ser utilizadas para otros fines, mientras se esté realizando la comunicación entre la planta y centro de control ya que durante este periodo estas lineas son utilizadas por los modems. Las lineas telefónicas pueden ser analógicas o digitales. El llamado es realizado automaticamente desde el modem del centro de control al modem de la planta el cual contesta la llamada y establece el enlace por lo que no es necesario tener un aparato telefónico conectado en la linea de la planta.

Computador PC o Compatible.

El computador a utilizar en el centro de control puede ser del tipo PC o compatible, debido principalmente a su bajo costo, aun que se podría usar otros modelos mas caros. Las características que debe cumplir este computador son las siguientes: Deberá tener por lo menos 320K de memoria RAM, deberá tener un monitor BW o COLOR de alta resolución debido a que durante la visualización se usan gráficos en alta resolución. Deberá poseer dos puertos seriales con conectores del tipo DB25, por medio de los cuales se realizará la comunicación a la planta a través del modem, y la

comunicación con el centro de estadística a través de un cable físico del tipo DB25.

Cable de Comunicacion RS232-C

El cable de comunicación RS232-C es utilizado para conectar el computador PC del operario con la computadora PS-60 que está ubicada en el centro de estadística. Se utilizan conectores del tipo DB25. Cabe señalar que el centro de estadística está situado en otro cuarto cercano al cuarto de control, lo cual hace posible realizar este tipo de conección.

Computador PS-60 o Compatible.

El Computador PS-60 o Compatible es utilizado para realizar las estadísticas del proceso realizado en la planta.
Dentro de las características requeridas para este Computador estan las siguientes : Un Mega Ryte de memoria RAM , 1 puerto serial RS232-C con conector del tipo DB-25 , un monitor BW o COLOR de alta Resolución con la finalidad de poder observar los gráficos Estadisticos , Disco duro , cuya capacidad sea de 10 Mb. o mayores .

OPERACION DEL SISTEMA SCADA .

Para operar el Sistema SCADA diseñado, se requiere que el

usuario siga en ORDEN cada uno de los siguientes pasos .

1 .- ENLACE TELEFONICO .

Para realizar el enlace telefónico, es necesario que el Computador del Operador sea arrancado con el Sistema Operativo DOS, Luego de esto usando el Disco Flexible , cuyo contenido es el Programa de Comunicación del modem, escribimos el comando **BITCOM** , con el cual ingresamos al programa de comunicación del modem . Escojemos la opcion **1** con el cual ingresamos al menu de registros de llamadas en donde deberá estar el número de teléfono de la Planta , moviendo el cursor al registro correspondiente al número de la Planta , donde presionamos la tecla **D** , luego de lo cual el programa hace la llamada a traves del modem al número respectivo. Si la llamada es realizada correctamente, los modem enlaran automaticamente , indicando este por medio del mensaje **CONECT 1200** , con el cual el enlace entre los modems estará realizado. Si no se produce el enlace, la causa de la falla sera mostrada en la pantalla . Para regresar al menú de registro de llamada se debera pulsar la tecla **F2**, si se desea reintentar la llamada se debera repetir los pasos anteriores y pulsar nuevamente la tecla **D**. Para salir del menú de llamadas, se debe presionar **F2** desde cualquier pantalla se regresa a la anterior por medio de este tecla. Luego de realizado el enlace y entando en el menú principal

seleccionamos la opción 4 con la cual salimos al DOS, desde donde continuamos los pasos siguientes.

2.- EJECUCION Y MONITOREO DEL PROCESO

EN LA ESTACION MAESTRA.

Luego de realizado el enlace telefónico procedemos a ejecutar la edición de un programa en lenguaje de procesos, para así tener el control de la planta.

Para ello ingresamos el disco flexible en el cual tenemos el bloque de programas ejecutables llamado **Planta.Bat** con el cual digitando este mismo nombre realizamos la ejecución de los siguientes programas de bloque.

PUERTO1.EXE

Por medio de este programa configuramos los parámetros de comunicación del puerto serial número 1, los cuales deberán ser idénticos a los ingresados en la configuración del puerto serial del computador PS - 60 del CENTRO DE ESTADISTICA.

P2.EXE

Cuando se ejecuta este programa automáticamente realiza la configuración del puerto serial número 2 del computador PC luego de lo cual muestra el mensaje de configuración realizada. Los parámetros de configuración son fijos y están de acuerdo a la velocidad de comunicación con el modem.

BACKUP.EXE

Al ejecutarse este programa se resaliza la edición de un programa en lenguaje de procesos, estos serán digitados en un editor de pantalla, al mismo que se ingresa , al ingresar la

opción adecuada del menú de selección, luego de implementada la acción del día a tratarse en la planta , se graba dicho programa , para luego proceder a la compilación del mismo, lo que se realiza al escoger la opción adecuada del menú de selección, una vez realizada la compilación del mismo se procede a transmitir dicho programa compilado al controlador que en nuestro caso se trata de un Micro Procesador 8080.

TOPICO1.COM.

Cuando el programa a sido cargado en la planta empieza a ejecutarse el programa de visualización de la planta el cual muestra dos pantallas con las cuales el operador podrá observar los estados de contactos y temperatura. Pulsando la tecla F1 el operador puede observar la primera pantalla del operador, con la tecla F2 cambia a la segunda pantalla si hay algun tipo de problema en la planta el operador puede pulsar la tecla ESC que es la tecla de emergencia con la cual se apaga la planta e informa de esto al operador. Cada hora este programa termina de ejecutar y transmite el archivo de temperatura y tiempo al centro de estadística. Luego de transmitido este archivo automaticamente nuevamente se inicia el programa de visualización sin necesidad de volver a iniciar todo el proceso .

3.- EJECUCION DEL PROCESO EN LA PLANTA.

Luego de establecida la llamada desde el centro de control el operario que se encuentra en la planta debe ejecutar el programa controlador del 8080, para lo cual debe escribir los comandos

address 0400

Run

Con lo cual se ejecuta el programa que se encarga de esperar cargar y transmitir los datos desde y hacia el centro de control.

En caso de producirse una falla en la transmisión o se ordene apagar desde el centro de control, en la pantalla del kit 8080 aparecerá la letra C con lo cual el operario deberá cargar el programa de control manual, por medio de la casetera por medio de los siguientes comandos.

addr (direccion de inici del programa) mem

addr 03AE

prenda casetera

run.

NOTA:

Si desea mayor detalles de funcionamiento por favor consultar el capítulo respectivo.

B .- LISTADOS DE LOS PROGRAMAS .

```
PROGRAM EDITOR;
{llamado a subrutinas externas en assembler}

{$L M_CURSOR.OBJ}
{$L VENTANA.OBJ}
{$L MENU.OBJ}
{$L MENU1.OBJ}
{$L SUBIR.OBJ}
{$L BAJAR.OBJ}
{$L IZQUIERD.OBJ}
{$L DERECHA.OBJ}
{$L PAGE1.OBJ}
{$L PAG1.OBJ}
{$L EXPLORE.OBJ}
{$L LIMPIAR.OBJ}

 type
     comando = record
   coma                :packed array[1..20] of char;
   linea               :integer;
       end;
var
    sentence                      : file of comando;
    sentences                     :text;
    respaldo,sentencia,apoyo      :array [1..100] of comando;
    ch                            :char;
    a,j,i,opc,variable,h,p,k,s,l  :integer;
    fijo,us,siempre,scan,ascii    :integer;
    nuevo,valor                   :integer;
    com                           :packed array [1..20] of char;
    done                          :boolean;
    sentencename                  :string[20];

{procedimientos externos}

       procedure ventana;external;
       procedure menu;external;
       procedure menu1;external;
       procedure m_cursor(colupar:integer;filapar:integer);external;
       procedure pintando;
       begin
     ventana;
     menu;
     menu1;
       end;
       procedure subir;external;
       procedure bajar;external;
       procedure izquierd;external;
       procedure derecha;external;
       procedure page1;external;
       procedure explore(var scan,ascii:integer);external;
       procedure limpiar;external;
       procedure pag1(var valor:integer);external;
{-------------------------------------------------------------------------}
```

```
{procedimientos del programa}

procedure Inicialice;
begin
  pintando;
  m_cursor(0,0);
end; { Inicialice }
{-------------------------------------------------------------------------}
Procedure Menuayuda;
{ imprime en una ventana que comandos y funciones se pueden ejecutar }

begin
   Writeln('          MENU DE AYUDA             ');
   writeln('------------------------------------ ');
   writeln('       Comandos                     ');
   writeln('                                    ');
   writeln('       prender(opcion1);            ');
   writeln('       apagar(opcion2);             ');
   writeln('       espere(opcion3);             ');
   writeln('                                    ');
   writeln('       Opciones:                    ');
   writeln('       1,2: a->Calentador           ');
   writeln('            b->Agitador             ');
   writeln('            c->Bomba Recirculacion  ');
   writeln('         3: t-> tiempo              ');
   writeln('            T-> Temperatura         ');
   writeln('                                    ');
   writeln(' F2-edita textos con extension SCP  ');
   writeln(' F3-Limpia la pantalla              ');
   writeln(' F4-Graba el texto editado          ');
   writeln(' Esc-Sale del Editor sin grabar     ');
   writeln('------------------------------------');
   writeln(' Digite Esc para Salir Menu Ayuda   ');
   exit;
end; { Menuayuda }

{-------------------------------------------------------------------------}
Procedure Ayudar;
{ hace la ventana de ayuda para el usuario }

var ch1                              :char;
    hecho                            :boolean;
begin
     hecho:=false;
     m_cursor(0,0);
     menuayuda;

     repeat
       explore(scan,ascii);
       if scan=$1B then hecho:=true;
       scan:=0;ascii:=0;
     until hecho;
     scan:=0;ascii:=0;
End; { Menu de Ayuda }
```

```
{-------------------------------------------------------------------------}
Procedure Editar;
var
   x                        :char;
   hecho                    :boolean;
   valor                    :integer;
Begin
     scan:=0;ascii:=0;valor:=0;
     for i:=1 to 100 do
     begin
   with sentencia[i] do
   begin
       for j:=1 to 20 do
       coma[j]:=' ';
   end;
     end;
     for i:=1 to 100 do  sentencia[i].linea:=i;
     hecho:=false;
     pintando;
     m_cursor(0,0);
     repeat
  writeln(' Comandos exclusivos del Editor ');
  writeln(' prender(op); -- apagar(op); -- esperar(opc);  ' );
  writeln(' op:a,b,c -- opc: T,t ' );
  writeln(' a:calentador, b:agitador, c:bomba recirculacion ');
  writeln(' T: temperatura, t: tiempo ');
  writeln;
  write(' Digite @ para editar---> ');readln(x);
  if x= '@' then hecho:=true
     else pintando;
     until hecho;
     pintando;
     m_cursor(0,0);
     p:=100;
     for i:=1 to p do
  begin
     page1;
     with sentencia[i] do
       begin
  write(linea:3);Write(coma[1]);
  j:=2;
  repeat
     read (ch);
     coma[j]:=ch;
     j:=j+1;
  until eoln;
  readln;
  fijo:=i;
  if coma[2]=' ' then i:=p;
       end;
     valor:=0;
  end;
     if fijo<>100 then fijo:=fijo-1;
     ascii:=0;
```

```
    for i:=1 to fijo do  respaldo[i]:=sentencia[i];
    valor:=0; siempre:=fijo;
    us:=fijo;
end;    {Editar}
{-------------------------------------------------------------------------}
procedure grabar;
var
    util                    :comando;
begin
     pintando;
     write('Digite nombre del archivo a grabar-> ');
     readln(sentencename);
     assign(sentence,sentencename);
     rewrite(sentence);
     pintando;
     writeln(' Se editaron ',us,' lineas');
     writeln;
     for i:=1 to us do
   begin
    util:=sentencia[i];
    write(sentence,util);
   end;
      close(sentence);
     writeln('Archivo copiado con nombre ',sentencename);
     write('Digite <ENTER> para seguir...');
     readln;
     pintando;
end;
{-------------------------------------------------------------------------}
procedure mostrar;
var
   op                           :char;

begin
    valor:=0;
    pintando;
    repeat
 write(' Desea el Archivo original o el modificado (O/M): ');
 readln(op);pintando;
    until ((op='O') or (op='M'));
    case op of
      'O': begin
      pintando;
      for i:=1 to SIEMPRE do
    begin
       valor:=0;
       with respaldo[i] do
     begin
       pag1(valor);
       if valor=1 then
          begin
      g_cursor(1,20);
      WRITE('Presione ENTER<CR> para
      seguir');
```

```
      readln;
      pintando
          end;
       writeln(linea:3,coma);
    end;
   end;
    end;
      'M': begin
       pintando;
       for i:=1 to fijo do
    begin
       valor:=0;
       with sentencia[i] do
   begin
     pagi(valor);
     if valor=1 then
        begin
          m_cursor(1,20);
          write('Presione ENTER<CR> para
          seguir');
          readln;
          pintando;
        end;
     writeln(linea:3,coma);
   end;
    end;
    end;
    end;
    valor:=0;
end;  {mostar}
{-------------------------------------------------------------------------}
procedure borrar;
var
   lugar                    :integer;
   opc                      :char;
begin
     pintando;
     write('Digite la Linea a borrar-->');
     readln(lugar);
     if lugar>fijo then begin
        writeln('No existe esa Linea');
        readln;pintando;
        exit;
     end;
     write('Seguro que desea borrar la linea ',lugar,' ? (S/N): ');
     readln(opc);
     if ((opc='S') or (opc='s'))
     then begin
      for i:=1 to (lugar-1) do apoyo[i]:=sentencia[i];
      for i:=lugar to (fijo-1) do apoyo[i]:=sentencia[i+1];
      fijo:=fijo-1;
      for i:=1 to fijo do sentencia[i]:=apoyo[i];
      for i:=1 to fijo do
   begin
```

```
        sentencia[i].linea:=i;
        apoyo[i].linea:=i;
    end;
    end
      Else
    begin
       writeln(' No hay cambios ');
       readln;pintando;
       exit;
    end;
      writeln('Linea borrada');
      us:=fijo;
      write('Digite <ENTER> para seguir...');
      readln;
      pintando;
end;
{-------------------------------------------------------------------}
procedure insertar;
var
   lugar,numli,hasta                    :integer;
begin
     hasta:=0;
     pintando;
     write('Digite la Linea a Insertar--> ');readln(lugar);
     if (lugar>=100)
     then begin
     writeln('No es posible insertar mas lineas');
     write('Digite <ENTER> para salir...');
     readln;
     pintando;
     exit;
   end;
     if (lugar>fijo)
     then begin
     pintando;
     write('Cuantas lineas va a insertar-> ');
     readln(numli);
     pintando;  hasta:=lugar+numli;
     if hasta>100
     then begin
       writeln('No se puede insertar mas lineas');
       write('Digite <ENTER> para salir...');
       readln;
       pintando;
       exit;
   end;
     for i:=lugar to  hasta do
     begin
   page1;
   with sentencia[i] do
   begin
      write(linea:3,coma[1]);
      j:=1;
      repeat
```

```
      j:=j+1;
      read(ch);
      coma[j]:=ch;
  until eoln;
  readln;
  fijo:=i;
  if coma[2]=' ' then i:=hasta
end; {with}
  end;     {for}
  for i:=1 to fijo do apoyo[i]:=sentencia[i];
 end {then}
  else begin
   for i:=1 to (lugar-1) do apoyo[i]:=sentencia[i];
   with apoyo[lugar] do
   begin
 linea:=lugar;
 for j:=1 to 20 do coma[j]:=' ';
 write(linea:3,coma[1]);
 j:=1;
 repeat
     j:=j+1;
     read(ch);
     coma[j]:=ch;
 until eoln;
 readln;
 if coma[2]=' ' then
 begin
    writeln(' No hay cambios');
    write('Digite <ENTER> para salir...');
    readln;pintando;
    exit;
 end;
  end;
  lugar:=lugar+1;
  fijo:=fijo+1;
  if fijo=101 then
  begin
     writeln(' No se puede insertar ');
     write('Digite <ENTER> para salir...');
     readln;
     pintando;exit;
  end;
  for i:=lugar to fijo do apoyo[i]:=sentencia[i-1];
  for i:=1 to fijo do  sentencia[i]:=apoyo[i];
  for i:=1 to fijo do
  begin
 sentencia[i].linea:=i;
 apoyo[i].linea:=i;
  end;
  end; {else}
  writeln('Lineas insertadas'); us:=fijo;
  write('Digite <ENTER> para continuar...');
  readln;
  pintando;
```

```
end; {insertar}
{-------------------------------------------------------------------------}
procedure busqueda;
var
   textos                :packed array[1..20] of char;
   i,e                   :integer;
   p                     :char;
begin
     pintando;
     m_cursor(30,1);
     write(' Ingrese el comando a buscar:  ');
     for i:=1 to 20 do textos[i]:=' ';
     i:=2;
     repeat
    read(p);
    textos[i]:=p;
    i:=i+1;
     until eoln;
     for i:=1 to fijo do
  begin
  with sentencia[i] do
       begin
   if textos=coma
   then begin
   m_cursor(30,3);
   writeln('linea: ',linea:3);
   readln;
   end;
       end;
  end;
     write('Digite <ENTER> para seguir...');
     readln;
     pintando;
     exit;
end; { Busqueda }

{-------------------------------------------------------------------------}

procedure reemplazar;
var     i,j,k                       :integer;
begin
     SCAN:=0;ASCII:=0;
     M_cursor(30,1);
     write('Escriba la linea a cambiar:');
     readln(k);
     M_cursor(30,2);
     write('Anterior: ',sentencia[k].coma);
     M_cursor(30,3);
     write('Nuevo: ');
     with sentencia[k] do
    begin
      for i:=1 to 20 do coma[i]:=' ';
      j:=2;
      repeat
```

```
  read(ch);
  coma[j]:=ch;
  j:=j+1;
      until eoln;
    end;
     for i:=1 to fijo do apoyo[i]:=sentencia[i];
end; { reemplazar }
{-------------------------------------------------------------------------}
procedure cargar;
var
   carga                    :ARRAY[1..100] OF COMANDO;
   util                     :comando;
   archi                    :file of comando;

begin
     limpiar;
     for i:=1 to 100 do
     begin
  with  carga[i] do
  begin
    for j:=1 to 20 do coma[j]:=' ';
    linea:=i;
  end;
     end;
     assign(archi,com);
     reset(archi);
     i:=0;
     while not eof(archi) do
  begin
       i:=i+1;
       read(archi,util);
       carga[i]:=util;
       sentencia[i]:=carga[i];
       writeln; '
  end;
     nuevo:=i;
     close(archi);
     writeln('Archivo ',com,' copiado ');
     write('Digite <ENTER> para seguir...');
     readln;
     limpiar;
end; { cargar }
{-------------------------------------------------------------------------}


begin
limpiar;
opc:=0;
while opc<>4 do
BEGIN
  limpiar;
  writeln;
  writeln('            SISTEMAS DE CONTROL DE PROCESOS         ver 1.0');
  writeln;
```

```
writeln('         ESCUELA SUPERIOR POLITECNICA DEL LITORAL');
writeln;
writeln;
writeln('                  SELECCIONE UNA OPCION  :');
writeln;
writeln;
writeln;
writeln(' 1._ EDITAR UN PROGRAMA');
writeln;
writeln(' 2._ COMPILAR UN PROGRAMA');
writeln;
writeln(' 3._ CARGAR UN PROGRAMA AL CONTROLADOR');
writeln;
writeln(' 4._ RETORNO A DOS');
writeln;
writeln;
write ('                  INGRESE UNA OPCION :  ');
readln(opc);
CASE opc OF
    1:   begin { programa Principal }
  LIMPIAR;
  Inicialice;
  Done:=False;
  scan:=0;ascii:=0;valor:=0;
  repeat
     SCAN:=0;ASCII:=0;
     explore(scan,ascii);
     case scan of
  $0:   { Teclas Funcion 1--10 }
     begin
      Case ascii of
 $3B:          { F1 }
     begin     {MENU AYUDA}
       Ayudar;
       pintando;
     end;
 $3C: pintando;  { F2  RESETEAR }
 $3D: insertar;  { F3  INSERT }
 $3E: borrar;    { F4  DEL }
 $3F: grabar;    { F5  GRABAR }
 $40: busqueda;  { F6  BUSCA  }
 $41: begin
  reemplazar;   { F7  REEMPLAZA }
  pintando;
      end;
 $42: mostrar;         { F8  MOSTRAR }
 $43: editar;          { F9  EDITOR LINEA}
 $47: m_cursor(0,0);   { INICIO (HOME) }
 $48: subir;           {/\}
 $4B: izquierd;        {<-}
 $4D: derecha;         {->}
 $4F: m_cursor(1,20);  { FIN (END) }
 $50: bajar;           {\/}
      end;
```

```
      end;
    $1B: Done:=True;            { Esc }
      end;
    Until Done;
    limpiar;
    end;
      2: ;
      3: ;
  end;  { CASE }
END;  {while}
limpiar;
end.
```

```
CODE            SEGMENT         PARA    PUBLIC 'CODE'
                ASSUME          CS:CODE
                PUBLIC          M_CURSOR
FILAPAR         EQU             4¡BP¿    ; localidad del parametro fila
COLUPAR         EQU             6¡BP¿    ; localidad del parametro columna

M_CURSOR        PROC            NEAR
                PUSH            BP
                MOV             BP,SP
                PUSH            DS

                MOV             DH,FILAPAR
                MOV             DL,COLUPAR
                MOV             AH,02
                SUB             BH,BH
                INT             10H

                POP             DS
                MOV             SP,BP
                POP             BP
                RET             4
M_CURSOR        ENDP
CODE            ENDS
                END
```

```
CODE      SEGMENT   PUBLIC
          ASSUME    CS:CODE
          PUBLIC    IZQUIERD

IZQUIERD  PROC      NEAR
                PUSH            BP
                MOV             BP,SP
                PUSH            DS

                MOV             AH,03
                MOV             BH,00
                INT             10H
                MOV             BL,DL
                CMP             BL,00
                JE              SALIDA

                MOV             AH,DH   ;FILA
                MOV             AL,DL   ;COLUMNA
                DEC             AL
                MOV             DH,AH
                MOV             DL,AL
                CALL            POSCUR

SALIDA:
                POP             DS
                MOV             SP,BP
                POP             BP
                RET
IZQUIERD        ENDP

POSCUR    PROC
          MOV       AH,02
          MOV       BH,00
          INT       10H
          RET
POSCUR    ENDP

CODE      ENDS
          END
```

```
CODE            SEGMENT         PUBLIC
                ASSUME          CS:CODE
                PUBLIC          VENTANA

VENTANA         PROC            NEAR
                PUSH            BP
                MOV             BP,SP
                PUSH            DS


                CALL            A10CLR          ;limpia la pantalla
                MOV             DX,1600H
                CALL            POSCUR
                CALL            C10MENU         ; Crea la pantalla de informe
                MOV             DX,1604H
                CALL            POSCUR
                MOV             AL,'F'
                CALL            D10ESCRIBE
                MOV             DX,1605H
                CALL            POSCUR
                MOV             AL,'I'
                CALL            D10ESCRIBE
                MOV             DX,1606H
                CALL            POSCUR
                MOV             AL,'-'
                CALL            D10ESCRIBE
                MOV             DX,1607H
                CALL            POSCUR
                MOV             AL,'A'
                CALL            D10ESCRIBE
                MOV             DX,1608H
                CALL            POSCUR
                MOV             AL,'Y'
                CALL            D10ESCRIBE
                MOV             DX,1609H
                CALL            POSCUR
                MOV             AL,'U'
                CALL            D10ESCRIBE
                MOV             DX,160AH
                CALL            POSCUR
                MOV             AL,'D'
                CALL            D10ESCRIBE
                MOV             DX,160BH
                CALL            POSCUR
                MOV             AL,'A'
                CALL            D10ESCRIBE

        MOV     DX,1710H
        CALL    POSCUR
        MOV     AL,'F'
        CALL    D10ESCRIBE
        MOV     DX,1711H
        MOV     AL,'A'
        CALL    POSCUR
        CALL    D10ESCRIBE
        MOV     DX,1712H
        CALL    POSCUR
```

```
            MOV     AL,'-'
            CALL    D10ESCRIBE
            MOV     DX,1713H
            CALL    POSCUR
            MOV     AL,'E'
            CALL    D10ESCRIBE
            MOV     DX,1714H
            CALL    POSCUR
            MOV     AL,'D'
            CALL    D10ESCRIBE
            MOV     DX,1715H
            CALL    POSCUR
            MOV     AL,'I'
            CALL    D10ESCRIBE
            MOV     DX,1716H
            CALL    POSCUR
            MOV     AL,'T'
            CALL    D10ESCRIBE
            MOV     DX,1717H
            CALL    POSCUR
            MOV     AL,'A'
            CALL    D10ESCRIBE
            MOV     DX,1718H
            CALL    POSCUR
            MOV     AL,'R'
            CALL    D10ESCRIBE


                    POP             DS
                    MOV             SP,BP
                    POP             BP


                    RET
VENTANA             ENDP



;                   LIMPIA PANTALLA
;                   ---------------

A10CLR              PROC            NEAR
                    MOV             AX,0600H
                    MOV             BH,07
                    MOV             CX,0000
                    MOV             DX,184FH
                    INT             10H
                    RET
A10CLR              ENDP

;                   POSICIONA CURSOR
;                   ----------------

POSCUR              PROC
                    MOV             BH,00
                    MOV             AH,02
                    INT             10H
                    RET
POSCUR              ENDP
```

```
;               CREA PANTALLA DE MENU
;               ---------------------

C10MENU         PROC
                MOV             CX,0320
                MOV             BL,05FH
                MOV             BH,00
                MOV             AH,09
                MOV             AL,00
                INT             10H
                RET
C10MENU         ENDP


;               ESCRIBE CON ATRIBUTO
;               --------------------

D10ESCRIBE      PROC            NEAR
                MOV             AH,09
                MOV             BH,00
                MOV             BL,05FH
                MOV             CX,01
                INT             10H
                RET
D10ESCRIBE      ENDP
;-------------------------------------------------------------------------

CODE            ENDS
                END
```

```
CODE        SEGMENT     PUBLIC
            ASSUME      CS:CODE
            PUBLIC      MENU

MENU        PROC        NEAR

            PUSH        BP
            MOV         BP,SP
            PUSH        DS

            MOV         DX,160EH
            CALL        CURSOR
            MOV         AL,'F'
            CALL        ESCRIBE
            MOV         DX,160FH
            CALL        CURSOR
            MOV         AL,'2'
            CALL        ESCRIBE
            MOV         DX,1610H
            CALL        CURSOR
            MOV         AL,'-'
            CALL        ESCRIBE
            MOV         DX,1611H
            CALL        CURSOR
            MOV         AL,'R'
            CALL        ESCRIBE
            MOV         DX,1612H
            CALL        CURSOR
            MOV         AL,'E'
            CALL        ESCRIBE
            MOV         DX,1614H
            CALL        CURSOR
            MOV         AL,'E'
            CALL        ESCRIBE
            MOV         DX,1613H
            CALL        CURSOR
            MOV         AL,'S'
            CALL        ESCRIBE
            MOV         DX,1615H
            CALL        CURSOR
            MOV         AL,'T'
            CALL        ESCRIBE

            MOV         DX,1618H
            CALL        CURSOR
            MOV         AL,'F'
            CALL        ESCRIBE
            MOV         DX,1619H
            CALL        CURSOR
            MOV         AL,'3'
            CALL        ESCRIBE
            MOV         DX,161AH
            CALL        CURSOR
            MOV         AL,'-'
            CALL        ESCRIBE
            MOV         DX,161BH
            CALL        CURSOR
            MOV         AL,'I'
```

```
        CALL            ESCRIBE
        MOV             DX,161CH
        CALL            CURSOR
        MOV             AL,'N'
        CALL            ESCRIBE
        MOV             DX,161DH
        CALL            CURSOR
        MOV             AL,'I'
        CALL            ESCRIBE
        MOV             DX,161EH
        CALL            CURSOR
        MOV             AL,'C'
        CALL            ESCRIBE
        MOV             DX,161FH
        CALL            CURSOR
        MOV             AL,'I'
        CALL            ESCRIBE
        MOV             DX,1620H
        CALL            CURSOR
        MOV             AL,'O'
        CALL            ESCRIBE

MOV     DX,1623H
CALL    CURSOR
MOV     AL,'F'
CALL    ESCRIBE
MOV     DX,1624H
CALL    CURSOR
MOV     AL,'4'
CALL    ESCRIBE
MOV     DX,1625H
CALL    CURSOR
MOV     AL,'-'
CALL    ESCRIBE
MOV     DX,1626H
CALL    CURSOR
MOV     AL,'F'
CALL    ESCRIBE
MOV     DX,1627H
CALL    CURSOR
MOV     AL,'I'
CALL    ESCRIBE
MOV     DX,1628H
CALL    CURSOR
MOV     AL,'N'
CALL    ESCRIBE

MOV     DX,162BH
CALL    CURSOR
MOV     AL,'F'
CALL    ESCRIBE
MOV     DX,162CH
CALL    CURSOR
MOV     AL,'5'
CALL    ESCRIBE
MOV     DX,162DH
CALL    CURSOR
MOV     AL,'-'
CALL    ESCRIBE
MOV     DX,162EH
CALL    CURSOR
MOV     AL,'C'
```

```
            MOV       AL,'R'
            CALL      ESCRIBE
            MOV       DX,1630H
            CALL      CURSOR
            MOV       AL,'A'
            CALL      ESCRIBE
            MOV       DX,1631H
            CALL      CURSOR
            MOV       AL,'B'
            CALL      ESCRIBE
            MOV       DX,1632H
            CALL      CURSOR
            MOV       AL,'A'
            CALL      ESCRIBE
            MOV       DX,1633H
            CALL      CURSOR
            MOV       AL,'R'
            CALL      ESCRIBE

                  POP             DS
                  MOV             SP,BP
                  POP             BP
                  RET
MENU              ENDP

;                 UBICA CURSOR

CURSOR            PROC            NEAR
                  MOV             BH,00
                  MOV             AH,02
                  INT             10H
                  RET
CURSOR            ENDP

;                 ESCRIBE CARACTER

ESCRIBE           PROC            NEAR
                  MOV             AH,09
                  MOV             BH,00
                  MOV             BL,05FH
                  MOV             CX,01
                  INT             10H
                  RET
ESCRIBE           ENDP

;
CODE              ENDS
                  END
```

```
code    segment     public
        assume      cs:code
        public      menu1

menu1   proc        near

        push        bp
        mov         bp,sp
        push        ds

        mov         dx,1636h
        call        cur
        mov         al,'F'
        call        esc
        mov         dx,1637h
        call        cur
        mov         al,'6'
        call        esc
        mov         dx,1638h
        call        cur
        mov         al,'-'
        call        esc
        mov         dx,1639h
        call        cur
        mov         al,'B'
        call        esc
        mov         dx,163ah
        call        cur
        mov         al,'U'
        call        esc
        mov         dx,163bh
        call        cur
        mov         al,'S'
        call        esc
        mov         dx,163ch
        call        cur
        mov         al,'C'
        call        esc
        mov         dx,163dh
        call        cur
        mov         al,'A'
        call        esc
        mov         dx,163eh
        call        cur
        mov         al,'R'
        call        esc

        mov         dx,1641h
        call        cur
        mov         al,'F'
        call        esc
        mov         dx,1642h
```

```
call        cur
mov         al,'7'
call        esc
mov         dx,1643h
call        cur
mov         al,'-'
call        esc
mov         dx,1644h
call        cur
mov         al,'R'
call        esc
mov         dx,1645h
call        cur
mov         al,'E'
call        esc
mov         dx,1646h
call        cur
mov         al,'E'
call        esc
mov         dx,1647h
call        cur
mov         al,'M'
call        esc
mov         dx,1648h
call        cur
mov         al,'P'
call        esc
mov         dx,1649h
call        cur
mov         al,'L'
call        esc
mov         dx,164ah
call        cur
mov         al,'A'
call        esc
mov         dx,164bh
call        cur
mov         al,'Z'
call        esc
mov         dx,164ch
call        cur
mov         al,'A'
call        esc
mov         dx,164dh
call        cur
mov         al,'R'
call        esc
mov         dx,1704h
call        cur
mov         al,'F'
call        esc
mov         dx,1705h
call        cur
mov         al,'8'
call        esc
mov         dx,1706h
mov         al,'-'
```

```
call        esc
mov         dx,1707h
call        cur
mov         al,'M'
call        esc
mov         dx,1708h
call        cur
mov         al,'O'
call        esc
mov         dx,1709h
call        cur
mov         al,'S'
call        esc
mov         dx,170ah
call        cur
mov         al,'T'
call        esc
mov         dx,170bh
call        cur
mov         al,'R'
call        esc
mov         dx,170ch
call        cur
mov         al,'A'
call        esc
mov         dx,170dh
call        cur
mov         al,'R'
call        esc

mov         dx,171bh
call        cur
mov         al,'E'
call        esc
mov         dx,171ch
call        cur
mov         al,'S'
call        esc
mov         dx,171dh
call        cur
mov         al,'C'
call        esc
mov         dx,171eh
call        cur
mov         al,'-'
call        esc
mov         dx,171fh
call        cur
mov         al,'S'
call        esc
mov         dx,1720h
call        cur
mov         al,'A'
mov         dx,1721h
call        cur
mov         al,'L'
call        esc
mov         dx,1722h
```

```
          call          cur
          mov           al,'I'
          call          esc
          mov           dx,1723h
          call          cur
          mov           al,'R'
          call          esc

          mov           dx,0000
          call          cur




          pop           ds
          mov           sp,bp
          pop           bp
          ret
menu1     endp

;         colocar cursor
cur       proc          near
          mov           bh,00
          mov           ah,02
          int           10h
          ret
cur       endp

;         escribe letra

esc       proc          near
          mov           ah,09
          mov           bh,00
          mov           bl,05fh
          mov           cx,01
          int           10h
          ret
esc       endp
;
code      ends
          end
```

```
;
;               SUBRUTINA  SUBIR.ASM
;

CODE            SEGMENT         PUBLIC
                ASSUME          CS:CODE
                PUBLIC          SUBIR

SUBIR           PROC            NEAR
                PUSH            BP
                MOV             BP,SP
                PUSH            DS

                MOV             AH,03
                MOV             BH,00
                INT             10H
                MOV             BH,DH
                CMP             BH,00
                JE              SALIDA

                MOV             AH,DH   ;FILA
                MOV             AL,DL   ;COLUMNA
                DEC             AH
                MOV             DH,AH
                MOV             DL,AL
                CALL            POSCUR

SALIDA:
                POP             DS
                MOV             SP,BP
                POP             BP
                RET
SUBIR           ENDP



POSCUR          PROC
                MOV             AH,02
                MOV             BH,00
                INT             10H
                RET
POSCUR          ENDP

CODE            ENDS
                END
```

```
;
;               SUBRUTINA BAJAR.ASM
;

                code            segment         public
                assume          cs:code
                public          bajar

                BAJAR     PROC      NEAR

                PUSH            BP
                MOV             BP,SP
                PUSH            DS

                MOV             AH,03
                MOV             BH,00
                INT             10H
                MOV             BH,DH
                CMP             BH,014H
                JE              SALIDA

                MOV             AH,DH   ;FILA
                MOV             AL,DL   ;COLUMNA
                INC             AH
                MOV             DH,AH
                MOV             DL,AL
                CALL            POSCUR

SALIDA:
                POP             DS
                MOV             SP,BP
                POP             BP
                RET
                BAJAR   ENDP

                POSCUR    PROC
                MOV       AH,02
                MOV       BH,00
                INT       10H
                RET
                POSCUR    ENDP

CODE            ENDS
                END
```

```
CODE      SEGMENT   PUBLIC
          ASSUME    CS:CODE
          PUBLIC    DERECHA

DERECHA   PROC      NEAR
                PUSH            BP
                MOV             BP,SP
                PUSH            DS

                MOV             AH,03
                MOV             BH,00
                INT             10H
                MOV             BL,DL
                CMP             BL,04FH
                JE              SALIDA

                MOV             AH,DH   ;FILA
                MOV             AL,DL   ;COLUMNA
                INC             AL
                MOV             DH,AH
                MOV             DL,AL
                CALL            POSCUR

SALIDA:
                POP             DS
                MOV             SP,BP
                POP             BP
                RET
DERECHA ENDP

POSCUR    PROC
          MOV       AH,02
          MOV       BH,00
          INT       10H
          RET
POSCUR    ENDP

CODE      ENDS
          END
```

```
;
;               SUBRUTINA  PAGE1
;


CODE            SEGMENT         PUBLIC
                ASSUME          CS:CODE
                PUBLIC          PAGE1

PAGE1           PROC            NEAR
                PUSH            BP
                MOV             BP,SP
                PUSH            DS

                MOV             AH,03
                MOV             BH,00
                INT             10H
                MOV             BH,DH
                MOV             BL,DL
                CMP             BH,014H
                JNE             SALIDA
                CALL            NUEVA
                CALL            POSCUR
SALIDA:
                POP             DS
                MOV             SP,BP
                POP             BP
                RET
PAGE1           ENDP

NUEVA           PROC            NEAR
                CALL            POSCUR
                MOV             AH,09
                MOV             AL,00
                MOV             BH,00
                MOV             BL,07
                MOV             CX,1600
                INT             10H
                RET
NUEVA           ENDP

POSCUR          PROC
                MOV             AH,02
                MOV             BH,00
                MOV             DX,0000
                INT             10H
                RET
POSCUR          ENDP

CODE            ENDS
                END
```

```
CODE        SEGMENT     PUBLIC
            ASSUME      CS:CODE
            PUBLIC      EXPLORE

            OFASCII     EQU     4¡BP¿     ; OFFSET DIREC TECLA
            SBASCII     EQU     6¡BP¿     ; BASE D DIRCC DEL SEGMENTO
            OFSCAN      EQU     8¡BP¿
            SBSCAN      EQU     10¡BP¿

EXPLORE     PROC        NEAR                ;INICIALIZACION
            PUSH        BP
            MOV         BP,SP
            PUSH        DS
            PUSH        DI

            CALL        RASTREO
            CMP         AL,00
            JZ          FIN

            CALL        FUNCION
            CMP         AL,00
            JNZ         LETRA       ; NO ES TECLA DE FUNCION
            MOV         AH,00
            MOV         DS,SBSCAN
            MOV         DI,OFSCAN
            MOV         DS:¡DI¿,AX
            CALL        FUNCION
            MOV         AH,00
            MOV         DS,SBASCII
            MOV         DI,OFASCII
            MOV         DS:¡DI¿,AX
            JMP         FIN
LETRA:
            CMP         AL,1BH      ; TECLA ESCAPE
            JNE         FIN
            MOV         AH,00       ; MANDA CODIGO 30
            MOV         DS,SBSCAN
            MOV         DI,OFSCAN
            MOV         DS:¡DI¿,AX
            JMP         FIN
FIN:
            POP         DI
            POP         DS
            MOV         SP,BP
            POP         BP
            RET         8
EXPLORE     ENDP

RASTREO     PROC        NEAR
            MOV         AH,0BH
            INT         21H
            RET
RASTREO     ENDP

FUNCION     PROC        NEAR
            MOV         AH,08H
            INT         21H
            RET




FUNCION     ENDP
CODE        ENDS
```

```
;
;       SUBRUTINA LIMPIAR
;

code    segment     public
        assume      cs:code
        public      limpiar

limpiar proc        near
        push        bp
        mov         bp,sp
        push        ds

        mov         ah,02
        mov         bh,00
        mov         dx,0000
        int         10h

        mov         ax,0900h
        mov         bh,00
        mov         bl,07
        mov         cx,2000
        int         10h

        pop         ds
        mov         sp,bp
        pop         bp
        ret
limpiar endp
code    ends
        end
```

```
PROGRAM COMPILADOR;

{$L LIMPIAR.OBJ}
{$L COMUNICA.OBJ}

 type
   comando = record
   coma :packed array[1..20] of char;
   linea : integer;
   end;
var
    compiler                :file of integer;
    Trans2                  :array [1..6] of integer;
    Trans1                  :array [1..5] of integer;
    Trans                   :array[1..8,1..3] of integer;
    archivo,puerto,modo     :integer;
    car,resulta,final       :integer;
    tcar,tresult,codigo,opc :integer;
    comp                    :array[1..100,1..7] of integer;
    archi                   :file of comando;
    sentence                :file of comando;
    sentences               :text;
    lineas                  :packed array[1..17,1..69] of char;
    sentencia               :array [1..100] of comando;
    dup1                    :packed array [1..4,1..10] of char;
    ch                      :char;
    a,j,i,variable,h,p,k,s,l :integer;
    fijo,US                 :integer;
     com                    :packed array [1..20] of char;
     dup                    :packed array [1..4,1..11] of char;
     dup2                   :packed array [1..2,1..12] of char;
     dup3                   :packed array [0..9] of char;
     dup4                   :packed array [1..15] of char;
     transmite              :packed array [1..20] of char;
     proc                   :array [1..20] of integer;
     sentencename           :packed array [1..20] of char;
     archiname              :packed array [1..20] of char;
     existe                 :boolean;

procedure comunica(puerto,modo:integer;var okey,car,resulta:integer);external;
procedure transmita(var tcar,tresult:integer);
       var
 tran:integer;
 good:integer;
  begin
  tran:=1;
  puerto:=1;
  comunica(puerto,tran,good,tcar,tresult);
  end;
procedure convertir;
  begin
  if h<10 then
  begin
  lineas[1,29]:=chr(ord(ch) + h);
```

```
  writeln(sentences,lineas[1]);
  writeln(sentences,lineas[17]);
  end
        else
        begin
        if h<100 then
   begin
   k:=h div 10;
   lineas[1,28]:=chr(ord(ch) + k);
   l:= h - 10*k;
   lineas[1,29]:=chr(ord(ch) + l);
   writeln(sentences,lineas[1]);
   writeln(sentences,lineas[17]);
   end
   else
   begin
   lineas[1,27]:='1';
   lineas[1,28]:='0';
   lineas[1,29]:='0';
   writeln(sentences,lineas[1]);
   writeln(sentences,lineas[17]);
   end;
  end;
  end;
procedure extension;
  begin
  p:=0;
  for j:=1 to i do
  begin
  if com[j]='.' then p:=j+1;
  end;
  h:=0;
  if com[p]='s' then
     begin
     h:=p+1 ;
     if com[h]='c' then
       begin
       h:=h+1;
       if com[h]='p' then h:=1 else h:=0;
       end
       else h:=0;
     end;
  if (h=0) and (p=0) then
  begin
  writeln;
  writeln('INGRESE LA EXTENSION  .scp');
  writeln;
  writeln('PRESIONE <ENTER> PARA CONTINUAR...');
  readln;
  end;
  end;
procedure limpiar;external;
{-------------------------------------------------------------------------}
procedure cargar;
```

```
var
   util:comando;
begin
     limpiar;
     for i:=1 to 100 do
     begin
 with sentencia[i] do
 begin
 for j:=1 to 20 do
 coma[j]:=' ';
 linea:=i;
 end;
     end;
     for i:=1 to 20 do
     archiname[i]:=com[i];
     assign(archi,archiname);
     {$I-}
     reset(archi);
     {$I+}
     existe:=(IOresult=0);
     if existe then
     begin
     i:=0;
     while not eof(archi) do
  begin
             i:=i+1;
             read(archi,util);
       sentencia[i]:=util;
             writeln;
  end;
  close(archi);
  final:=i;
     writeln('Archivo ',archiname,' copiado ');
     writeln;
     write('PRESIONE <ENTER> PARA CONTINUAR...');
     readln;
     limpiar;
     end
     else
       begin
 writeln('ARCHIVO NO ENCONTRADO');
 writeln;
 writeln('PRESIONE <ENTER> PARA CONTINUAR...');
 readln;
 end;
end;
{------------------------------------------------------------------------}
   begin
   limpiar;
   opc:=0;
   while opc<>4 do
   BEGIN
   limpiar;
     WRITELN;
```

```
    WRITELN;
    WRITELN('                  SISTEMAS DE CONTROL DE PROCESOS         ver 1.0');
    WRITELN;
    WRITELN;
    WRITELN('              ESCUELA SUPERIOR POLITECNICA DEL LITORAL');
    WRITELN;
    WRITELN;
    WRITELN('                     SELECCIONE UNA OPCION  :');
    WRITELN;
    WRITELN;
    WRITELN;
    WRITELN;
    WRITELN(' 1._ EDITAR UN PROGRAMA');
    WRITELN;
    WRITELN(' 2._ COMPILAR UN PROGRAMA');
    WRITELN;
    WRITELN(' 3._ CARGAR UN PROGRAMA AL CONTROLADOR');
    WRITELN;
    WRITELN(' 4._ RETORNO A DOS');
    WRITELN;
    WRITELN;
    WRITELN;
    WRITE ('                      INGRESE UNA OPCION :  ');
    READLN(opc);
    CASE opc OF

2:  BEGIN
    repeat
    limpiar;
    write('INGRESE EL NOMBRE DEL ARCHIVO A COMPILAR :  ');
    for i:=1 to 20 do
    begin
    com[i]:= ' ';
    end;
    i:=0;
    repeat
    i:=i+1;
    read(ch);
    com[i]:=ch;
    until eoln;
    readln;
    p:=0;
    for j:=1 to i do
    begin
    if com[j]='.' then p:=j+1;
    end;
    h:=0;
    if com[p]='s' then
    begin
    h:=p+1 ;
    if com[h]='c' then
      begin
      h:=h+1;
      if com[h]='p' then h:=1 else h:=0;
```

```
      end
      else h:=0;
    end;
    if (h=0) and (p=0) then
    begin
    writeln;
     writeln('INGRESE LA EXTENSION  .scp');
     writeln;
     writeln('PRESIONE <ENTER> PARA CONTINUAR...');
readln;
    end; cargar;
    until (h<>0) and (p<>0) and (existe);
    for i:=1 to 17 do
    begin
    for j:=1 to 69 do lineas[i,j]:=' ';
end;
    dup[1]:='prender(a);';
    dup[2]:='prender(b);';
    dup[3]:='prender(c);';
    dup[4]:='prender(v);';
    dup1[1]:='apagar(a);';
    dup1[2]:='apagar(b);';
    dup1[3]:='apagar(c);';
    dup1[4]:='apagar(v);';
    dup2[1]:='esperar(t=);';
    dup2[2]:='esperar(T>);';
    dup4   :='mantener(T,t=);';
    lineas[1]:='CON RESPECTO A LA LINEA :                                    ';
    lineas[2]:='EXISTEN    ERROR(ES) DE SINTAXIS                             ';
    lineas[3]:='EL COMANDO DEBE SER :                                        ';
    lineas[4]:='ESCRIBA EL ARGUMENTO ENTRE PARENTESIS ( )                    ';
    lineas[5]:='DIGITE BIEN EL ARGUMENTO, DEBE SER:  A ,B, C  o  V           ';
    lineas[6]:='EL ARGUMENTO DE MANTENER DEBE SER  T,t                       ';
    lineas[7]:='DEBE IR ; COMO DELIMITADOR DE INSTRUCCION                    ';
    lineas[8]:='EXISTEN ERRORES SEVEROS, DIGITE BIEN LA SENTENCIA            ';
    lineas[9]:='LA SENTENCIA DEBE SER  PRENDER, APAGAR, ESPERAR O MANTENER   ';
    lineas[10]:='Y NO DEBE SER :                                             ';
    lineas[11]:='SE HAN DIGITADO :      SENTENCIAS DE CONTROL                ';
    lineas[12]:='LA TEMPERATURA DEBE CONSTAR DE 3 DIGITOS DEL 0-9            ';
    lineas[14]:='DEBE EXISTIR ENTRE EL SIMBOLO DE TEMPERATURA Y SU VALOR EL OPERADOR >';
    lineas[15]:='DEBE EXISTIR ENTRE EL SIMBOLO DE TIEMPO Y SU VALOR EL OPERADOR  =    ';
    lineas[16]:='EL TIEMPO DEBE CONSTAR DE 2 DIGITOS DEL 0-9                 ';
  writeln;
  readln;
  writeln('COMPILANDO ARCHIVO ',com);
  ch:='0';
  for i:= 0 to 9 do
  dup3[i]:=chr(ord(ch) + i);
     j:=final;
  l:=0;
  repeat
  l:=l+1;
  until com[l]='.';
  l:=l+1;
```

```
com[l]:='l'; archiname[l]:='a';
l:=l+1;
com[l]:='s'; archiname[l]:='s';
l:=l+1;
com[l]:='t'; archiname[l]:='c';
assign(sentences,com);
rewrite(sentences);
s:=j; CH:='0';
writeln(sentences,lineas[17]);
writeln(sentences,lineas[17]);
if s<10 then
 begin
 lineas[11,21]:=chr(ord(ch) + s);
 writeln(sentences,lineas[11]);
 end
  else
  begin
  if s<100 then
  begin
  k:=s div 10;
  lineas[11,20]:=chr(ord(ch) + k);
  l:= s - 10*k ;
  lineas[11,21]:=chr(ord(ch) + l);
  writeln(sentences,lineas[11]);
  end
  else
  begin
  lineas[11,19]:='1';
  lineas[11,20]:='0';
  lineas[11,21]:='0';
  end;
  end;
  for i:=1 to s do
  begin
   with sentencia[i] do
    begin
    if (coma[2]>'a') then
  begin
   if (coma[2]>'e') then
      begin
 if coma[2]>'m' then  proc[i]:=1
  else  proc[i]:=4;
      end
      else proc[i]:=3;
      end
 else proc[i]:=2;
   end;
 end;
 for i:=1 to 100 do
 begin
 for j:=1 to 7 do comp[i,j]:=0;
 end;
for a:=1 to s do
begin
```

```
with sentencia[a] do
 begin
if proc[a]=1 then
   begin
    p:=0; h:=a; comp[a,6]:=1; comp[a,7]:=1;
    for j:=2 to 8 do
    begin
    if coma[j]<>dup[1,j-1] then p:=p+1;
    end;
    if p=0 then comp[a,1]:=1;
    writeln(sentences,lineas[17]);
    convertir;
    writeln(sentences,coma);
    writeln(sentences,lineas[17]);
  if p<=4 then
     begin
lineas[2,9]:=chr(ord(ch) + p );
writeln(sentences,lineas[2]);
if p<>0 then
begin
for i:=1 to 7  do lineas[3,i+22]:= dup[1,i];
writeln(sentences,lineas[3]);
end;
l:=3;
repeat
l:=l+1;
until (coma[l]=dup[1,8]) or (l=20);
if l=20 then writeln(sentences,lineas[4])
else comp[a,2]:=1;
k:=0;
for i:=1 to 4 do
begin
l:=6;
repeat
l:=l+1;
until (coma[l]=dup[i,9]) or (l=20);
if coma[l]=dup[i,9] then k:=1;
end;
if  k<>1 then  writeln(sentences,lineas[5])
else comp[a,3]:=1;
l:=3;
repeat
l:=l+1;
until (coma[l]=dup[1,10]) or (l=20);
if l=20 then writeln(sentences,lineas[4])
else comp[a,4]:=1;
l:=1;
repeat
l:=l+1;
until (coma[l]=dup[1,11]) or (l=20);
if l=20 then writeln(sentences,lineas[7])
else comp[a,5]:=1;
end
else
```

```
 begin
 writeln(sentences,lineas[8]);
 writeln(sentences,lineas[9]);
 end;
 end;
if proc[a]=2 then
  begin
     p:=0 ; h:=a;  comp[a,6]:=1; comp[a,7]:=1;
     for j:=2 to 7 do
     begin
     if coma[j]<>dupl[1,j-1] then p:= p+1;
     end;
     if p=0 then comp[a,1]:=1;
     writeln(sentences,lineas[17]);
     writeln(sentences,lineas[17]);
     convertir;
     writeln(sentences,coma);
     writeln(sentences,lineas[17]);
      if p<=3 then
       begin
       lineas[2,9]:=chr(ord(ch) + p);
       writeln(sentences,lineas[2]);
       if p<>0 then
       begin
       for i:=1 to 6 do lineas[3,i+22]:=dupl[1,i];
       writeln(sentences,lineas[3]);
       end;
       l:=3;
       repeat
       l:=l+1;
       until (coma[l]=dupl[1,7]) or (l=20);
       if l=20 then writeln(sentences,lineas[4])
       else comp[a,2]:=1;
       k:=0;
       for i:=1 to 4 do
       begin
       l:=6;
       repeat
       l:=l+1;
       until (coma[l]=dupl[i,8]) or (l=20);
       if (coma[l]=dupl[i,8]) then k:=1;
       end;
       if k<>1 then writeln(sentences,lineas[5])
       else comp[a,3]:=1;
       l:=3;
       repeat
       l:=l+1;
       until (coma[l]=dupl[1,9]) or (l=20);
       if l=20 then writeln(sentences,lineas[4])
       else comp[a,4]:=1;
       l:=1;
       repeat
       l:=l+1;
       until (coma[l]=dupl[1,10]) or (l= 20) ;
```

```
  if l=20 then writeln(sentences,lineas[7])
  else comp[a,5]:=1;
  end
  else
  begin
  writeln(sentences,lineas[8]);
  writeln(sentences,lineas[9]);
  end;
  end;
    if proc[a]=4 then
begin
p:=0; h:=a;
for j:=2 to 9 do
begin
if coma[j]<>dup4[j-1] then p:=p+1;
end;
if p=0 then comp[a,1]:=1;
writeln(sentences,lineas[17]);
writeln(sentences,lineas[17]);
convertir;
writeln(sentences,coma);
writeln(sentences,lineas[17]);
if p<=4 then
 begin
 lineas[2,9]:=chr(ord(ch) + p);
 writeln(sentences,lineas[2]);
 if p<>0 then
 begin
 for i:=1 to 8 do lineas[3,i+22]:=dup4[i];
 writeln(sentences,lineas[3]);
 end;
 l:=3;
 repeat
 l:=l+1;
 until (coma[l]=dup4[9]) or (l=21);
 if l=21 then writeln(sentences,lineas[4])
  else comp[a,2]:=1 ;
 l:=3;
 repeat
 l:=l+1;
 until (coma[l]=dup4[10]) or (l=20);
 k:=3;
 repeat
 k:=k+1;
 until (coma[k]=dup4[11]) or (k=20);
 i:=5;
 repeat
 i:=i+1;
 until (coma[i]=dup4[12]) or (i=20);
 if ((l=20) or (k=20) or (i=20) or (i<k)) then
 writeln(sentences,lineas[6])
 else comp[a,3]:=1;
 l:=3;
 repeat
```

```
      l:=l+1;
      until (coma[l]=dup4[13]) or (l=20);
      if l=20 then writeln(sentences,lineas[15])
       else comp[a,4]:=1 ;
      if coma[l]=dup4[13] then k:=l+1
     else k:=15;
      l:=k+1; p:=0;
      for i:=k to l do
      begin
      for j:=0 to 9 do
      begin
      if coma[i]=dup3[j] then p:=p+1;
      end;
      end;
      if p<>2 then writeln(sentences,lineas[16])
       else comp[a,5]:=1;
      l:=3;
      repeat
      l:=l+1;
      until (coma[l]=dup4[14]) or (l=20);
      if l=21 then writeln(sentences,lineas[4])
       else comp[a,6]:=1 ;
      l:=3;
      repeat
      l:=l+1;
      until (coma[l]=dup4[15]) or (l=20);
      if l=20 then writeln(sentences,lineas[7])
       else comp[a,7]:=1 ;
      end
      else
      begin
      writeln(sentences,lineas[8]);
      writeln(sentences,lineas[9]);
      end;
      end;
if proc[a]=3 then
   begin
    p:=0; h:=a;
    for j:=2 to 8 do
    begin
    if coma[j]<>dup2[1,j-1] then p:=p+1;
    end;
    if p=0 then comp[a,1]:=1;
    writeln(sentences,lineas[17]);
    writeln(sentences,lineas[17]);
    convertir;
    writeln(sentences,coma);
    writeln(sentences,lineas[17]);
    if p<=4 then
      begin
      lineas[2,9]:=chr(ord(ch) + p);
      writeln(sentences,lineas[2]);
      if p<>0 then
      begin
```

```
        for i:=1 to 7 do lineas[3,i+22]:=dup2[1,i];
        writeln(sentences,lineas[3]);
        end;
        l:=3;
        repeat
        l:=l+1;
        until (coma[l]=dup2[1,8]) or (l=20);
        if l=20 then writeln(sentences,lineas[4])
        else comp[a,2]:=1;
        j:=10;
        if (coma[j]='t') or (coma[j]='T') then comp[a,3]:=1 else
        begin
lineas[13]:='EL ARGUMENTO DE ESPERAR SOLO PUEDE SER TIEMPO (t) o TEMPERATURA (T)  ';
writeln(sentences,lineas[13]);
        l:=0;
        repeat
        l:=l+1;
        until (coma[l]=')') OR (l=20);
        if l=20 then
        begin
        writeln(sentences,lineas[4]);
        end;
        l:=1;
        repeat
        l:=l+1;
        until (coma[l]=dup2[1,12]) or (l=20);
        if l=20 then
        begin
        writeln(sentences,lineas[7]);
        end;
        end;
        j:=10;
        if (coma[j]='T') then
        begin
        j:=j+1;
        if coma[j]<>')' then
        begin
        writeln(sentences,lineas[14]);
        end
        else comp[a,4]:=1;
        p:=0;
        for i:=12 to 14 do
        begin
        for j:=0 to 9 do
        begin
        if coma[i]=dup3[j] then p:=p+1;
        end;
        end;
        if p<>3 then writeln(sentences,lineas[12])
        else comp[a,5]:=1;
        l:=3;
        repeat
        l:=l+1;
        until (coma[l]=dup2[1,11]) or(l=20);
```

```
    if l=20 then writeln(sentences,lineas[4])
    else comp[a,6]:=1;
    l:=1;
    repeat
    l:=l+1;
    until (coma[l]=dup2[2,12]) or (l=20);
    if l=20 then  writeln(sentences,lineas[7])
    else comp[a,7]:=1;
    end
    else
    begin
    if coma[j]=dup2[1,9] then
    begin
    j:=j+1;
    if coma[j]<>dup2[1,10] then
    begin
    writeln(sentences,lineas[15]);
    end
    else comp[a,4]:=1;
    p:=0;
    for i:=12 to 13 do
    begin
    for j:=0 to 9 do
    begin
    if coma[i]=dup3[j] then p:=p+1;
    end;
    end;
    if p<>2 then writeln(sentences,lineas[16])
    else comp[a,5]:=1;
    l:=3;
    repeat
    l:=l+1;
    until (coma[l]=dup2[1,11]) or (l=20);
    if l=20 then writeln(sentences,lineas[4])
    else comp[a,6]:=1;
    l:=1;
    repeat
    l:=l+1;
    until (coma[l]=dup2[1,12]) or (l=20);
    if l=20 then writeln(sentences,lineas[7])
    else comp[a,7]:=1;
          end;
    end;
    end
    else
    begin
    writeln(sentences,lineas[8]);
    writeln(sentences,lineas[9]);
    end;
    end;
    end;
    end;
    close(sentences);
limpiar;
```

```
archivo:=1;
for a:=1 to s do
begin
for j:=1 to 7 do
begin
if comp[a,j]<>1 then archivo:=0;
end;
end;
if archivo<>1 then
begin
writeln('                        COMPILACION NO EXITOSA');
writeln;
write('PRESIONE <ENTER> PARA CONTINUAR...');
readln;
end;
if archivo=1 then
begin
writeln('                        COMPILACION EXITOSA');
writeln;
write('PRESIONE <ENTER> PARA CONTINUAR...');
writeln;
readln;
assign(compiler,archiname);
rewrite(compiler);
for i:=1 to 8 do
begin
Trans[i,1]:=205;
Trans[i,3]:=05;
end; i:=2;
Trans[1,i]:=208; Trans[2,i]:=224; Trans[3,i]:=192; Trans[4,i]:=240;
Trans[5,i]:=117; Trans[6,i]:=152; Trans[7,i]:=101; Trans[8,i]:=133;
for a:=1 to s do
begin
with sentencia[a] do
begin
if proc[a]=1 then
begin
for i:=1 to 4 do
begin
if coma[10]=dup[i,9] then k:=i;
end;
for i:=1 to 3 do write(compiler,Trans[k,i]);
end;
if proc[a]=2 then
begin
for i:=5 to 8 do
begin
if coma[9]=dupi[i-4,8] then k:=i;
end;
for i:=1 to 3 do write(compiler,Trans[k,i]);
end;
if proc[a]=3 then
begin
if coma[10]='t' then
```

```
    begin
    Trans1[1]:=62; Trans1[3]:=205; Trans1[4]:=21; Trans1[5]:=05;
    for j:=0 to 9 do
    begin
    if coma[12]=dup3[j] then k:=10*j;
    end;
    for j:=0 to 9 do
    begin
    if coma[13]=dup3[j] then k:=k+j;
    end;
    Trans1[2]:=k;
    for i:=1 to 5 do write(compiler,Trans1[i]);
    end;
    if coma[10]='T' then
    begin
    Trans2[1]:=6; Trans2[3]:=197; Trans2[4]:=205; Trans2[5]:=21; Trans2[6]:=6;
    for i:=0 to 9 do
    begin
    if coma[12]=dup3[i] then k:=100*i;
    end;
    for i:=0 to 9 do
    begin
    if coma[13]=dup3[i] then k:=k+10*i;
    end;
    for i:=0 to 9 do
    begin
    if coma[14]=dup3[i] then k:=k+i;
    end;
    Trans2[2]:=k;
    for i:=1 to 6 do write(compiler,Trans2[i]);
    end;
    end;
    if proc[a]=4 then
    begin
    Trans1[1]:=46; Trans1[3]:=205; Trans1[4]:=53; Trans1[5]:=6;
    for i:=0 to 9 do
    begin
    if coma[final]=dup3[i] then k:=10*i;
    end;
    for i:=0 to 9 do
    begin
    if coma[final+1]=dup3[i] then k:=k+i;
    end;
    Trans1[2]:= k;
    for i:=1 to 5 do write(compiler,Trans1[i]);
    end;
    end;
    end;
    close(compiler);
    end;
    end;

 3:    begin
for i:=0 to 20 do transmite[i]:=' ';
```

```
ch:=' ';
repeat
limpiar;
writeln('EL ARCHIVO A TRANSMITIR AL CONTROLADOR DEBE LLEVAR LA EXTENSION .ASC');
writeln;
write('INGRESE EL NOMBRE DEL ARCHIVO A TANSMITIR :  ');
i:=0;
repeat
i:=i+1;
read(ch);
transmite[i]:=ch;
until eoln;
readln;
assign(compiler,transmite);
{$I-}
reset(compiler);
{$I+}
existe:=(IOresult=0);
if existe then
begin
finfile:=filesize(compiler);
 for i:=0 to (finfile-1) do
 begin
 read(compiler,codigo);
 transmita(codigo,resulta);
writeln('car enviado ',codigo);
 end;
codigo:=42;
transmita(codigo,resulta);
close(compiler);
writeln;
writeln;
writeln('                          TRANSMISION EXITOSA');
writeln;
writeln('PRESIONE <ENTER> PARA CONTINUAR...');
readln;
end
else
begin
writeln;
writeln('ARCHIVO NO ENCONTRADO');
writeln;
writeln('PRESIONE <ENTER> PARA CONTINUAR...');
readln;
end;
until existe;
end;

 4:
end;
end;
limpiar;
     end.
```

```
;
;       SUBRUTINA LIMPIAR.ASM
;

code    segment     public
        assume      cs:code
        public      limpiar

limpiar proc        near
        push        bp
        mov         bp,sp
        push        ds

        mov         ah,02
        mov         bh,00
        mov         dx,0000
        int         10h

        mov         ax,0900h
        mov         bh,00
        mov         bl,07
        mov         cx,2000
        int         10h

        pop         ds
        mov         sp,bp
        pop         bp
        ret
limpiar endp
code    ends
        end
```

```
;-------------------------------------------------------
; SUBRUTINA EXTERNA -- COMUNICA.ASM-- PARA TRANSMISION
;       RECEPCION DE 1 CARACTER POR EL PUERTO SERIAL
;   ESPECIFICADO DESDE PASCAL, RETORNA EL CODIGO
;     DE EXITO O FRACASO DE LA COMUNICACION
;-------------------------------------------------------

CODE            SEGMENT         PUBLIC
                ASSUME          CS:CODE
                PUBLIC          COMUNICA

            OBTENGO LA DIRECCION DE LAS VARIABLES

                OFRESULT        EQU 4[BP]
                SBRESULT        EQU 6[BP]
                OFCAR           EQU 8[BP]
                SBCAR           EQU 10[BP]
                OFOKEY          EQU 12[BP]
                SBOKEY          EQU 14[BP]
                MODO            EQU 16[BP]
                PUERTO          EQU 18[BP]

COMUNICA        PROC            NEAR
                PUSH            BP
                MOV             BP,SP
                PUSH            DS
                PUSH            DI

           INICIO DE LA SUBRUTINA PRINCIPAL


                MOV             AX,MODO
                CMP             AX,01
                JNZ             RECIBE

                MOV             DS,SBCAR
                MOV             DI,OFCAR
                MOV             AX,DS:[DI]
                MOV             AH,01
                MOV             DX,PUERTO
                INT             14H
                MOV             DS,SBRESULT
                MOV             DI,OFRESULT
                MOV             DS:[DI],AX
                MOV             BX,AX
                AND             BX,8000H
                MOV             DS,SBOKEY
                MOV             DI,OFOKEY
                [illegible]     [illegible]BX

RECIBE:         MOV             AX,MODO
                CMP             AX,02
                JNZ             STATUS
                MOV             DX,PUERTO
                MOV             AH,02
                INT             14H
                SUB             BX,BX
                MOV             DX,AX
                MOV             BL,AL
```

```
                MOV             DS,SBCAR
                MOV             DI,OFCAR
                MOV             DS:[DI],BX
                MOV             DS,SBRESULT
                MOV             DI,OFRESULT
                MOV             DS:[DI],DX
                MOV             CX,DX
                AND             CX,8000H
                MOV             DS,SBOKEY
                MOV             DI,OFOKEY
                MOV             DS:[DI],CX
                JMP             TERMINAR

STATUS:         MOV             DX,PUERTO
                MOV             AH,03
                INT             14H
                SUB             CX,CX
                SUB             DX,DX
                MOV             DL,AL
                MOV             CL,AH
                MOV             DS,SBRESULT
                MOV             DI,OFRESULT
                MOV             DS:[DI],DX

                MOV             DS,SBOKEY
                MOV             DI,OFOKEY
                MOV             DS:[DI],CX

TERMINAR:

                POP             DI
                POP             DS
                MOV             SP,BP
                POP             BP
                RET             12
COMUNICA        ENDP
CODE            ENDS
                END
```

PROGRAMA DE CONFIGURACION DEL PUERTO SERIAL : 2.

Este Programa nos permite realizar la configuracion del puerto serial # 2, el cual nos servira para realizar la comunicación planta - control.

* *

```
#include <bios.h>

main()

{
   _bios_serialcom(_COM_INIT,1,_COM_CHR8|_COM_STOP2|_COM_NOPARITY|_COM_1200);

   printf("\n    CONFIGURACION DEL PUERTO # 2 IMPLEMENTADA.   ");

}
```

* *

PROGRAMA DE COMUNICACION : 1.

Este programa,nos permite realizar el envio del Archivo donde esta el programa de control de la planta.Este programa esta incluido dentro del programa principal del EDITOR-COMPILADOR El envio se lo hara atravez del puerto serial : 2

* * * * * * * *

```
program  ENVIARCHIVO(input,output);
var
   puerto,valor:integer;
   modo,car,result:integer;
   nump:integer;
   okey:integer;
   i:integer;
   archivo: file of integer;
   codigo:integer;
   m:integer;
   finfile:integer;
   tcar,tresult:integer;

procedure comunica(puerto,modo:integer; var okey,car,result
                    :integer);external 'comunica.com';
procedure transmita(var tcar,tresult:integer);
var
     tran:integer;
     good:integer;
     begin
         tran:=1; { 1 para transmitir 1 caracyter }
         comunica(puerto,tran,good,tcar,tresult);
     end;
begin
  assign(archivo,'datos.asc');
  reset(archivo);
  puerto:=1;
  finfile:=filesize(archivo);
  for m:=0 to (finfile-1) do
    begin
         read(archivo,codigo);
         transmita(codigo,result);
         writeln('car enviado ',codigo);
    end;
 codigo:=42;
 transmita(codigo,result);
 close(archivo);
 writeln('bien');
end.
```

* *

```
SUBRUTINA EXTERNA -- COMUNICA.ASM-- PARA TRANSMISION
     RECEPCION DE 1 CARACTER POR EL PUERTO SERIAL
  ESPECIFICADO DESDE PASCAL, RETORNA EL CODIGO
    DE EXITO O FRACASO DE LA COMUNICACION .


    Esta subrutina es utilizada en el programa de Comunicacion
    # 1 el cual llama a esta subrutina en Ensamblador.

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *


CODE            SEGMENT         PUBLIC
                ASSUME          CS:CODE
                PUBLIC          COMUNICA

          OBTENGO LA DIRECCION DE LAS VARIABLES

                OFRESULT        EQU 4[BP]
                SBRESULT        EQU 6[BP]
                OFCAR           EQU 8[BP]
                SBCAR           EQU 10[BP]
                OFOKEY          EQU 12[BP]
                SBOKEY          EQU 14[BP]
                MODO            EQU 16[BP]
                PUERTO          EQU 18[BP]

COMUNICA        PROC            NEAR
                PUSH            BP
                MOV             BP,SP
                PUSH            DS
                PUSH            DI

          INICIO DE LA SUBRUTINA PRINCIPAL

                MOV             AX,MODO
                CMP             AX,01
                JNZ             RECIBE

                MOV             DS,SBCAR
                MOV             DI,OFCAR
                MOV             AX,DS:[DI]
                MOV             AH,01
                MOV             DX,PUERTO
                INT             14H
                MOV             DS,SBRESULT
                MOV             DI,OFRESULT
                MOV             DS:[DI],AX
                MOV             BX,AX
                AND             BX,8000H
                MOV             DS,SBOKEY
```

```
          MOV        DI,OFOKEY
          MOV        DS:[DI],BX
          JMP        TERMINAR

RECIBE:   MOV        AX,MODO
          CMP        AX,02
          JNZ        STATUS
          MOV        DX,PUERTO
          MOV        AH,02
          INT        14H
          SUB        BX,BX
          MOV        DX,AX
          MOV        BL,AL
          MOV        DS,SBCAR
          MOV        DI,OFCAR
          MOV        DS:[DI],BX
          MOV        DS,SBRESULT
          MOV        DI,OFRESULT
          MOV        DS:[DI],DX
          MOV        CX,DX
          AND        CX,8000H
          MOV        DS,SBOKEY
          MOV        DI,OFOKEY
          MOV        DS:[DI],CX
          JMP        TERMINAR


STATUS:   MOV        DX,PUERTO
          MOV        AH,03
          INT        14H
          SUB        CX,CX
          SUB        DX,DX
          MOV        DL,AL
          MOV        CL,AH
          MOV        DS,SBRESULT
          MOV        DI,OFRESULT
          MOV        DS:[DI],DX

          MOV        DS,SBOKEY
          MOV        DI,OFOKEY
          MOV        DS:[DI],CX

TERMINAR:

          POP        DI
          POP        DS
          MOV        SP,BP
          POP        BP
          RET        12
COMUNICA  ENDP
CODE      ENDS
          END
```

* *

P R O G R A M A D E C O M U N I C A C I O N : 2 .

Este programa nos permite hacer la recepcion de las informaciones
que llegan de la PLANTA ,Este programa se encuentra insertado
dentro del programa principal de la VISUALIZACION.
La comunicación se lo hara a traves del puerto serial : 2.

* *

```
Program RECEPTINFOR(input,ouput);

var
   car,atrib,pntalla:integer;
   tecla:integer;
   archivo: text;
   puerto,valor:integer;
   i:integer;
   plandta:longplan;
   aplandta:longplan;
   c1,c2,c3,cip,cfp:integer;

procedure reciba(puerto:integer; var car:integer);external
                                    'reciba.com';
procedure transmita(puerto:integer; var car:integer);external
                                     'transmi.com';
procedure verstatus(puerto:integer; var valor:integer);external
                                     'estatus.com';
procedure ingresardatos;
var
    c:integer;
    margen:integer;
    aster:integer;
     Begin
           aster:=42;
           if cfp=234 then
              begin
                transmita(puerto,aster);
                cfp:=0;
               end;
           verstatus(puerto,valor);
           if valor=1 then
            begin
                 reciba(puerto,cip);
                  if cip= 224 then
                    begin
                      reciba(puerto,c1);
                      reciba(puerto,c2);
                      reciba(puerto,c3);
                      reciba(puerto,cfp);
                      plandta[1]:=c1;
                      plandta[2]:=c2;
```

```
                   plandta[3]:=c3;
                   plandta[4]:=cfp;
                   if (plandta[4]=234) then
                       begin
                               for c:=1 to 3 do
                                 begin
                                   aplandta[c]:=plandta[c];
                                 end;
                         end;
              end;
          end;
     end;                        { fin del procedimiento }
procedure falla;
var
    aster:integer;
Begin
           cfp:=00;
           aster:=$ff;
           write('TECLA DE EMERGENCIA PULSADA');
           repeat
             verstatus(puerto,valor);
             reciba(puerto,cip);
             if cip=224 then
                  begin
                       reciba(puerto,c1);
                       reciba(puerto,c2);
                       reciba(puerto,c3);
                       reciba(puerto,cfp);
                       if cfp=234 then
                       transmita(puerto,aster);
                  end;
              end;
             until cfp=234;
    end;                        { fin del procedimiento }

{        P R O G R A M A   P R I N C I P A L      }

Begin
     puerto:=1;
     cfp:=234;
  repeat
           ingresardatos;
        if tecla= 3 then
           begin
             falla;
             tecla:=4;
           end;
   until tecla=4;
end.                        { FIN DEL PROGRAMA PRINCIPAL }

                            {    DE   COMUNICACION    }
```

* *

SUBRUTINA EXTERNA -- RECIBA.ASM-- PARA RECEPCION DE 1 CARACTER POR EL PUERTO SERIAL .

Esta subrutina la utilizamos para el programa de comunicacion # 2 ,hecho en Pascal que llama subrutinas en Ensamblador.

* *

```
SEGCODIGO       SEGMENT     PUBLIC      'CODE'
                ASSUME      CS:SEGCODIGO
                PUBLIC      RECIBA

                OFCAR       EQU 4[BP]
                SBCAR       EQU 6[BP]
                PUERTO      EQU 8[BP]

RECIBA          PROC        NEAR
                PUSH        BP
                MOV         BP,SP
                PUSH        DS
                PUSH        DI

                MOV         DX,PUERTO
                MOV         AH,02
                INT         14H
                SUB         BX,BX
                MOV         BL,AL
                MOV         DS,SBCAR
                MOV         DI,OFCAR
                MOV         DS:[DI],BX

                POP         DI
                POP         DS
                MOV         SP,BP
                POP         BP
                RET         6
 RECIBA         ENDP
SEGCODIGO       ENDS
                END
```

* *

SUBRUTINA EXTERNA -- TRANSMI.ASM-- PARA TRANSMISION
DE 1 CARACTER POR EL PUERTO SERIAL

Esta Subrutina lo vamos a utilizar en el programa de
Comunicacion # 2 que llama a subrutinas en ENSAMBLADOR .

* *

```
SEGCODIGO       SEGMENT     PUBLIC      'CODE'
                ASSUME      CS:SEGCODIGO
                PUBLIC      TRANSMITA

                OFCAR       EQU 4[BP]
                SBCAR       EQU 6[BP]
                PUERTO      EQU 8[BP]

TRANSMITA       PROC        NEAR
                PUSH        BP
                MOV         BP,SP
                PUSH        DS
                PUSH        DI

                MOV         DS,SBCAR
                MOV         DI,OFCAR
                MOV         AX,DS:[DI]
                MOV         AH,01
                MOV         DX,PUERTO
                INT         14H
                MOV         DS,SBCAR
                MOV         DI,OFCAR
                SUB         BX,BX
                MOV         BL,AH
                MOV         DS:[DI],BX

                POP         DI
                POP         DS
                MOV         SP,BP
                POP         BP
                RET         6
 TRANSMITA      ENDP
 SEGCODIGO      ENDS
                END
```

* *

SUBRUTINA EXTERNA -- ESTATUS.ASM-- PARA VER EL STATUS DEL PUERTO SERIAL

Esta Subrutina nos permite que el programa de comunicacion
2 hecho en pascal ser llamado en subrutina en ENSAMBLADOR

* *

```
SEGCODIGO       SEGMENT     PUBLIC      'CODE'
                ASSUME      CS:SEGCODIGO
                PUBLIC      VERSTATUS

                OFVALOR     EQU 4[BP]
                SBVALOR     EQU 6[BP]
                PUERTO      EQU 8[BP]

VERSTATUS       PROC        NEAR
                PUSH        BP
                MOV         BP,SP
                PUSH        DS
                PUSH        DI

                MOV         DX,PUERTO
                MOV         AH,03
                INT         14H
                AND         AX,0300H
                SUB         BX,BX
                MOV         BL,AH
                MOV         DS,SBVALOR
                MOV         DI,OFVALOR
                MOV         DS:[DI],BX


                POP         DI
                POP         DS
                MOV         SP,BP
                POP         BP
                RET         6
VERSTATUS       ENDP
SEGCODIGO       ENDS
                END
```

* *

PROGRAMA DE CONFIGURACION DEL PUERTO SERIAL # 1 .

Este programa me permite realizar la seleccion de los parametros de comunicación del puerto serial # 1 , el cual nos servira para realizar la comunicación entre las 2 computadoras.

* *

```
#include <bios.h>

main()

{
  int a,b,c ;
  int STATUS ;

  printf("\n  C O M U N I C A C I O N   E N T R E   2  P. C.  \n");
  printf("\n   -----------------------------------------------\n");
  printf("\n");
  printf("\n Digite 1 si es paridad par o 2 si es paridad impar: ");
  scanf("%d",&a);
  printf("\n");
  printf("\n Digite 1 si es un bit-parada o 2 si es dos bit-parada ");
  scanf("%d",&b);
  printf("\n");
  printf("\n   SELECCIONE LAS VELOCIDADES DE COMUNICACION : \n");
  printf("\n          Digite 1 si es 300 bauds   \n");
  printf("\n          Digite 2 si es 1200 bauds  \n");
  printf("\n          Digite 3 si es 2400 bauds  \n");
  printf("\n          Digite 4 si es 4800 bauds  \n");
  printf("\n   CUAL DESEA : ");
  scanf("%d",&c);
  printf("\n");

  if (a == 1)
     if (b == 1)
       if (c == 1)
         STATUS=_bios_serialcom(_COM_INIT,0,_COM_CHR8|_COM_STOP1|_COM_EVENPARITY|_COM_300);
       else if (c == 2)
              STATUS=_bios_serialcom(_COM_INIT,0,_COM_CHR8|_COM_STOP1|_COM_EVENPARITY|_COM_1200);
         else if (c == 3)
              STATUS=_bios_serialcom(_COM_INIT,0,_COM_CHR8|_COM_STOP1|_COM_EVENPARITY|_COM_2400);
     else
       STATUS=_bios_serialcom(_COM_INIT,0,_COM_CHR8|_COM_STOP1|_COM_EVENPARITY|_COM_4800);
       else if (c == 1)
            STATUS=_bios_serialcom(_COM_INIT,0,_COM_CHR8|_COM_STOP2|_COM_EVENPARITY|_COM_300);
         else if (c == 2)
              STATUS=_bios_serialcom(_COM_INIT,0,_COM_CHR8|_COM_STOP2|_COM_EVENPARITY|_COM_1200);
          else if (c == 3)
              STATUS=_bios_serialcom(_COM_INIT,0,_COM_CHR8|_COM_STOP2|_COM_EVENPARITY|_COM_2400);
  else
```

```
STATUS=_bios_serialcom(_COM_INIT,0,_COM_CHR8|_COM_STOP2|_COM_EVENPARITY|_COM_4800);
else if (b == 1)
    if (c == 1)
      STATUS=_bios_serialcom(_COM_INIT,0,_COM_CHR8|_COM_STOP1|_COM_ODDPARITY|_COM_300);
     else if (c == 2)
           STATUS=_bios_serialcom(_COM_INIT,0,_COM_CHR8|_COM_STOP1|_COM_ODDPARITY|_COM_1200);
          else if (c == 3)
               STATUS=_bios_serialcom(_COM_INIT,0,_COM_CHR8|_COM_STOP1|_COM_ODDPARITY|_COM_2400);
   else
      STATUS=_bios_serialcom(_COM_INIT,0,_COM_CHR8|_COM_STOP1|_COM_ODDPARITY|_COM_4800);
    else if (c == 1)
        STATUS=_bios_serialcom(_COM_INIT,0,_COM_CHR8|_COM_STOP2|_COM_ODDPARITY|_COM_300);
      else if (c == 2)
          STATUS=_bios_serialcom(_COM_INIT,0,_COM_CHR8|_COM_STOP2|_COM_ODDPARITY|_COM_1200);
         else if (c == 3)
             STATUS=_bios_serialcom(_COM_INIT,0,_COM_CHR8|_COM_STOP2|_COM_ODDPARITY|_COM_2400);
    else
       STATUS=_bios_serialcom(_COM_INIT,0,_COM_CHR8|_COM_STOP2|_COM_ODDPARITY|_COM_4800);


 printf("\n  LA CONFIGURACION DEL PUERTO  # 1 IMPLEMENTADA \n ");
}
```

* *

PROGRAMA DE COMUNICACION : 3

Este programa de comunicacion nos permite enviar el archivo PLANTA.DTA, desde la P.C. a la P.S.60. Para ejecutar este programa lo hacemos atravez de su programa ejecutable.

* *

```
#include <stdio.h>
#include <bios.h>

main()

{
  char chr='b';
  int STATUS;
  int CONTROLS=19;             Codigo de Seguir Transmitiendo.
  int CONTROLC=03;             Codigo de Parar el Programa.
  FILE *fp;

      fp =fopen("planta.dta","r");
      _bios_serialcom(_COM_SEND,0,CONTROLS);
      STATUS = _bios_serialcom(_COM_STATUS,0,0);
      while (STATUS>20000)
        {
         _bios_serialcom(_COM_SEND,0,CONTROLS);
         STATUS = _bios_serialcom(_COM_STATUS,0,0);
        }
     while (!feof(fp))
       {
         fscanf(fp,"%c",&chr);
         _bios_serialcom(_COM_SEND,0,chr);
         STATUS=_bios_serialcom(_COM_STATUS,0,0);
         while (STATUS>20000)
           {
             _bios_serialcom(_COM_SEND,0,chr);
             STATUS = _bios_serialcom(_COM_STATUS,0,0);
           }
       }
     _bios_serialcom(_COM_SEND,0,CONTROLC);
 }
```

* *

PROGRAMA DE COMUNICACION : 4

Este programa de Comunicacion nos permite ir recibiendo caracter por caracter e ir almacenando en un archivo TEMPHORA .DAT los datos ACTUALES que llegan de la P.C. La forma de ejecutar este programa es llamar al programa ejecutable desde la P.S.60.

* *

```
#include <stdio.h>
#include <bios.h>

main()

{
  int codigo,dummy;
  int PERMISO=00;
  int CONTROLS=19;          Codigo de Permiso en la Transmision.
  int CONTROLC=03;          Codigo de parada.
  int i=0;
  FILE *fp;

     fp = fopen("temphora.dat","w");
    dummy = _bios_serialcom(_COM_RECEIVE,0,0);
    dummy = _bios_serialcom(_COM_RECEIVE,0,0);
    while ( PERMISO != CONTROLS )
     {
      _bios_serialcom(_COM_SEND,0,CONTROLS);
      PERMISO = _bios_serialcom(_COM_RECEIVE,0,0);
     }
    while (codigo != CONTROLC)
     {
       codigo=_bios_serialcom(_COM_RECEIVE,0,0);
       if (codigo == CONTROLC) break;
       if (codigo != CONTROLS)
         {
           ++i;
           if (i > 1) fprintf(fp,"%c",codigo);
         }
       _bios_serialcom(_COM_SEND,0,CONTROLS);
     }
   if (i > 1) printf( "\n COMUNICACION BUENA \n");
}
```

* *

PROGRAMA DE CONTROL ALMACENADO EN LA EPROM .

DIRECCION	CODIGO DE OPERACION	ETIQ. PRINCIPAL	INSTRUCCION	COMENTARIO
0400	C3	INICIO	JMP REC. MODEM	
0401	40			
0402	04			
0403	0E	SIGUE	MVI C , 1A	
0404	1A			
0405	21		LXIH , 04C8	
0406	C8			
0407	04			
0408	11		LXI D , 8228	
0409	28			
040A	82			
040B	CD		CALL TRASLADO	
040C	00			
040D	05			
040E	3E		MVI A , 93	
040F	93			
0410	D3		OUT 07	
0411	07			
0412	3E		MVI A , 82	
0413	82			
0414	D3		OUT 0F	
0415	0F			
0416	21		LXI H , 83F0	
0417	F0			

DIRECCION	CODIGO DE OPERACION	ETIQ. PRINCIPAL	INSTRUCCION	COMENTARIO
0418	83			
0419	36		MVI M , 00	
041A	00			
041B	3E	AQUI	MVI A , 37	
041C	37			
041D	32		STA 83F9	
041E	F9			
041F	83			
0420	DB		IN 05	
0421	05			
0422	E6		ANI 01	
0423	01			
0424	47		MOV B , A	
0425	C5		PUSH B	
0426	CD		CALL MUESTRE	
0427	0A			
0428	05		MVI A , 1E	
0429	3E			
042A	1E			
042B	CD		CALL DELAY	
042C	15			
042D	05			
042E	C1		POP B	
042F	78		MOV A , B	
0430	FE		CPI 01	

DIRECCION	CODIGO DE OPERACION	ETIQ. PRINCIPAL	INSTRUCCION	COMENTARIO
0431	01			
0432	CA		JZ CONTINUA	
0433	00			
0434	84			
0435	C3		JMP AQUI	
0436	1B			
0437	04			
0438	00			
0439	00			
043A	00			
043B	00			
043C	00			
043D	00			
043E	00			
043F	00			
0440	0E	RECEP. MODEM	MVI C , 0A	
0441	0A			
0442	11		LXI D , 8400	
0443	00			
0444	84			
0445	CD	UNO	CALL WAITS MODEM	
0446	2A			
0447	05			
0448	CD		CALL RCV	
0449	40			
044A	05			

DIRECCION	CODIGO DE OPERACION	ETIQ. PRINCIPAL	INSTRUCCION	COMENTARIO
044B	D2		JNC ERROR	
044C	5A			
044D	04			
044E	12		STAX D	
044F	FE		CPI 2A	
0450	2A			
0451	CA		JZ SIGUE	
0452	03			
0453	04			
0454	13		INX D	
0455	C3		JMP UNO	
0456	45			
0457	04			
0458	00			
0459	00			
045A	E7	ERROR	RST 4	
045B	C3		JMP CASETERA	
045C	60			
045D	04			
045E	00			
045F	00			
0460	0E	CASETERA	MVI C , 16	
0461	16			
0462	21		LXI H , LAZO CASSETERA	
0463	80			

DIRECCION	CODIGO DE OPERACION	ETIQ. PRINCIPAL	INSTRUCCION	COMENTARIO
0464	04			
0465	11		LXI D , 8200	
0466	00			
0467	82			
0468	CD		CALL TRASLADO	
0469	00			
046A	05			
046B	0E		MVI C , 1A	
046C	1A			
046D	21		LXI H , Ts. SERV. INT.	
046E	C8			
046F	09			
0470	11		LXI D , 8228	
0471	28			
0472	82			
0473	CD		CALL TRASLADO	
0474	00			
0475	05			
0476	3E	MUESTRE C	MVI A , 39	
0477	39			
0478	32		STA 83F9	
0479	F9			
047A	83		JMP MUESTRE C	
047B	C3			
047C	76			
047D	04			

DIRECCION	CODIGO DE OPERACION	ETIQ. PRINCIPAL	INSTRUCCION	COMENTARIO
047E	00			
047F	00			
0480	CD	LAZO CASSETERA	CALL ENTWD	
0481	46			
0482	03			
0483	22		SHLD 83E4	
0484	E4			
0485	83			
0486	CD		CALL ENTWD	
0487	46			
0488	03			
0489	EB		XCHG	
048A	2A		LHLD 83E4	
048B	E4			
048C	83			
048D	01		LXI B , 007D	
048E	7D			
048F	00			
0490	CA		JZ REC. LAZO	
0491	E5			
0492	04			
0493	C3		JMP TRANS. LAZO	
0494	A0			
0495	04			
0496	00			

DIRECCION	CODIGO DE OPERACION	ETIQ. PRINCIPAL	INSTRUCCION	COMENTARIO
0497	00			
0498	00			
0499	00			
049A	00			
049B	00			
049C	00			
049D	00			
049E	00			
049F	00			
04A0	2A	TRANS. LAZO	LHLD 83E4	
04A1	E4			
04A2	83			
04A3	7E		MOV A , M	
04A4	23		INX H	
04A5	22		SHLD 83E4	
04A6	E4			
04A7	83			
04A8	6F		MOV L , A	
04A9	26		MVI H , 07	
04AA	07			
04AB	29		DAD H	
04AC	22		SHLD 8300	
04AD	00			
04AE	83			
04AF	00		NOP	
04B0	EF	TRES	RST 5	

DIRECCION	CODIGO DE OPERACION	ETIQ. PRINCIPAL	INSTRUCCION	COMENTARIO
04B1	CD		CALL DELYT	
04B2	EB			
04B3	03			
04B4	2A		LHLD 8300	
04B5	00			
04B6	83			
04B7	7C		MOV A , H	
04B8	B5		ORA L	
04B9	C2		JNZ TRES	
04BA	B0			
04BB	04			
04BC	1B		DCX D	
04BD	7A		MOV A , D	
04BE	B3		ORA E	
04BF	C2		JNZ TRANS LAZO	
04C0	A0			
04C1	04			
04C2	E7		RST 4	
04C3	C3		JMP 8200	
04C4	00			
04C5	82			
04C6	00			
04C7	00			
04C8	F3	TRANS.SERIAL INT.	DI	
04C9	F5		PUSH PSW	

DIRECCION	CODIGO DE OPERACION	ETIQ. PRINCIPAL	INSTRUCCION	COMENTARIO
04CA	E5		PUSH H	
04CB	2A		LHLD 8300	
04CC	00			
04CD	83			
04CE	CD		CALL SHLRT	
04CF	2D			
04D0	02			
04D1	CA		JZ 823C	
04D2	3C			
04D3	82			
04D4	22		SHLD 8300	
04D5	00			
04D6	83			
04D7	9F		SBB A	
04D8	E6		ANI 05	
04D9	05			
04DA	F6		ORI 02	
04DB	02			
04DC	D3		OUT PORTDC	
04DD	02			
04DE	E1		POP H	
04DF	F1		POP PSW	
04E0	FB		EF	
04E1	C9		RET	
04E2	00			
04E3	00			

DIRECCION	CODIGO DE OPERACION	ETIQ. PRINCIPAL	INSTRUCCION	COMENTARIO
04E4	00			
04E5	2A	RECEP. LAZO	LHLD 83E4	
04E6	E4			
04E7	83			
04E8	CD	CUATRO	CALL WAIT CASSETERA	
04E9	55			
04EA	05			
04EB	CD		CALL RCV	
04EC	40			
04ED	05			
04EE	D2		JNC ERROR CASSSETERA	
04EF	FA			
04F0	04			
04F1	EB		XCHG	
04F2	12		STAX D	
04F3	EB		XCHG	
04F4	23		INX H	
04F5	C3		JMP CUATRO	
04F6	E8			
04F7	04			
04F8	00			
04F9	00			
04FA	E7	ERROR CASSETERA	RST 4	
04FB	C3		JMP 8200	
04FC	00			

DIRECCION	CODIGO DE OPERACION	ETIQ. PRINCIPAL	INSTRUCCION	COMENTARIO
04FD	82			
04FE	00			
04FF	00			
0500	7E	TRASLADO	MOV A , M	
0501	12		STAX D	
0502	23		INX H	
0503	13		INX D	
0504	0D		DCR C	
0505	C2		JNZ TRASLADO	
0506	00			
0507	05			
0508	C9			
0509	00			
050A	16	MUESTRE	MVI D , 83	
050B	83			
050C	1E		MVI E , FF	
050D	FF			
050E	CD		CALL DBYTE	
050F	95			
0510	02			
0511	C9		RET	
0512	00			
0513	00			
0514	00			
0515	0E	DELAY	MVI C , 64	
0516	64			

DIRECCION	CODIGO DE OPERACION	ETIQ. PRINCIPAL	INSTRUCCION	COMENTARIO
0517	06	SIETE	MVI B , 85	
0518	85			
0519	05	SEIS	DCR B	
051A	C2		JNZ SEIS	
051B	19			
051C	05			
051D	0D		DCR C	
051E	C2		JNZ SIETE	
051F	17			
0520	05			
0521	3D		DCR A	
0522	C2		JNZ DELAY	
0523	15			
0524	05			
0525	C9		RET	
0526	00			
0527	00			
0528	00			
0529	00			
052A	CD	WAITS MODEM	CALL SCAN	
052B	57			
052C	02			
052D	DA		JC CASSETERA	
052E	60			
052F	04			

DIRECCION	CODIGO DE OPERACION	ETIQ. PRINCIPAL	INSTRUCCION	COMENTARIO
0530	DB		IN PORTOB	
0531	01			
0532	1F		RAR	
0533	DA		JC WAITS MODEM	
0534	2A			
0535	05			
0536	CD		CALL DELYC	
0537	EE			
0538	03			
0539	DA		JC WAITS MODEM	
053A	2A			
053B	05			
053C	C9		RET	
053D	00			
053E	00			
053F	00			
0540	AF	RCV	XRA A	
0541	37		STC	
0542	1F	NUEVE	RAR	
0543	DA		JC DELYT	
0544	4C			
0545	05			
0546	CD		CALL DELYT	
0547	EB			
0548	03			
0549	C3		JMP NUEVE	

DIRECCION	CODIGO DE OPERACION	ETIQ. PRINCIPAL	INSTRUCCION	COMENTARIO
054A	42			
054B	05			
054C	CD	DELYT	CALL DELYT	
054D	EB			
054E	03			
054F	C9		RET	
0550	00			
0551	00			
0552	00			
0553	00			
0554	00			
0555	DB	WAITS CASSETERA	IN PORTOB	
0556	01			
0557	1F		RAR	
0558	DA		JC WAITS CASSETERA	
0559	55			
055A	05			
055B	CD		CALL DELYC	
055C	EE			
055D	03			
055E	DA		JC WAITS CASSETERA	
055F	55			
0560	05			
0561	C9		RET	
0562	00			

DIRECCION	CODIGO DE OPERACION	ETIQ. PRINCIPAL	INSTRUCCION	COMENTARIO
0563	00			
0564	00			
0565	21	APAGAR CALENT.	LXI H , 83F0	
0566	F0			
0567	83			
0568	3E		MVI A , EF	
0569	EF			
056A	A6		ANA M	
056B	EB		XCHG	
056C	12		STAX D	
056D	D3		OUT 06	
056E	06			
056F	CD		CALL TRANS.EST.MODEM	
0570	70			
0571	06			
0572	C9		RET	
0573	00			
0574	00			
0575	21	APAGAR AGITADOR	LXI H , 83F0	
0576	F0			
0577	83			
0578	3E		MVI A , DF	
0579	DF			
057A	A6		ANA M	
057B	EB		XCHG	
057C	12		STAX D	

DIRECCION	CODIGO DE OPERACION	ETIQ. PRINCIPAL	INSTRUCCION	COMENTARIO
057D	D3		OUT 06	
057E	06			
057F	CD		CALL TRANS.EST.MODEM	
0580	70			
0581	06			
0582	C9		RET	
0583	00			
0584	00			
0585	21	APAGAR VALVULA	LXI H , 83F0	
0586	F0			
0587	83			
0588	3E		MVI A , 7F	
0589	7F			
058A	A6		ANA M	
058B	EB		XCHG	
058C	12		STAX D	
058D	D3		OUT 06	
058E	06			
058F	CD		CALL TRANS.EST.MODEM	
0590	70			
0591	06			
0592	C9		RET	
0593	00			
0594	00			
0595	3E	APAGAR BOMBA	MVI A , 01	

DIRECCION	CODIGO DE OPERACION	ETIQ. PRINCIPAL	INSTRUCCION	COMENTARIO
0596	01			
0597	D3		OUT 0C	
0598	0C			
0599	CD		CALL DELAC	
059A	15			
059B	05			
059C	3E		MVI A , 00	
059D	00			
059E	D3		OUT 0C	
059F	0C			
05A0	DB		IN 0D	
05A1	0D			
05A2	FE		CPI FF	
05A3	FF			
05A4	C2		JNZ APAG. BOMBA	
05A5	98			
05A6	05			
05A7	DB		IN 04	
05A8	04			
05A9	FE		CPI 19	
05AA	19			
05AB	00			
05AC	C2		JNZ APAG. BOMBA	
05AD	98			
05AE	05			
05AF	21		LXI H , 83F0	

DIRECCION	CODIGO DE OPERACION	ETIQ. PRINCIPAL	INSTRUCCION	COMENTARIO
05B0	F0			
05B1	83			
05B2	3E		MVI A , BF	
05B3	BF			
05B4	A6		ANA M	
05B5	EB		XCHG	
05B6	12		STAX D	
05B7	D3		OUT 06	
05B8	06			
05B9	CD		CALL TRANS.EST.MODEM	
05BA	70			
05BB	06			
05BC	C9		RET	
05BD	00			
05BE	00			
05BF	00			
05C0	21	PRENDER CALENT.	LXI H , 83F0	
05C1	F0			
05C2	83			
05C3	3E		MVI A , 10	
05C4	10			
05C5	B6		ORA M	
05C6	EB		XCHG	
05C7	12		STAX D	
05C8	D3		OUT 06	

DIRECCION	CODIGO DE OPERACION	ETIQ. PRINCIPAL	INSTRUCCION	COMENTARIO
05C9	06			
05CA	CD		CALL TRANS.EST.MODEM	
05CB	70			
05CC	06			
05CD	C9		RET	
05CE	00			
05CF	00			
05D0	21	PRENDER AGITADOR	LXI H , 83F0	
05D1	F0			
05D2	83			
05D3	3E		MVI A , 20	
05D4	20			
05D5	B6		ORA M	
05D6	EB		XCHG	
05D7	12		STAX D	
05D8	D3		OUT 06	
05D9	06			
05DA	CD		CALL TRANS.EST.MODEM	
05DB	70			
05DC	06			
05DD	C9		RET	
05DE	00			
05DF	00			
05E0	21	PRENDER BOMBA	LXI H , 83F0	
05E1	F0			
05E2	83			

DIRECCION	CODIGO DE OPERACION	ETIQ. PRINCIPAL	INSTRUCCION	COMENTARIO
05E3	3E		MVI A , 40	
05E4	40			
05E5	B6		ORA M	
05E6	EB		XCHG	
05E7	12		STAX D	
05E8	D3		OUT 06	
05E9	06			
05EA	CD		CALL TRANS.EST.MODEM	
05EB	70			
05EC	06			
05ED	C9		RET	
05EE	00			
05EF	00			
05F0	21	PRENDER VALVULA	LXI H , 83F0	
05F1	F0			
05F2	83			
05F3	3E		MVI A , 80	
05F4	80			
05F5	B6		ORA M	
05F6	EB		XCHG	
05F7	12		STAX D	
05F8	D3		OUT 06	
05F9	06			
05FA	CD		CALL TRANS.EST.MODEM	
05FB	70			

DIRECCION	CODIGO DE OPERACION	ETIQ. PRINCIPAL	INSTRUCCION	COMENTARIO
05FC	06			
05FD	C9		RET	
05FE	00			
05FF	00			
0600	CD	ESP. X MINUTOS	CALL TRANS.EST.MODEM	
0601	70			
0602	06			
0603	16		MVI D , 3C	
0604	3C			
0605	3E	TRECE	MVI A , 0A	
0606	0A			
0607	CD		CALL DELAY	
0608	15			
0609	05			
060A	15		DCR D	
060B	C2		JNZ TRECE	
060C	05			
060D	06			
060E	1D		DCR E	
060F	C2		JNZ ESP. X MINUTOS	
0610	00			
0611	06			
0612	C9		RET	
0613	00			
0614	00			
0615	CD	SET POINT	CALL TRANS.EST.MODEM	

DIRECCION	CODIGO DE OPERACION	ETIQ. PRINCIPAL	INSTRUCCION	COMENTARIO
0616	70			
0617	06			
0618	3E		MVI A , 01	
0619	01			
061A	D3		OUT 0C	
061B	0C			
061C	CD		CALL DELAY	
061D	15			
061E	05			
061F	3E		MVI A , 00	
0620	00			
0621	D3		OUT 0C	
0622	0C			
0623	DB		IN 0D	
0624	0D			
0625	FE		CPI FF	
0626	FF			
0627	C2		JNZ SET POINT	
0628	15			
0629	06			
062A	DB		IN 04	
062B	04			
062C	C1		POP B	
062D	B8		CMP B	
062E	C5		PUSH B	

DIRECCION	CODIGO DE OPERACION	ETIQ. PRINCIPAL	INSTRUCCION	COMENTARIO
062F	C2		JNZ SET POINT	
0630	15			
0631	06			
0632	C9		RET	
0633	00			
0634	00			
0635	3E	MANTENG. X MINUTOS	MVI A , 01	
0636	01			
0637	D3		OUT 0C	
0638	0C			
0639	CD		CALL DELAY	
063A	15			
063B	05			
063C	3E		MVI A , 00	
063D	00			
063E	D3		OUT 0C	
063F	0C			
0640	DB		IN 0D	
0641	0D			
0642	FE		CPI FF	
0643	FF			
0644	C2		JNZ MANTENG. X MINUT.	
0645	35			
0646	06			
0647	DB		IN 04	
0648	04			

DIRECCION	CODIGO DE OPERACION	ETIQ. PRINCIPAL	INSTRUCCION	COMENTARIO
0649	01		POP B	
064A	B8		CMP B	
064B	C5		PUSH B	
064C	DA		JC MENOR	
064D	56			
064E	06			
064F	E5		PUSH H	
0650	CD		CALL APAG. CALENT.	
0651	65			
0652	05			
0653	C3		JMP DIESICEIS	
0654	5A			
0655	06			
0656	E5	MENOR	PUSH H	
0657	CD		CALL PRENDER CALENT.	
0658	C0			
0659	05			
065A	E1	DIESICEIS	POP H	
065B	16		MVI D , 3C	
065C	3C			
065D	3E	DIESCIOCHO	MVI A , 04	
065E	04			
065F	CD		CALL DELAY	
0660	15			
0661	05			

DIRECCION	CODIGO DE OPERACION	ETIQ. PRINCIPAL	INSTRUCCION	COMENTARIO
0662	15		DCR D	
0663	C2		JNZ DIESCIOCHO	
0664	5D			
0665	06			
0666	2D		DCR L	
0667	C2		JNZ MANT. X MINUTOS	
0668	35			
0669	06			
066A	C9		RET	
066B	00			
066C	00			
066D	00			
066E	00			
066F	00			
0670	0E	TRANS.EST.MODEM	MVI C , 09	
0671	09			
0672	3E		MVI A , E0	
0673	E0			
0674	CD		CALL TRANSMISION	
0675	AA			
0676	06			
0677	DB		IN 04	
0678	04			
0679	CD		CALL TRANSMISION	
067A	AA			
067B	06			

DIRECCION	CODIGO DE OPERACION	ETIQ. PRINCIPAL	INSTRUCCION	COMENTARIO
067C	DB		IN 05	
067D	05			
067E	CD		CALL TRANSMISION	
067F	AA			
0680	06			
0681	DB		IN 06	
0682	06			
0683	E6		ANI 0F	
0684	0F			
0685	CD		CALL TRANSMISION	
0686	AA			
0687	06			
0688	3E		MVI A , EA	
0689	EA			
068A	CD		CALL TRANSMISION	
068B	AA			
068C	06			
068D	CD		CALL MENAB	
068E	22			
068F	02			
0690	0E		MVI C , 0A	
0691	0A			
0692	CD		CALL WAITS MODEM	
0693	2A			
0694	05			

DIRECCION	CODIGO DE OPERACION	ETIQ. PRINCIPAL	INSTRUCCION	COMENTARIO
0695	CD		CALL RCV	
0696	40			
0697	05			
0698	FE		CPI 2A	
0699	2A			
069A	C2		JNZ PARADA	
069B	A0			
069C	06			
069D	C9		RET	
069E	00			
069F	00			
06A0	3E	PARADA	MVI A , 00	
06A1	00			
06A2	D3		OUT 06	
06A4	C3		JMP CASSETERA	
06A5	60			
06A6	04			
06A7	00			
06A8	00			
06A9	00			
06AA	6F	TRANSMISION	MOV L , A	
06AB	26		MVI H , 03	
06AC	03			
06AD	29		DAD H	
06AE	22		SHLD 8300	
06AF	00			

DIRECCION	CODIGO DE OPERACION	ETIQ. PRINCIPAL	INSTRUCCION	COMENTARIO
06B0	B3			
06B1	06		MVI B , 0B	
06B2	0B			
06B3	EF	DIESICIETE	RST 5	
06B4	CD		CALL DELYT	
06B5	EB			
06B6	03			
06B7	2A		LHLD 8300	
06B8	00			
06B9	83			
06BA	05		DCR B	
06BB	C2		JNZ DIESICIETE	
06BC	B3			
06BD	06			
06BE	C9		RET	
06BF	00			

```
Program TOPICO(input,ouput);  { PROGRAMA PRINCIPAL DE VISUALIZACION }
const                         { REALIZADO POR JAVIER ALOVILLO M 91  }
   maxtemp=57;
   encendido=4;
   apagado=1;
   cfondo=3;
   division=%A1;
type
   buffer= array [0..42] of integer;

   longitud= string[10];
   carlong= string[4];
   bufarch= array[1..12] of integer;

   conversion=array[1..5] of integer;
   longplan=array[1..5] of integer;
var
   fondo:integer;
   car,atrib,pntalla:integer;
   poc,pof,numf,numc:integer;
   conta:integer;

   hora,minuto,segundo:integer;
   dia,mes,anio:integer;

   temp:integer;
   tecla:integer;
   anthora,antminuto,antsegundo:integer;
   antemp:integer;
   cocient:integer;
   antcocient:integer;
   antfbloq:integer;

   tipom:integer;
   sel,color:integer;
   ejelv,ejelh,fbase,cbase:integer;
   abarrat:integer;
   antfblqn:integer;
   lcol:integer;
   fi,ci,ff,cf:integer;
   ponga:integer;

   cntiempo:integer;
   ftemp,ctemp:buffer;
   fborre,cborre:buffer;

   banderacurva:integer;
   puntoant:real;
   rango:integer;
   cmc:integer;

   archivo: text;
   filehora:longitud;
   filetemp:longitud;
```

```
   carhora,carmin:carlong;
   archhora,archminuto,archtemp: bufarch;
   cntarchivo:integer;
   archiminant:integer;

   eagit,ecalen,ellav :integer;
   ebomb,enive   :integer;
   estado:integer;
   estadcon:conversion;
   disp:conversion;

   puerto,valor:integer;
   i:integer;
   plandta:longplan;
   aplandta:longplan;

   c1,c2,c3,cip,cfp:integer;

   cambio:integer;

{-------------------------------------------------------------------------}

procedure cursor(ponga:integer);external 'cursor.com';

procedure limpiar(division,fondo:integer);external'limpiar.com';

procedure subg(car,atrib,pntalla,
                numc,posf,posc:integer);external'subg.com';


procedure car1(car,atrib,pntalla,
                posf,posc:integer);external'car1.com';


procedure tiempo(VAR hora,minuto,segundo:integer);external 'tiempoe.com';


procedure fecha(VAR dia,mes,anio:integer);external 'fechae.com';


procedure explora(VAR tecla:integer);external 'explora.com';

procedure modo(tipom:integer);external 'modo1.com';

procedure paletafondo(sel,color:integer); external 'paletfon.com';

procedure pixel(color,numf,numc:integer); external'pixel.com';

procedure timbre;external 'timbre.com';


procedure reciba(puerto:integer; var car:integer);
                                        external 'reciba.com';
```

```
procedure transmita(puerto:integer; var car:integer);
                                        external 'transmi.com';

procedure verstatus(puerto:integer; var valor:integer);
                                        external 'estatus.com';

{-------------------------------------------------------------------}

{-------------------------------------------------------------------}
procedure subgraf(car,atrib,pntalla,numf,numc,posf,posc:integer);
     Begin
        conta:=00;
        while numf>conta do
          begin
               subg(car,atrib,pntalla,numc,posf,posc);
               conta:=conta+1;
               posf:=posf+1;
          end;
    end;
{-------------------------------------------------------------------}

procedure caldera(fond:integer);
   const
     ac=$2E;
     p=0;
     Begin
         subgraf(179,fond,p,10,2,12,24);
         subgraf(196,fond,p,2,30,21,25);
         subgraf(179,fond,p,10,2,12,54);
         subgraf(219,fond,p,3,1,21,43);
         car1(192,fond,p,21,25);
         car1(192,fond,p,22,24);
         car1(217,fond,p,21,54);
         car1(217,fond,p,22,55);
         car1(191,fond,p,22,32);
         car1(218,fond,p,22,33);
     end;
{-------------------------------------------------------------------}

procedure agitador(aa:integer);
   const

     p=0;
     Begin
         subgraf(222,aa,p,3,1,07,37);
         subgraf(221,aa,p,3,1,07,41);
         subgraf(223,aa,p,1,3,07,38);
         subgraf(220,aa,p,1,3,09,38);

         subgraf(186,aa,p,4,1,10,39);
         car1(205,aa,p,14,39);
         car1(16,aa,p,14,38);
         car1(17,aa,p,14,40);
         car1(65,fondo+encendido,p,08,39);
```

```
    end;
{-------------------------------------------------------------------------}


procedure calentador(aa:integer);
   const

     p=0;
     Begin
         subgraf(179,aa,p,4,1,07,46);
         car1(47,aa,p,11,46);
         car1(92,aa,p,12,46);
         car1(47,aa,p,13,46);
         car1(92,aa,p,14,46);

     end;
{-------------------------------------------------------------------------}

procedure termometro(aa:integer);
   const

     p=0;
     Begin
         subgraf(215,aa,p,7,1,08,33);
         car1(79,aa,p,07,33);
     end;
{-------------------------------------------------------------------------}

procedure llave(aa:integer);
   const

     p=0;
     Begin
         subgraf(223,aa,p,1,8,20,16);
         car1(219,aa,p,19,19);
         car1(16,aa,p,19,18);
         car1(17,aa,p,19,20);
     end;
{-------------------------------------------------------------------------}

procedure bomba(aa:integer);
   const

     p=0;
     Begin
         subgraf(222,aa,p,3,1,12,16);
         subgraf(221,aa,p,3,1,12,22);
         subgraf(223,aa,p,1,5,12,17);
         subgraf(220,aa,p,1,5,14,17);
         car1(220,aa,p,13,23);
         car1(66,fondo+encendido,p,13,19);
     end;
{-------------------------------------------------------------------------}
```

```
procedure nivel(aa:integer);
   const

     p=0;
     Begin
         subgraf(219,aa,p,6,28,15,26);
     end;
{-------------------------------------------------------------------------}

procedure imprimafecha;
 const
     aa=$0E;
     p=0;
     Begin
         car1(00,aa,p,2,7);
         write( '  Fecha  ');
         car1(00,aa,p,2,38);
         write( '  Hora  ');
         car1(00,aa,p,2,70);
         write(' Temp C');
         car1(00,aa,p,3,70);
         write('       ');
         fecha(dia,mes,anio);
         car1(00,aa,p,3,7);
         write(dia,'-',mes,'-',anio,' ')
     end;
{-------------------------------------------------------------------------}

procedure imprimahora;
 const
     aa=$0E;
     p=0;
 var
     borre:integer;

     Begin
         borre:=00;
         tiempo(hora,minuto,segundo);
         if anthora<> hora then
            begin
               if hora<10 then
                  begin
                     car1(48,aa,p,3,38);
                     car1(00,aa,p,3,39);
                     write(hora,':');
                     anthora:=hora;
                  end
               else
                  begin
                     car1(00,aa,p,3,38);
                     write(hora,':');
                     anthora:=hora;
                  end;
```

```
            end;

            if antminuto<> minuto then
            begin
               if minuto<10 then
                  begin
                     car1(48,aa,p,3,41);
                     car1(00,aa,p,3,42);
                     write(minuto,':');
                     antminuto:=minuto;
                  end
               else
                  begin
                     car1(00,aa,p,3,41);
                     write(minuto,':');
                     antminuto:=minuto;
                  end;
            end;

            if antsegundo<> segundo then
            begin
               if segundo<10 then
                  begin
                     car1(48,aa,p,3,44);
                     car1(00,aa,p,3,45);
                     write(segundo);
                     antsegundo:=segundo
                  end
               else
                  begin
                     car1(00,aa,p,3,44);
                     write(segundo);
                     antsegundo:=segundo;
                  end;
            end;

    end;                                    { fin procedimiento}
{-------------------------------------------------------------------------}
procedure imprimatemp;
 const
     aa=$0E;
     p=0;
     Begin
         temp:=plandta[1];
         if (antemp<> temp) or (cambio=1) then
            begin
               if temp<10 then
                  begin
                     car1(48,aa,p,3,72);
                     car1(00,aa,p,3,73);
                     write(temp);
                     antemp:=temp;
                  end
               else
```

```
                begin
                   car1(00,aa,p,3,72);
                   write(temp);
                   antemp:=temp;
                end;
          end;

    end;                                        { fin procedimiento}

{-------------------------------------------------------------------------}
procedure pixgraf(atrib,numf,numc,posf,posc:integer);
var
     conf,conc:integer;
     Begin
        conf:=00;
        conc:=00;

        while numf>conf do
          begin
               while numc>conc do
                  begin
                    pixel(atrib,posf,posc);
                    posc:=posc+1;
                    conc:=conc+1;
                  end;
               posc:=posc-conc;
               conc:=0;
               posf:=posf+1;
               conf:=conf+1;
          end;
    end;
{-------------------------------------------------------------------------}

procedure cero(at,pfi,pco:integer);
     Begin

            pixgraf(at,1,3,pfi,pco);
            pixgraf(at,1,3,pfi+4,pco);
            pixgraf(at,5,1,pfi,pco);
            pixgraf(at,5,1,pfi,pco+2);

     end;
{-------------------------------------------------------------------------}

procedure ejes;
   const
     aa=1;
     d=40;
   var
     n:integer;
     Begin

         n:=0;
         pixgraf(aa,100,1,50,n+20);
```

```
pixgraf(3,1,2,60,n+20);
   pixgraf(3,4,1,41,n+14);
   pixgraf(3,4,1,41,n+17);
   pixel(3,42,n+15);
   pixel(3,43,n+16);

   pixgraf(3,5,1,58,n+2);

   cero(3,58,4);

   cero(3,58,8);

   pixel(3,58,n+13);
   pixel(3,61,n+16);

   pixel(3,58,n+16);
   pixel(3,59,n+15);
   pixel(3,60,n+14);
   pixel(3,61,n+13);

n:=52;
pixgraf(aa,100,1,50,49);
pixgraf(aa,1,30,150,20);
pixgraf(3,1,2,120,49);

   pixgraf(3,1,3,118,0+n);
   pixgraf(3,1,3,120,0+n);
   pixgraf(3,1,3,122,0+n);
   pixgraf(3,5,1,118,2+n);

   cero(3,118,56);

pixgraf(3,1,2,90,9+d);
   pixgraf(3,1,3,88,0+n);
   pixgraf(3,1,3,90,0+n);
   pixgraf(3,1,3,92,0+n);
   pixgraf(3,3,1,90,2+n);
   pixgraf(3,5,1,88,0+n);

   cero(3,88,n+4);

pixgraf(3,1,2,60,9+d);
   pixgraf(3,1,3,58,0+n);
   pixgraf(3,1,3,60,0+n);
   pixgraf(3,1,3,62,0+n);
   pixgraf(3,5,1,58,2+n);
   pixgraf(3,3,1,58,0+n);

   cero(3,58,n+4);


   pixgraf(3,5,1,41,4+n);
   pixgraf(3,1,5,41,2+n);
```

```
          pixgraf(2,1,2,70,9+d);


        n:=d+15;
        pixgraf(2,101,1,50,9+n);
        pixgraf(3,1,2,120,9+n);
        pixgraf(3,1,2,90,9+n);
        pixgraf(3,1,2,60,9+n);
        pixgraf(2,1,213,49,9+n);
        pixgraf(2,1,213,151,9+n);
        n:=250+17;
        pixgraf(2,102,1,49,9+n);
        pixgraf(3,1,2,120,9+n);
        pixgraf(3,1,2,90,9+n);
        pixgraf(3,1,2,60,9+n);
        pixgraf(1,1,211,60,66);

        pixgraf(3,2,1,152,135);
        pixgraf(3,2,1,152,205);

        pixgraf(3,5,1,156,135);

        pixgraf(3,1,4,156,205);
        pixgraf(3,1,4,158,205);
        pixgraf(3,1,4,160,205);
        pixgraf(3,3,1,156,208);
        pixgraf(3,3,1,158,205);

        n:=274;
        pixgraf(3,5,1,154,n+1);
        pixgraf(3,1,3,155,n);
        pixgraf(3,1,2,158,n+1);

    end;
{-------------------------------------------------------------------------------}

procedure barra( temp:integer);
   const
     aa=3;
     p=0;
   var
      bloq:integer;
      cocient,extra:integer;
      residuo:real;
      nbloqborrar,bloqimp:integer;
      fbloqa:integer;

    Begin
        abarrat:=5;
        fbase:=150;
        cbase:=40;
        bloq:=temp;
        fbloqa:=fbase-bloq;
        if fbloqa>antfbloq then
```

```
            begin
              nbloqborrar:=fbloqa-antfbloq;
              pixgraf(0,nbloqborrar,abarrat,antfbloq,cbase);
            end
        else
            begin
               pixgraf(3,bloq,abarrat,fbloqa,cbase);
             end;
          antfbloq:=fbloqa;

   end;                          { fin del procedimiento }
{-------------------------------------------------------------------------}
procedure nivelbar( nivel:integer);
   const
     aa=3;
   var
      blq:integer;
      nblqborrar:integer;
      fblqa:integer;

    Begin
        abarrat:=5;
        fbase:=150;
        cbase:=25;
        if estadcon[5]=1 then blq:=90 else blq:=0;
        fblqa:=fbase-blq;
        if fblqa>antfblqn then
            begin
              nblqborrar:=fblqa-antfblqn;
              pixgraf(0,nblqborrar,abarrat,antfblqn,cbase);
            end
        else
            begin
               pixgraf(3,blq,abarrat,fblqa,cbase);
             end;
          antfblqn:=fblqa;

   end;                          { fin del procedimiento }
{-------------------------------------------------------------------------}

procedure glinea(lcol,yi,xi,yf,xf:integer);

var
      error,deltax,deltay:integer;
      mitad,suma,delx,dely:integer;
      m,fi,ci,ff,cf:integer;
begin
         error:=0;
         if (xi>=0)and(xf>=0)and(yi>=0)and(yf>=0) then
            begin
               ci:=xi; cf:=xf; fi:=yi; ff:=yf;
               if cf<ci then
                  begin
                     ci:=xf;
```

```
            cf:=xi;
            fi:=yf;
            ff:=yi;
          end;
        deltax:=abs(cf-ci);
        deltay:=abs(ff-fi);
        delx:=cf-ci;
        dely:=ff-fi;
        if (dely>0) then m:=1  else m:=-1;
        if ( deltax > deltay ) then
          begin
            mitad:=trunc((deltax)/2);
            while ci<=cf do
              begin
                pixel(lcol,fi,ci);
                ci:=ci+1;
                suma:=error+ deltay;
                if suma>mitad  then
                  begin
                    error:= suma-deltax;
                    fi:=fi+m;
                  end
                else
                  error:=suma;
              end;
          end
        else
            begin
              mitad:=trunc((deltay)/2);
              while fi<>ff do
                begin
                  pixel(lcol,fi,ci);
                  fi:=fi+m;
                  suma:=error+deltax;
                  if suma> mitad  then
                    begin
                      error:= suma-deltay;
                      ci:=ci+1;
                    end
                  else
                    error:= suma;
                end;
                if fi=ff then
                      begin
                        pixel (lcol,fi,ci);
                      end;
            end;
      end;


end;              {     fin del procedimiento    }
{-------------------------------------------------------------------}
procedure actualice( temp:integer);
  var
```

```
    multiplo:real;
    contad:integer;
    s:integer;
  Begin
      multiplo:= segundo/5 - trunc(segundo/5);
      if  (temp >= maxtemp) then timbre;
      if segundo<>puntoant then
      begin
        if  multiplo=0.0 then
           begin
           if cntiempo> rango then cmc:=2;
              case cmc of
              1:
                 begin
                     ftemp[cntiempo]:=150-temp;
                     ctemp[cntiempo]:= cntiempo*5+65;
                     cntiempo:= cntiempo+1;
                 end;
              2:
                 begin
                     contad:=0;
                     cmc:=2;
                     while contad<=rango do
                      begin
                         fborre[contad]:= ftemp[contad];
                         cborre[contad]:= ctemp[contad];
                         contad:= contad+1;
                      end;
                     contad:=0;
                     while contad<= rango-1 do
                       begin
                          ftemp[contad]:=fborre[contad+1];
                          contad:= contad+1;
                        end;
                      ftemp[contad]:=150-temp;  { 1 menos margen }
                      cmc:=2;
                   end;
             end;  { case }
            banderacurva:=1;
            puntoant:=segundo;
         end;                  { if multiplo 0        }
      end;                     { if multiplo anterior  }
    end;                       { fin del procedimiento }
{-------------------------------------------------------------------}

procedure lineatiempo;
   var
      cntloc:integer;
      colinea:integer;  { color dela linea }
      fil,cil,ffl,cfl:integer;
      tempoloc:integer;
    Begin
       if (banderacurva=1) and (cntiempo>=1) then
          begin
```

```
            cntloc:=0; colinea:=3;
            tempoloc:=cntiempo-1;
            case cmc of
            1:
                begin
                       if cntiempo >= 1 then
                         begin
                           cntloc:=0;
                           while  cntloc < cntiempo-1 do
                             begin
                               fil:=ftemp[cntloc];
                               cil:=ctemp[cntloc];
                               ffl:=ftemp[cntloc+1];
                               cfl:=ctemp[cntloc+1];
                               glinea(3,fil,cil,ffl,cfl);
                                cntloc:= cntloc+1;
                             end;
                         end;
                 end;
            2:
                 begin
                      cntloc:=0;
                      while cntloc<rango do  { cambio }
                          begin
                             fil:=fborre[cntloc];   { borro 1 anterior}
                             cil:=cborre[cntloc];
                             ffl:=fborre[cntloc+1];
                             cfl:=cborre[cntloc+1];
                             glinea(0,fil,cil,ffl,cfl);

                             fil:=ftemp[cntloc];    { grafico nueva lin }
                             cil:=ctemp[cntloc];
                             ffl:=ftemp[cntloc+1];
                             cfl:=ctemp[cntloc+1];
                             glinea(3,fil,cil,ffl,cfl);
                             cntloc:= cntloc+1;
                           end;
                     end;        { 1 }
                end;   { case }
                banderacurva:=0;
              end;               { if }

    end;                        { fin del procedimiento }
{-------------------------------------------------------------------}
procedure hagarchivo;
var
    i: integer;
    blancoh,blancom:carlong;  { 4 caracters}
begin
     assign(archivo,'planta.dta');
     rewrite(archivo);
     for i:= 1 to 12 do
       begin
          str(archhora[i],carhora);
```

```
          str(archminuto[i],carmin);
          str(archtemp[i]:8,filetemp);
          if(archhora[i]<10) then
                  blancoh:='0'
              else
                  blancoh:='';
          if(archminuto[i]<10) then
                   blancom:=':0'
                else
                   blancom:=':';
          filehora:= concat(blancoh,carhora,blancom,carmin );
          writeln(archivo,filehora,filetemp);
        end;
     close(archivo);
end;
{-------------------------------------------------------------------}
procedure actarchivo( temp:integer);
   var
      multiplof:real;
      contadf:integer;
      sf:integer;
      intervalo:integer;
    Begin

           intervalo:=5;
           multiplof:= minuto/intervalo -trunc(minuto/intervalo);
           if minuto<>archiminant then
           begin
             if  multiplof=0.0 then
                begin
                   archhora[cntarchivo]:=hora;
                   archminuto[cntarchivo]:=minuto;
                   archtemp[cntarchivo]:=temp;
                   cntarchivo:=cntarchivo+1;
                end;                  { if multiplo 0 }
                archiminant:=minuto;
                if cntarchivo-1=12 then
                  begin
                    hagarchivo;
                    timbre;
                    cntarchivo:=1;
                    tecla:=4;          { para que empieze luego la trans}
                  end;
                  archiminant:=minuto;
           end;                          { if multiplo }
     end;                                { fin del procedimiento }
{-------------------------------------------------------------------}

procedure conversiondatos;         { conversion de datos de la planta }
var
    aleatorio: integer;
    residestado:real;
    numi,a: integer;
begin
```

```
        temp:=aplandta[1];          { meto la temperatura   }
        estado:=aplandta[3];        {  }
        aplandta[2]:=1;
        num1:=estado;
        estadcon[5]:=aplandta[2];     { asigno nivel para grafi  }
        for a:=1 to 4 do
         begin
              residestado:=num1/2 - trunc(num1/2);
              if residestado<>0.0 then
                  begin
                       estadcon[a]:=1;
                       num1:= trunc(num1/2);
                  end
              else
                  begin
                       estadcon[a]:=0;
                       num1:=num1 div 2;
                  end;
          end;

end;
{-------------------------------------------------------------------------}


procedure ingresardatos;                { verifico cod inicio de datos   }
var                                     { ingreso datos mando a convertir}
    c:integer;
    margen:integer;
    aster:integer;                      { y envio car de recibido        }
     Begin                              {  retorna en aplandta[] valor de}
                                        { tem nivel estado como enteros  }
            aster:=42;
            if cfp=234 then
               begin
                 transmita(puerto,aster);
                 cfp:=0;
                end;
            verstatus(puerto,valor);
            if valor=1 then
             begin
                  reciba(puerto,cip);
                   if cip= 224 then
                     begin
                       reciba(puerto,c1);
                       reciba(puerto,c2);
                       reciba(puerto,c3);
                       reciba(puerto,cfp);
                       plandta[1]:=c1;
                       plandta[2]:=c2;
                       plandta[3]:=c3;
                       plandta[4]:=cfp;
                       if (plandta[4]=234) then
                           begin
                                    for c:=1 to 3 do
```

```
                                    begin
                                      aplandta[c]:=plandta[c];
                                    end;
                                    conversiondatos;
                                   cambio:=1;
                            end;

                    end;
                end;

    end;                          { fin del procedimiento }
{-------------------------------------------------------------------------}

procedure grafestadoplanta;
var
   a: integer;
begin
          for a:= 1 to 4 do
             begin
                if estadcon[a]=1 then disp[a];=encendido else disp[a]:=apagado;
             end;
           if estadcon[5]=1 then disp[5]:=7 else disp[5]:=fondo div 16;

          calentador(fondo+disp[1]);
          agitador(fondo+disp[2]);
          bomba(fondo+disp[3]);
          llave(fondo+disp[4]);             { 1 varia atributo }
          nivel(fondo+disp[5]);            { 7 color blanco 0 vacio }
          cambio:=0;
end;
{-------------------------------------------------------------------------}

procedure falla;
var                                  { cuando pulsan esc mando a apagar}
   aster:integer;                    { la planta                       }
   margen:integer;                   { y envio car de recibido       }
   aa:integer;
    Begin                            {  retorna en aplandta[] valor de}
          aa:=$0E;
          cfp:=00;
          aster:=$ff;
          cursor(0);
          timbre;
          carl(00,aa,0,10,20);
          write('.........................');
          carl(00,aa,0,12,20);
          write('TECLA DE EMERGENCIA PULSADA');
          carl(00,aa,0,14,20);
          write('.........................');
          repeat                 { Caracter de falla }
            verstatus(puerto,valor);
            if valor=1 then
            begin
                 reciba(puerto,cip);
```

```
                if cip=224 then
                  begin
                    reciba(puerto,c1);
                    reciba(puerto,c2);
                    reciba(puerto,c3);
                    reciba(puerto,cfp);
                    if cfp=234 then
                      transmita(puerto,aster);
                  end;
              end;
            until cfp=234;
            modo(3);
            car1(00,aa,0,10,20);
            write('LA PLANTA HA SIDO APAGADA POR COMPLETO ');
            car1(00,aa,0,12,20);
            write(' POR FAVOR VERIFICAR FALLA PRODUCIDA  ');
            car1(00,aa,0,14,20);
            write('.....');
            cursor(0);
            readln;
    end;                          { fin del procedimiento }
{-------------------------------------------------------------------------}

{-------------------------------------------------------------------------}

Begin
    tecla:=1;
    anthora:=25;
    antminuto:=61;
    antsegundo:=61;
    antemp:=-1;
    antfbloq:=150;
    antfblqn:=antfbloq;
    fondo:=cfondo*16;
    cntiempo:=0;
    banderacurva:=0;
    puntoant:=-1;
    rango:=42;
    cmc:=1;
    cntarchivo:=1;
    archiminant:=-1;
    estado:=0;
    cambio:=1;
    plandta[1]:=25;
    puerto:=1;
    cfp:=234;

    for i:=1 to 5 do
      estadcon[i]:=00;
    cfp:=0;
  repeat
    explora(tecla);
    while tecla=1 do
      begin
```

```
        modo(3);
        cursor(0);
        cambio:=1;
        limpiar(division,fondo);
        caldera(fondo+14);
        termometro(fondo+14);
        imprimafecha;
       repeat
         imprimahora;
         ingresardatos;
         if cambio=1 then
            begin
               imprimatemp;
               grafestadoplanta;
            end;
         explora(tecla);
         actualice(temp);
         actarchivo(temp);
       until tecla<>1;
       cambio:=1;
     end;
    while tecla=2  do
       begin
          paletafondo(0,$04);
          paletafondo(1,0);
          modo(04);
          paletafondo(1,1);
          ejes;
          writeln('Panel 2                Alarma T=90°C');
          banderacurva:=1;
          repeat
            tiempo(hora,minuto,segundo);
            ingresardatos;
            barra(temp);
            nivelbar(temp);
            explora(tecla);
            actualice(temp);
            actarchivo(temp);
            lineatiempo;
          until tecla<>2;
       end;
       if tecla= 3 then
          begin
            modo(3);
            falla;
            cursor(1);
            tecla:=4;
          end;
       anthora:=25;
       antminuto:=61;
       antsegundo:=61;

   until tecla=4;
end.                              { FIN DEL PROGRAMA PRINCIPAL  }
```

```
;-------------------------------------------------------------
; SUBRUTINA EXTERNA CURSOR PARA ELIMINAR EL CURSOR EN MODO DE
; TEXTO
;-------------------------------------------------------------
;-------------------------------------------------------------

SEGCODIGO SEGMENT  PUBLIC        'CODE'
  ASSUME  CS:SEGCODIGO
  PUBLIC  CURSOR

;             OBTENGO LA VARIABLES DE ENTRADA

  PONGA  EQU 4[BP]

 CURSOR  PROC     NEAR
   PUSH  BP
   MOV  BP,SP
   PUSH  DS

                 MOV  AX,PONGA
   CMP  AX,00
   JZ  BORRE
   SUB  AX,AX
   MOV  CH,06
   MOV  CL,07
   MOV  AH,01
   INT  10H
   JMP  SALGA

  BORRE:  SUB  AX,AX
    MOV  CH,20H
    MOV  CL,00
    MOV  AH,01
    INT  10H
  SALGA:

    POP  DS
    MOV  SP,BP
    POP  BP
    RET  2
  CURSOR  ENDP
  SEGCODIGO ENDS
    END
```

```
;-----------------------------------------------------------------
;       SUBRUTINA EXTERNA  LIMPIAR
;-----------------------------------------------------------------

SEGCODIGO SEGMENT  PUBLIC  'CODE'
  ASSUME  CS:SEGCODIGO
  PUBLIC  LIMPIAR

  FONDO  EQU 4[BP]
  DIVISION EQU 6[BP]

LIMPIAR  PROC     NEAR
  PUSH  BP
  MOV  BP,SP
  PUSH  DS

  MOV  DX,0000
  MOV  BH,00
  MOV  AH,02
  INT  10H

  MOV  AL,00
  MOV  BH,00
  MOV  BL,FONDO
  MOV  CX,2000
  MOV  AH,09
  INT  10H

  MOV  DX,0000
  MOV  BH,00
  MOV  AH,02
  INT  10H


  MOV  AL,00
  MOV  BH,00
  MOV  BL,DIVISION
  MOV  CX,480
  MOV  AH,09
  INT  10H

  POP  DS
  MOV  SP,BP
  POP  BP
  RET  4
 LIMPIAR  ENDP
 SEGCODIGO ENDS
   END
```

```
;-------------------------------------------------------------------
; SUBRUTINA EXTERNA PARA GRAFICACION DE LINEAS O CARACTER
;-------------------------------------------------------------------

;-------------------------------------------------------------------
SEGCODIGO SEGMENT  PUBLIC        'CODE'
  ASSUME  CS:SEGCODIGO
  PUBLIC  SUBG

;            OBTENGO LA VARIABLES DE ENTRADA

  POSC  EQU 4[BP]
  POSF  EQU 6[BP]
  NUMC  EQU 8[BP]
  PNTALLA  EQU 10[BP]
  ATRIB  EQU 12[BP]
  CAR  EQU 14[BP]

SUBG  PROC     NEAR
  PUSH  BP
  MOV  BP,SP
  PUSH  DS


  MOV  DH,POSF
  MOV  DL,POSC
  MOV  BH,PNTALLA
  MOV  AH,02
  INT  10H


  MOV  AH,09
  MOV  AL,CAR
  MOV  BH,PNTALLA
  MOV  BL,ATRIB
  MOV  CX,NUMC
  INT  10H

  POP  DS
  MOV  SP,BP
  POP  BP
  RET  12
SUBG  ENDP
SEGCODIGO ENDS
  END
```

```
;-------------------------------------------------------------------
; SUBRUTINA EXTERNA CAR1 PARA GRAFICACION DE 1 CARACTER
;-------------------------------------------------------------------

;-------------------------------------------------------------------
SEGCODIGO SEGMENT  PUBLIC        'CODE'
  ASSUME  CS:SEGCODIGO
  PUBLIC  CAR1

  POSC  EQU 4[BP]
  POSF  EQU 6[BP]
  PNTALLA  EQU 8[BP]
  ATRIB  EQU 10[BP]
  CAR  EQU 12[BP]

CAR1  PROC     NEAR
  PUSH  BP
  MOV  BP,SP
  PUSH  DS


  MOV  DH,POSF
  MOV  DL,POSC
  MOV  BH,PNTALLA
  MOV  AH,02
  INT  10H


  MOV  AL,CAR
  MOV  BH,PNTALLA
  MOV  BL,ATRIB
  MOV  CX,01
  MOV  AH,09
  INT  10H

  POP  DS
  MOV  SP,BP
  POP  BP
  RET  10
 CAR1  ENDP
 SEGCODIGO ENDS
   END
```

```
;-------------------------------------------------------------------
; SUBRUTINA EXTERNA TIEMPO PARA EXTRAER EL TIEMPO
;-------------------------------------------------------------------
;-------------------------------------------------------------------

SEGCODIGO SEGMENT  PUBLIC        'CODE'
  ASSUME  CS:SEGCODIGO
  PUBLIC  TIEMPO

  OFSEGUNDO EQU 4[BP]
  SBSEGUNDO EQU 6[BP]
  OFMINUTO EQU 8[BP]
  SBMINUTO EQU 10[BP]
  OFHORA  EQU 12[BP]
  SBHORA  EQU 14[BP]

TIEMPO  PROC     NEAR
  PUSH  BP
  MOV  BP,SP
  PUSH  DS
  PUSH  DI


  MOV  AH,2CH
  INT  21H
  MOV  AX,0000
  MOV  AL,CH

  MOV  DS,SBHORA
  MOV  DI,OFHORA
  MOV  DS:[DI],AX

  MOV  AX,0000
  MOV  AL,CL
   MOV  DS,SBMINUTO
  MOV  DI,OFMINUTO
  MOV  DS:[DI],AX

  MOV  AX,0000
  MOV  AL,DH
  MOV  DS,SBSEGUNDO
  MOV  DI,OFSEGUNDO
  MOV  DS:[DI],AX


  POP  DI

  POP  DS
  MOV  SP,BP
  POP  BP
  RET  12
TIEMPO  ENDP
SEGCODIGO ENDS
  END
```

```
;-------------------------------------------------------------------
; SUBRUTINA EXTERNA -- FECHAE.ASM-- PARA EXTRAER LA FECHA
;-------------------------------------------------------------------

;-------------------------------------------------------------------
SEGCODIGO SEGMENT  PUBLIC        'CODE'
  ASSUME  CS:SEGCODIGO
  PUBLIC  FECHA

  OFANIO  EQU 4[BP]
  SBANIO  EQU 6[BP]
  OFMES   EQU 8[BP]
  SBMES   EQU 10[BP]
  OFDIA   EQU 12[BP]
  SBDIA   EQU 14[BP]

FECHA  PROC     NEAR
  PUSH  BP
  MOV  BP,SP
  PUSH  DS
  PUSH  DI


  MOV  AH,2AH
  INT  21H
  MOV  AX,0000
  MOV  AL,CL
  SUB  AX,108

  MOV  DS,SBANIO
  MOV  DI,OFANIO
  MOV  DS:[DI],AX

  MOV  AX,0000
  MOV  AL,DH
   MOV  DS,SBMES
  MOV  DI,OFMES
  MOV  DS:[DI],AX

  MOV  AX,0000
  MOV  AL,DL
  MOV  DS,SBDIA
  MOV  DI,OFDIA
  MOV  DS:[DI],AX


   POP  DI
   POP  DS
   MOV  SP,BP
   POP  BP
   RET  12
 FECHA  ENDP
 SEGCODIGO ENDS
   END
```

```
;-------------------------------------------------------------------
; SUBRUTINA EXTERNA -- EXPLORA.ASM-- PARA EXPLORAR EL TECLADO
;-------------------------------------------------------------------

;-------------------------------------------------------------------
SEGCODIGO SEGMENT  PUBLIC        'CODE'
  ASSUME  CS:SEGCODIGO
  PUBLIC  EXPLORA

  OFTECLA  EQU 4[BP]
  SBTECLA  EQU 6[BP]

EXPLORA  PROC     NEAR
  PUSH    BP
  MOV     BP,SP
  PUSH    DS
  PUSH    DI

  MOV     AH,0BH
  INT     21H
  CMP     AL,00
  JZ      EXPLORAFIN

  MOV     AH,08H
  INT     21H
  CMP     AL,00
  JNZ     LETRA
  MOV     AH,08H
  INT     21H
  CMP     AL,3BH
  JZ      MONITOR1
  CMP     AL,3CH
  JZ      MONITOR2
  JMP     EXPLORAFIN

 LETRA:

  CMP     AL,1BH
  JZ      FINPROG
  JMP     EXPLORAFIN

 MONITOR1:

  MOV   AX,0000
  MOV   AL,01H

  MOV  DS,SBTECLA
  MOV  DI,OFTECLA
  MOV  DS:[DI],AX

  JMP  EXPLORAFIN

 MONITOR2:
```

```
  MOV  AX,0000
  MOV  AL,02H

  MOV  DS,SBTECLA
  MOV  DI,OFTECLA
  MOV  DS:[DI],AX

  JMP  EXPLORAFIN

FINPROG:

  MOV  AX,0000
  MOV  AL,03H

  MOV  DS,SBTECLA
  MOV  DI,OFTECLA
  MOV  DS:[DI],AX

EXPLORAFIN:
  POP  DI
  POP  DS
  MOV  SP,BP
  POP  BP
  RET  4
EXPLORA  ENDP
SEGCODIGO ENDS
  END
```

```
;-------------------------------------------------------------------
; SUBRUTINA EXTERNA --MODO(pantalla)-- CAMBIAMOS DE MODO TEXTO
;  25* 80 A ALTA RESOLUCION 320*200
;-------------------------------------------------------------------

;-------------------------------------------------------------------
SEGCODIGO SEGMENT  PUBLIC        'CODE'
ASSUME  CS:SEGCODIGO
PUBLIC  MODO

TIPO    EQU 4[BP]

MODO   PROC     NEAR
PUSH   BP
MOV    BP,SP
PUSH   DS

MOV    AL,TIPO
MOV    AH,00H
INT    10H

POP    DS
MOV    SP,BP
POP    BP
RET    2

MODO  ENDP
SEGCODIGO ENDS
  END
```

```
;-------------------------------------------------------------------
; SUBRUTINA EXTERNA --PALETAFONDO(sel,color)-- PARA SELECCIONAR
; LA PALETA Y EL COLOR DE FONDO DE LA PANTALLA EN ALTA RESOL
;-------------------------------------------------------------------

;-------------------------------------------------------------------
  SEGCODIGO SEGMENT  PUBLIC        'CODE'
  ASSUME  CS:SEGCODIGO
  PUBLIC  PALETAFONDO

  COLOR  EQU 4[BP]
  SEL  EQU 6[BP]

  PALETAFONDO PROC     NEAR

  PUSH     BP
  MOV      BP,SP
  PUSH     DS

  SUB      AX,AX
  MOV      BH,SEL
  MOV      BL,COLOR
  MOV      AH,0BH
  INT      10H

  POP      DS
  MOV      SP,BP
  POP      BP
  RET      4

  PALETAFONDO ENDP
  SEGCODIGO ENDS
  END
```

```
;-------------------------------------------------------------------
;       SUBRUTINA EXTERNA --PIXEL(color,pfila,pcol)--
;-------------------------------------------------------------------

;-------------------------------------------------------------------
SEGCODIGO       SEGMENT         PUBLIC          'CODE'
                ASSUME          CS:SEGCODIGO
                PUBLIC          PIXEL

                PCOL            EQU     4[BP]
                PFILA           EQU     6[BP]
                COLOR           EQU     8[BP]

PIXEL           PROC            NEAR
                PUSH            BP
                MOV             BP,SP
                PUSH            DS

                SUB             AX,AX
                SUB             DX,DX
                MOV             AL,COLOR
                MOV             DL,PFILA
                MOV             CX,PCOL
                MOV             AH,0CH
                INT             10H

                POP             DS
                MOV             SP,BP
                POP             BP
                RET             6
PIXEL           ENDP
SEGCODIGO       ENDS
                END
```

```
;-------------------------------------------------------------------
;       SUBRUTINA EXTERNA --TIMBRE-- PARA SONAR LA CAMPANA
;-------------------------------------------------------------------

;-------------------------------------------------------------------
SEGCODIGO       SEGMENT         PUBLIC          'CODE'
                ASSUME          CS:SEGCODIGO
                PUBLIC          TIMBRE

TIMBRE          PROC            NEAR
                PUSH            BP
                MOV             BP,SP
                PUSH            DS

                MOV             DH,00
                MOV             DL,07
                MOV             BH,00
                MOV             AH,02
                INT             21H

                POP             DS
                MOV             SP,BP
                POP             BP
                RET
TIMBRE          ENDP
SEGCODIGO       ENDS
                END
```

```
;-------------------------------------------------------------------
;       SUBRUTINA EXTERNA -- RECIBA.ASM-- PARA RECEPCION
;            DE 1 CARACTER POR EL PUERTO SERIAL
;-------------------------------------------------------------------

;-------------------------------------------------------------------
SEGCODIGO       SEGMENT         PUBLIC          'CODE'
                ASSUME          CS:SEGCODIGO
                PUBLIC          RECIBA

                OFCAR           EQU     4[BP]
                SBCAR           EQU     6[BP]
                PUERTO          EQU     8[BP]

RECIBA          PROC            NEAR
                PUSH            BP
                MOV             BP,SP
                PUSH            DS
                PUSH            DI

                MOV             DX,PUERTO
                MOV             AH,02
                INT             14H
                SUB             BX,BX
                MOV             BL,AL
                MOV             DS,SBCAR
                MOV             DI,OFCAR
                MOV             DS:[DI],BX

                POP             DI
                POP             DS
                MOV             SP,BP
                POP             BP
                RET             6
RECIBA          ENDP
SEGCODIGO       ENDS
                END
```

```
;-------------------------------------------------------------------
;       SUBRUTINA EXTERNA -- TRANSMI.ASM-- PARA TRANSMISION
;            DE 1 CARACTER POR EL PUERTO SERIAL
;-------------------------------------------------------------------

;-------------------------------------------------------------------
SEGCODIGO       SEGMENT         PUBLIC          'CODE'
                ASSUME          CS:SEGCODIGO
                PUBLIC          TRANSMITA

                OFCAR           EQU     4[BP]
                SBCAR           EQU     6[BP]
                PUERTO          EQU     8[BP]

TRANSMITA       PROC            NEAR
                PUSH            BP
                MOV             BP,SP
                PUSH            DS
                PUSH            DI

                MOV             DS,SBCAR
                MOV             DI,OFCAR
                MOV             AX,DS:[DI]
                MOV             AH,01
                MOV             DX,PUERTO
                INT             14H
                MOV             DS,SBCAR
                MOV             DI,OFCAR
                SUB             BX,BX
                MOV             BL,AH
                MOV             DS:[DI],BX

                POP             DI
                POP             DS
                MOV             SP,BP
                POP             BP
                RET             6
 TRANSMITA      ENDP
 SEGCODIGO      ENDS
                END
```

```
;-------------------------------------------------------------------
;       SUBRUTINA EXTERNA -- ESTATUS.ASM-- PARA VER EL
;                  STATUS DEL PUERTO SERIAL
;-------------------------------------------------------------------

;-------------------------------------------------------------------
SEGCODIGO       SEGMENT         PUBLIC          'CODE'
                ASSUME          CS:SEGCODIGO
                PUBLIC          VERSTATUS

                OFVALOR         EQU     4[BP]
                SBVALOR         EQU     6[BP]
                PUERTO          EQU     8[BP]

VERSTATUS       PROC            NEAR
                PUSH            BP
                MOV             BP,SP
                PUSH            DS
                PUSH            DI

                MOV             DX,PUERTO
                MOV             AH,03
                INT             14H
                AND             AX,0300H
                SUB             BX,BX
                MOV             BL,AH
                MOV             DS,SBVALOR
                MOV             DI,OFVALOR
                MOV             DS:[DI],BX


                POP             DI
                POP             DS
                MOV             SP,BP
                POP             BP
                RET             6
VERSTATUS       ENDP
SEGCODIGO       ENDS
                END
```

```
/% Programa de Presentacion General %/

# include <stdio.h>
# include <dos.h>
# include <conio.h>
# include <bios.h>
# include <fcntl.h>
#include <math.h>
#include <graphics.h>
#include <function.h>

  /%INICIO DEL BLOQUE AUXILIAR%/
typedef struct {
 char horai[8];
 char horaf[8];
 float xbar;
 float med;
 float var;
 } tabla;


void marcas(datoM,n,MFC,MFA)
float datoM[];
int MFC[],MFA[];
int n;
{
 int i,j;
 float lc1,lc2;
 for (i=0;i<=9;++i)
  {
  MFC[i]=0;
  MFA[i]=0;
  }
 lc1=-15.00;
 lc2=0.00;
 for (i=0;i<=8;++i)
  {
  lc1 +=15;
  lc2 +=15;
  for (j=0;j<n;++j)
   {
   if((datoM[j]>=lc1) && (datoM[j] < lc2))
    MFC[i] +=1;
   }
  }
 for (j=0;j<n;++j)
  if (datoM[j] >= lc2)
  {
   MFC[9] +=1;
  }
 lc2=0.00;
 for (i=0;i<=9;++i)
 {
```

```
  lc2 +=15;
  for (j=0;j<n;++j)
   if (datoM[j] < lc2)
       MFA[i] +=1;
 }
return;
}


   /*CALCULO DE LA FRECUENCIAS*/


void muesfrec(frec,freca,n,hisoj)
int frec[],freca[];
FILE *hisoj;
int n;
{
   int i,sumf=0;
   float mlc1,mlc2,p,M=-7.5;
   float porc[10];
   mlc1=-15.00;
   mlc2=0.00;
   printf("\n\n\tDISTRIBUCION DE FRECUENCIA ");
   fprintf(hisoj,"\n\t\tDISTRIBUCION DE FRECUENCIAS CON INT.CLASE = 15\n\n");
   printf("CON UN INTERVALO DE CLASE = 15\n\n");
   fprintf(hisoj,"FRONTERA DE CLASE ¦    FREC ¦ FACUM\t¦ PORCENTAJE\t ¦  MARCA\n\n");
   printf("FRONTERA DE CLASE ¦    FREC ¦ FACUM\t¦ PORCENTAJE\t ¦  MARCA\n\n");
   for (i=0;i<=9;++i)
 {
  p =((float) frec[i]/n);
  porc[i]= p*100;
 }
 for(i=0;i<=9;++i)
 {
  mlc1 +=15;
  mlc2 +=15;
  M +=15;
printf("%3.2f-%3.2f\t ¦\t%2d  ¦\t%2d\t¦   %3.2f\t ¦   %3.1f\n",mlc1,mlc2,frec[i],freca[i],porc[i],M);
fprintf(hisoj,"%3.2f-%3.2f\t ¦\t%2d  ¦\t%2d\t¦   %3.2f\t ¦   %3.1f\n",mlc1,mlc2,frec[i],freca[i],porc[i],M);
    sumf +=frec[i];
 }
   printf("\n");
   fprintf(hisoj,"\n");
   printf("TOTAL DE DATOS =\t%d\t\t    %s\n",sumf,"100%");
   fprintf(hisoj,"TOTAL DE DATOS =\t%d\t\t   %s\n",sumf,"100%");
return;
}


  /*GRAFICO DE EL HISTOGRAMA*/

void histo(frcl,frx)
int frcl[],frx;
```

```
{
int graphdriver=DETECT;
int graphmode,conta=0;
int x,y,k,tp,res=2;
char cam ='y';
do{
 conta+=1;
 x=-20;
 y=0;
 clrscr();
 initgraph(&graphdriver,&graphmode,"c:\\turboc");
 setlinestyle(SOLID_LINE,0xf0f,NORM_WIDTH);
 setviewport(0,0,600,600,1);
 rectangle(100,0,550,300);
 for (k=0;k<=9;k++)
  {
  x +=40;
  if (k==9)
     tp = frx;
  else
     tp = frcl[k];
  y=300-tp*res;
 rectangle(x+100,y,(x+140),300);
 outtextxy(97,y-3,"-");
 outtextxy(x+116,301,"¦");
outtextxy(105,310,"   7.5 22.5 37.5 52.5 67.5 82.5 97.5 112.5 127.7 142.5");
outtextxy(250,getmaxy()-150,"MARCAS DE CLASE");
outtextxy(150,380,"HISTOGRAMA DE LA TEMPERATURA DEL SISTEMA");
  }
outtextxy(150,getmaxy()-50,"DESEA CAMBIAR LA RESOLUCION (y-n)?");
 cam=getche();
 if (cam=='y')
 {
 do{
 cleardevice();
 if (conta==2) break;
 printf("\nELIJA EL TIPO DE RESOLUCION (1,2,3,4,5): ");
 scanf("%d",&res);
   }while((res>=1)&&(res<=5)!=1);
 }
 else
 cleardevice();
}while (((cam!='n')&&(conta<2))==1);
cleardevice();
outtextxy(150,getmaxy()-20,"PRESIONE CUALQUIER TECLA PARA CONTINUAR");
 getche();
closegraph();
return;
}


  /*GRAFICO DEL POLIGONO DE FRECUENCIA*/
```

```
polfrc(y)
int y[];
{
int  graphdriver =DETECT,graphmode,i;
int conta1=0,py,yf[12],m,res=2;
char cam='y';
do{
 conta1+=1;
 initgraph(&graphdriver,&graphmode,"c:\\turboc");
 setlinestyle(SOLID_LINE,0xf0f,NORM_WIDTH);
 yf[0]=0;
 yf[11]=0;
 for (i=1;i<11;++i)
     yf[i] = y[i-1];
 setviewport(0,0,600,600,1);
 rectangle(100,0,600,300);
 m=-40;
 for (i=0;i<=11;++i)
      {
  m +=40;
  outtextxy(97+m,301,"¦");
      }
 m =-40;
 moveto(100,300);
 for (i=0;i<=11;++i)
  {
  m +=40;
  lineto(100+m,300-res*yf[i]);
  outtextxy(94,297-res*yf[i],"-");
  moveto(100+m,300-res*yf[i]);
  }
outtextxy(97,310,"0  7.5 22.5 37.5 52.5 67.5 82.5 97.5 112.5 127.7 142.5 157.5");
outtextxy(250,getmaxy()-150,"MARCAS DE CLASE");
outtextxy(150,380,"POLINOMIO DE FRECUENCIAS DE LA TEMPERATURA");
outtextxy(150,getmaxy()-50,"DESEA CAMBIAR AL RESOLUCION (y-n)?");
 cam=getche();
 if (cam=='y')
 {
 do{
 cleardevice();
 if (conta1==2) break;
 printf("\nELIJA EL TIPO DE RESOLUCION (1,2,3,4,5): ");
 scanf("%d",&res);
   }while((res>=1)&&(res<=5)!=1);
 }
 else
 cleardevice();
}while(((cam!='n')&&(conta1<2))==1);
cleardevice();
outtextxy(150,getmaxy()-20,"PRESIONE CUALQUIER TECLA PARA CONTINUAR");
 getche();
closegraph();
```

```
return;
}


  /%GRAFICO DE LA OJIVA%/

ojiv(y)
int y[];
{
int  graphdriver =DETECT,graphmode,i;
int py,conta2=0,yf[12],m,res=2;
char cam='y';
do{
 conta2+=1;
 initgraph(&graphdriver,&graphmode,"c:\\turboc");
 setlinestyle(SOLID_LINE,0xf0f,NORM_WIDTH);
 yf[0]=0;
 yf[11]=0;
 for (i=1;i<11;++i)
     yf[i] = y[i-1];
 setviewport(0,0,600,600,1);
 rectangle(100,0,600,300);
 m=-40;
 for (i=0;i<11;++i)
      {
  m +=40;
  outtextxy(97+m,301,"¦");
      }
 m =-40;
 moveto(100,300);
 for (i=0;i<11;++i)
  {
  m +=40;
  lineto(100+m,300-res%yf[i]);
  outtextxy(96,297-res%yf[i],"-");
  moveto(100+m,300-res%yf[i]);
  }
outtextxy(97,310,"0   15   30   45   60   75   90   105  120  135  140");
outtextxy(250,getmaxy()-150,"FRONTERAS DE CLASE");
outtextxy(150,380,"OJIVA DE LA TEMPERATURA DEL SISTEMA");
outtextxy(150,getmaxy()-50,"DESEA CAMBIAR LA RESOLUCION (y-n)?");
 cam=getche();
 if (cam=='y')
 {
 do{
 cleardevice();
 if (conta2==2) break;
 printf("\nELIJA EL TIPO DE RESOLUCION (1,2,3,4,5): ");
 scanf("%d",&res);
   }while((res>=1)&&(res<=5)!=1);
 }
 else
 cleardevice();
```

```
}while(((cam!='n')&&(conta2<2))==1);
cleardevice();
outtextxy(150,getmaxy()-20,"PRESIONE CUALQUIER TECLA PARA CONTINUAR");
 getche();
closegraph();
return;
}

   /*GRAFICO DE LOS PUNTOS EXPERIMENTALES DE LA APROXIMACION POLINOMICA*/

grafaprox(dexp,yexp,ne)
float dexp[],yexp[];
int ne;
{
float maxdexp,maxyexp,res=2;
int graphdriver= DETECT,graphmode,i,numpt=0,xscale,yscale;
 initgraph(&graphdriver,&graphmode,"c:\\turboc");
 setlinestyle(SOLID_LINE,0xf0f,NORM_WIDTH);
 maxdexp=maximo(dexp,ne);
 maxyexp=maximo(yexp,ne);
 yscale=300/(res*maxyexp);
 xscale=500/(res*maxdexp);
 setviewport(0,0,600,600,1);
 rectangle(100,0,600,300);
       numpt=ne;
 moveto(100,300);
    for (i=0;i<numpt;++i)
    {
    lineto(100+xscale*dexp[i],300-yscale*yexp[i]);
    outtextxy(96,298-yscale*yexp[i],"-");
    outtextxy(97+xscale*dexp[i],301,"!");
    moveto(100+xscale*dexp[i],300-yscale*yexp[i]);
    }
 outtextxy(150,380,"GRAFICO DE LOS PUNTOS EXPERIMENTALES");
     outtextxy(150,getmaxy()-20,"PRESIONE CUALQUIER TECLA PARA CONTINUAR");
 getche();
     closegraph();
return;
}

 /*SUBRUTINA DE LOS GRAFICOS ESTADISTICOS*/

void histojiv(hisoj)
FILE *hisoj;
{
 FILE *lectura,*fopen();
 float dato[500];
 char horai[8],horaf[8],hora[8];
 int fc[12],fa[12],fx;
 int n=0,i;
 char c;
 hisoj = fopen("A:\HISTOJIV.R","w+");
 printf("\n\tDISTRIBUCIONES DE FRECUENCIA\n\n");
```

```
 fprintf(hisoj,"\t\tDISTRIBUCIONES DE FRECUENCIA\n\n");
printf("INGRESE EL INTERVALO DE TIEMPO DE LA MUESTRA\n\n");
rango(horai,horaf);
 if((lectura=fopen("a:\Tabla.dbf","r"))==NULL)
 {
 printf("archivo vacio-fallo apertura");
 exit(0);
 }
do{
  fscanf(lectura,"%s%f",hora,&dato[n]);
  }while(strcmp(hora,horai)!=0);
do{
   n+=1;
   fscanf(lectura,"%s%f",hora,&dato[n]);
   }while(strcmp(hora,horaf)!=0);
fprintf(hisoj,"\n\tDATOS DEL INTERVALO DE TIEMPO TOMADO\n\n");
 for (i=0;i<n;++i)
  {
  fprintf(hisoj,"X[%d]=%3.1f ",i,dato[i]);
     if (i%5==0)
   fprintf(hisoj,"\n");
  }
 fprintf(hisoj,"\n\n");
printf("\n");
 fprintf(hisoj,"RANGO DE LA MUESTRA :%s - %s\n",horai,horaf);
 fprintf(hisoj,"NUMERO DE DATOS :%d\n\n",n);
 marcas(dato,n,fc,fa);
 clrscr();
 muesfrec(fc,fa,n,hisoj);
printf("\nSI NO DESEA VER EL HISTOGRAMA Y LA OJIVA PRESIONE @\n");
printf("PARA CONTINUAR PRESIONE CUALQUIER OTRA TECLA\n");
 c=getche();
 if (c!='@')
 {
    fx = fc[9];
    histo(fc,fx);
    polfrc(fc);
    ojiv(fa);
 }
printf("\n\nLOS DATOS PUEDEN SER OBSERVADOS EN EL ARCHIVO HISTOJIV.R\n\n");
return;
}



 /*ESTO CALCULA LAS SUMAS PARCIALES DE Xi y Yi*/

void sumas(ndatos,grado,xdat,ydat,Ex,Eyx)
int ndatos,grado;
float xdat[],ydat[],Ex[],Eyx[];
{
 int i,p;
 float k=0;
```

```
 for (p=0;p<=2*grado;++p)
  Ex[p] =0;
 for (p=0;p<=grado;++p)
  Eyx[p] =0;
  for (p=0;p<=2*grado;++p)
   for(i=0;i<ndatos;++i)
    {
    k = potencia(p,xdat[i]);
    Ex[p] += k;
    if (p<= grado)
     Eyx[p] += ydat[i]*k;
    }
 return;
}



 /*ESTA PARTE OBTIENE LOS COEFICIENTES DEL POLINOMIO*/

void resolvedor(matx,nx,cor,appol)
float matx[][6];
FILE *appol;
float cor[];
int nx;
{
int j,i,k;
float suma;
cor[nx]=(matx[nx][nx+1]/matx[nx][nx]);
for (i=nx-1;i>=0;--i)
 {
 suma=0;
 for (j=nx;j>i;--j)
  suma +=cor[j] * matx[i][j];
 cor[i]=(matx[i][nx+1]-suma)/matx[i][i];
 }
printf("\n");
fprintf(appol,"\n\t\tPOLINOMIO DE APROXIMACION\n\n");
printf("\n\nEL POLINOMIO DE APROXIMACION ES EL SIGUIENTE :\n\n");
for(i=0;i<=nx;++i)
{
printf("+(%4.4f)*X^%d",cor[i],i);
fprintf(appol,"+(%4.4f)*X^%d",cor[i],i);
}
printf("\n\n");
fprintf(appol,"\n\n");
return;
}

   /* EVALUACION POLINOMICA */

int evaluador(cov,nv,appol,dexp,yexp)
float dexp[],yexp[];
float cov[];
```

```
FILE *appol;
int nv;
{
int nd,i,j;
float k,en;
 do{
     printf ("\nINGRESE EL NUMERO DE DATOS A SER EVALUADOS =");
     scanf("%d",&nd);
     printf("\n");
     if (nd<0)
     printf("\nEL NUMERO DE DATOS DEBEN SER MAYOR A 0\n");
 }while(nd<0);
printf("INGRESE LOS DATOS A EVALUAR\n\n");
for(i=0;i<nd;++i)
 {
 printf("X[%d] =",i+1);
 scanf("%f",&dexp[i]);

 }
for(i=0;i<nd;++i)
 yexp[i]=0;
for(i=0;i<nd;++i)
 for (j=0;j<=nv;++j)
  {
  k=potencia(j,dexp[i]);
  yexp[i]+=cov[j]*k;
  }
printf("\nLOS DATOS EVALUADOS APROXIMADAMENTE SON :\n");
fprintf(appol,"\tLOS DATOS EVALUADOS APROXIMADAMENTE SON :");
fprintf(appol,"\n\n");
fprintf(appol," X[i]\t  Y[i]\n");
for(i=0;i<nd;++i)
 {
 printf("%10.2f\t%10.2f\n",dexp[i],yexp[i]);
 fprintf(appol,"%10.2f\t%10.2f\n",dexp[i],yexp[i]);
 }
return(nd);
}



   /*MAIN DE LA APROX. POLINOMICA*/

void aproxpol(appol)
FILE *appol;
{
float dexp[50],yexp[50];
char ok='@',o='@',g;
int M=0,n=0,d,ne;
float x[100],y[100],sumax[10],sumayx[10],cop[5];
float aum[10][10],A[5][6];
while (o=='@')
  {
```

```
 printf("\nSI DESEA CONTINUAR PRESIONE @ SINO CUALQUIER TECLA\n");
  o=getche();
  }
printf("\n\nLOS DATOS PUEDEN SER OBSERVADOS EN EL ARCHIVO APROXPOL.R\n\n");
return;
}




  /*ESTO CALCULA LA MEDIA Y LA MEDIANA */

void proceso(x,y,z,p,sump,scp)
int p;
float *x,*y,*sump,*scp;
float z[];
{
 int i;
 float suma,sumcuad,a,b;
 suma=0;
 sumcuad=0;
 for(i=0;i<p;++i)
  {
  suma += z[i];
  sumcuad +=z[i]*z[i];
  }
 *x=suma/p;                /*MEDIA*/

 *sump=suma;
 *scp=sumcuad;
 if(p%2==0)                 /*MEDIANA*/
 {
 a=z[p/2];
 b=z[(p+2)/2];
 *y=(a +b)/2;
 }
 else
  *y=z[(p+1)/2];
 return;
}

  /*ESTE PROCEDIMIENTO CALCULA LA DESVIACION ESTANDAR*/

 float variaz(k,sumpv,scpv)
 float *sumpv,*scpv;
 int k;
 {
 float sp,s1,sc1,parc;
  s1=*sumpv;
  sc1=*scpv;
  parc=((k*sc1)-(s1*s1))/(k*(k-1));
```

```
 sp=sqrt(parc);
 return(sp);
}


 /*GRAFICO DE LOS DATOS DEL SISTEMA*/


void datostotal(datol)
FILE *datol;
{
FILE *xavier;
tabla muestra[100];
FILE *lectura;
float ptemp[20];
float sum=0,media,mediana,s,sc=0;
char hora[8],horai[8]="08:00",horaf[8];
int i,n,h,j=0;
printf("\n\n\t\t\tESPERE UN MOMENTO\n");
 datol=fopen("A:\EVALDIA.R","w+");
 xavier=fopen("A:\media.dta","w+");
 if((lectura=fopen("a:\Tabla.dbf","r"))==NULL)
 {
 printf("archivo vacio-fallo apertura");
 exit(0);
 }
while(j<14)
{
      n=0;
      do{
    fscanf(lectura,"%s%f",hora,&ptemp[n]);
   }while(strcmp(hora,horai)!=0);
 strcpy(muestra[j].horai,hora);
 for(i=0;i<12;++i)
    {
    n+=1;
    fscanf(lectura,"%s%f",hora,&ptemp[n]);
    strcpy(horai,hora);
    }
 ordena(ptemp,n);
 proceso(&media,&mediana,ptemp,n,&sum,&sc);
 s=variaz(n,&sum,&sc);        /*MUESTRA DATOS*/
 muestra[j].xbar=media;
      muestra[j].med=mediana;
 muestra[j].var=s;
       rewind(lectura);
   j+=1;
  }
    clrscr();
  printf("\n\nHORA\tMEDIA  MEDIANA\tDES.ESTD\n\n");
 fprintf(datol,"\n\nHORA\tMEDIA  MEDIANA\tDES.ESTD\n\n");
  for (i=0,h=8;i<j;++i,h+=1)
  {
```

```
printf("%s\t%3.2f\t%3.2f\t%3.2f\n",muestra[i].horai,muestra[i].xbar,muestra[i].med,muestra[i].var);
fprintf(datol,"%s\t%3.2f\t%3.2f\t%3.2f\n",muestra[i].horai,muestra[i].xbar,muestra[i].med,muestra[i].var);
fprintf(xavier,"%d\t%3.2f\n",h,muestra[i].xbar);
  }
fclose(lectura);
fclose(xavier);
printf("\n\nLOS DATOS PUEDEN SER OBSERVADOS EN EL ARCHIVO EVALDIA.R\n\n");
return;
}



/*ESTA PARTE HACE LA ADQUICICION DE LOS DATOS DE LA BASE*/

void adqmuestra(tprom)
FILE *tprom;
{
     tabla muestra[100];
     FILE *lectura,*fopen();
 int nmuestra;
 float ptemp[500];
 char hora[8];
 char nm;
 int i=0,j=0,n;
 float sum=0,media,mediana,s,sc=0;
 tprom=fopen("a:\ESTIMADORES.R","w+");
 do{
    clrscr();
    fprintf(tprom,"\n\t\tESTIMADORES DE LAS MUESTRAS DEL PROCESO\n\n");
    printf("\nINGRESE EL NUMERO DE MUESTRAS A TOMAR (MAX=5):");
    scanf("%d",&nmuestra);
 }while(((nmuestra>=1)&&(nmuestra<=5))!=1);
 fprintf(tprom,"NUMERO TOTAL DE MUESTRAS : %d\n\n",nmuestra);
 if((lectura=fopen("a:\Tabla.dbf","r"))==NULL)
  {
  printf("archivo vacio-fallo apertura");
  exit(0);
  }
for(j=0; j<nmuestra; j++)
   {
do{
   n=0;
    printf("\nINGRESE EL RANGO DE LA MUESTRA %d\n",j+1);
    printf("\n VALORES ENTRE 08:00 Y 22:00 SOLAMENTE\n\n");
    rango(muestra[j].horai,muestra[j].horaf);
 do{
    fscanf(lectura,"%s%f",hora,&ptemp[n]);
   }while(strcmp(hora,muestra[j].horai)!=0);
 do{
   n+=1;
   fscanf(lectura,"%s%f",hora,&ptemp[n]);
   }while(strcmp(hora,muestra[j].horaf)!=0);
```

```
  if (n==1)
      {
printf("\nTIENEN QUE EXISTIR MAS DE UN DATO\n");
rewind(lectura);
      }
 }while(n==1);
ordena(ptemp,n);
proceso(&media,&mediana,ptemp,n,&sum,&sc);
s=variaz(n,&sum,&sc);        /*MUESTRA DATOS*/
muestra[j].xbar=media;
     muestra[j].med=mediana;
muestra[j].var=s;
rewind(lectura);
  }
  /*ESTA PARTE MUESTRA LOS DATOS*/
 clrscr();
 printf("\n MEDIA\t\tMEDIANA\t\tDES.ESTD\t   RANGO\n\n");
 fprintf(tprom,"MEDIA\t\tMEDIANA\t\tDES.ESTD\t   RANGO\n\n");
 for (i=0;i<nmuestra;++i)
 {
printf("%f\t%f\t%f\t",muestra[i].xbar,muestra[i].med,muestra[i].var);
fprintf(tprom,"%f\t%f\t%f\t",muestra[i].xbar,muestra[i].med,muestra[i].var);
 printf("%s",muestra[i].horai);
 printf(" - ");
 printf("%s",muestra[i].horaf);
 printf("\n");
 fprintf(tprom,"%s",muestra[i].horai);
 fprintf(tprom," - ");
 fprintf(tprom,"%s",muestra[i].horaf);
 fprintf(tprom,"\n");
 }
 fclose(lectura);
printf("\n\nLOS DATOS LOS PUEDE OBSERVAR EN EL ARCHIVO PROMEDIO.R\n\n");
return;
}


void continua()
{
 printf("\n\tPRESIONE CUALQUIER TECLA PARA CONTINUAR\n");
 getch();
 return;
}


void menu2()
{
 clrscr();
 printf("\n\t\tESCOJA UNA DE LAS OPCIONES\n\n");
 printf("1.-TABLA DE MEDIA,MEDIANA Y VARIANZA.\n\n");
 printf("2.-APROXIMACION POLINOMIAL.\n\n");
 printf("3.-HISTOGRAMA - OJIVA.\n\n");
 printf("4.-EVALUACION DEL DIA.\n\n");
```

```
 printf("5.-FIN DE EJECUCION\n\n");
return;
}


void calculo()
{
char opcion;
FILE *appol1=NULL,*hisoj1=NULL,*prom=NULL,*datol=NULL;
 clrscr();
 do
 {
 menu2();
 printf("OPCION =");
 opcion=getche();
 clrscr();
 switch (opcion)
  {
 case '1': { adqmuestra(prom);
  fclose(prom);

  continua();
   break;}
 case '2':{ aproxpol(appol1);
  fclose(appol1);
  continua();
   break;}
 case '3': { histojiv(hisoj1);
  fclose(hisoj1);
  continua();
   break;}
 case '4':{ datostotal(datol);
  fclose(datol);
  continua();
    break;}
 case '5': break;
 default : break;
  }
 }
 while (opcion!='5');
return;
}


/*******************    BLOQUE  PRINCIPAL ***********************************/


main()

{

 int r,a,j;
```

```
int radius,angle;
int h,t;
int i;
char c,caso,opcion;
char command[80];
      .

/* bloque principal */
  clrscr();
  opcion='0';
  while (1){
clrscr();
presenta();
opcion=getche();
switch(opcion){
 case '1': /* guarda datos en tabla */
  clrscr();
             printf("\n Archivando datos.. Espere.. ");
  otros();
  printf("\n Datos ingresados en tabla..");
  printf("\n Presione < ENTER > para seguir...");
  scanf("%c");
  break;
 case '2':  /* mostrar datos recibidos e archivarlos*/
  clrscr();
  mostrar();
  break;
 case '3': /* Realizar calculos estadisticos y procesos */
  calculo();
  printf("\n Digite <ENTER> para seguir...");
  scanf("%c%c");
  break;
 case '4': /* Imprimir Reporte del Dia */
  clrscr();
  reportar();
  printf("\n Digite <ENTER> para ir al menu...");
  scanf("%c");
  break;
 case '5':/* salida del sistema */
        clrscr();
        exit();
        break;
 default : break;

  }
     }
   clrscr();

}

/*********************   FIN DE BLOQUE PRINCIPAL ******************************/
```

```
/************************  GRUPO DE FUNCIONES XGR ****************************/

/*  PRESENTA():  MUESTRA EL MENU DE PRESENTACION */

presenta()

{
struct date today;

getdate(&today);
gotoxy(70,1);
printf("%d/%d/%d\n",today.da_mon,today.da_day,today.da_year);
gotoxy(1,3);
printf("           ESCUELA SUPERIOR POLITECNICA DEL LITORAL \n  ");
printf("\n                    CENTRAL DE DATOS \n ");
printf("\n\n                      MENU PRINCIPAL         ");
printf("\n\n      1.- INSERTAR DATOS EN TABLA ");
printf("\n\n                      2.- MOSTRAR DATOS LLEGADOS ");
printf("\n\n                      3.- PROCESAMIENTO DE DATOS    ");
printf("\n\n                      4.- IMPRIMIR REPORTE ");
printf("\n\n                      5.- SALIR AL SISTEMA     ");

   printf("\n\n\n\n%s",
 "                  DIGITE LA OPCION DESEADA---> ");
   return(0);
}

/*    FIN DE PRESENTA() */



/* MOSTRAR: VISUALIZA LOS DATOS LLEGADOS DESDE LA PLANTA */

mostrar()

{
    struct date hoy;
    int  i,proceso,status;
    char  hora[10];
    char temperatura[10],c;

    FILE *f1,*fopen();

    clrscr();
    getdate(&hoy);
    gotoxy(70,1);
    printf("%d/%d/%d\n",hoy.da_mon,hoy.da_day,hoy.da_year);
    f1= fopen("temphora.dat","r");
    if (f1==NULL)
  printf("No existe archivo de datos de Planta \n");
    else {
   printf("\n             HORA        TEMPERATURA   \n\n");
   for(i=1;i<=12;i++){
```

```
   fscanf(f1,"%s  %s\n",hora,temperatura);
   printf("            %s             %s\n",hora,temperatura);
   }
   fclose(f1);
    }
    c='a';
    while (c!='@'){
  gotoxy(1,20);
  printf("Digite < @ > para salir..");
  c=getchar();
  }
    return(0);
}

/*    FIN DE MOSTRAR()  */


/*  REPORTAR(): DA LOS VALORES MEDIOS DE TEMPERATURA POR HORA Y GRAFICA*/
reportar()
{

    int graphdriver=DETECT;
    int graphmode;
    int  i;

    int  hora1,angle[15],ag;
    float temp_pro,radio[15],ra;
    FILE *f3,*fopen();


    f3= fopen("a:\media.dta","r");
    if (f3==NULL)
 printf("fallo el fopen \n");
    else {
  printf("\n             HORA        TEMP_PROMEDIO   \n\n");
  for(i=1;i<=15;i++){
   fscanf(f3,"%d  %f\n",&hora1,&temp_pro);
   printf("            %d             %f\n",hora1,temp_pro);
   angle[i]=hora1;
   radio[i]=temp_pro;
   }
   fclose(f3);
    }
    printf("\n Digite <ENTER> para visualizar...");
    scanf("%c%c");

    initgraph(&graphdriver,&graphmode,"C:\\TURBOC ");
    setlinestyle(DOTTED_LINE,0xf0f,NORM_WIDTH);
    for(i=0;i<461;i+=10){
    moveto(0,i);
    lineto(800,i);
     }
   for (i=0;i<=800;i+=10){
```

```
  moveto(i,0);
  lineto(i,460);
  }

   for(i=1;i<=15;i++){
   ag=angle[i]%15;
   ra=radio[i]%2;
   arc(300,230,0,ag,ra);
   }
   outtextxy(190,460,"Presione < ENTER > para salir...");
   scanf("%c");
   closegraph();
   printf("\n");
   return(0);
}

/% FIN DE REPORTAR() %/


/% OTROS(): SI STATUS=1 ANADE DATOS A LA TABLA %/

 otros()
 {
     char hora[10],temperatura[10];
     FILE %f1,%f2,%fopen();
     int i;

  f1= fopen("temphora.dat","r");
  f2= fopen("Tabla.dbf","a+");
    if (f1==NULL){
 printf("No existe archivo de datos de Planta \n");
 printf("Datos no ingresados\n");
 }
    else {
   for(i=1;i<=12;i++){
   fscanf(f1,"%s  %s\n",hora,temperatura);
   fprintf(f2,"%s  %s\n",hora,temperatura);
   }
   fclose(f1);
   fclose(f2);
    }
    printf("\n");
    return(0);
 }
 /% FIN DE OTRO() %/


 /************** FIN DE FUNCIONES XGR  ***********************************/
```

```
#include <io.h>         /*  PROGRAMA DE CONTROL ESTADISTICO */
#include <stdio.h>
#include <math.h>
#include <conio.h>
#include <graphics.h>
#include <function.h>

typedef struct {
 char horai[8];
 char horaf[8];
 float xbar;
 float med;
 float var;
 } tabla;


void marcas(datoM,n,MFC,MFA)
float datoM[];
int MFC[],MFA[];
int n;
{
 int i,j;
 float lc1,lc2;
 for (i=0;i<=9;++i)
  {
  MFC[i]=0;
  MFA[i]=0;
  }
 lc1=-15.00;
 lc2=0.00;
 for (i=0;i<=8;++i)
  {
  lc1 +=15;
  lc2 +=15;
  for (j=0;j<n;++j)
   {
   if((datoM[j]>=lc1) && (datoM[j] < lc2))
    MFC[i] +=1;
   }
  }
 for (j=0;j<n;++j)
  if (datoM[j] >= lc2)
  {
   MFC[9] +=1;
  }
 lc2=0.00;
 for (i=0;i<=9;++i)
 {
  lc2 +=15;
  for (j=0;j<n;++j)
   if (datoM[j] < lc2)
       MFA[i] +=1;
 }
```

```
return;
}


void muesfrec(frec,freca,n,hisoj)
int frec[],freca[];
FILE *hisoj;
int n;
{
   int i,sumf=0;
   float mlc1,mlc2,p,M=-7.5;
   float porc[10];
   mlc1=-15.00;
   mlc2=0.00;
   printf("\n\n\tDISTRIBUCION DE FRECUENCIA ");
   fprintf(hisoj,"\n\t\tDISTRIBUCION DE FRECUENCIAS CON INT.CLASE = 15\n\n");
   printf("CON UN INTERVALO DE CLASE = 15\n\n");
   fprintf(hisoj,"FRONTERA DE CLASE ¦    FREC ¦ FACUM\t¦ PORCENTAJE\t ¦  MARCA\n\n");
   printf("FRONTERA DE CLASE ¦    FREC ¦ FACUM\t¦ PORCENTAJE\t ¦  MARCA\n\n");
   for (i=0;i<=9;++i)
 {
  p =((float) frec[i]/n);
  porc[i]= p*100;
 }
 for(i=0;i<=9;++i)
 {
  mlc1 +=15;
  mlc2 +=15;
  M +=15;
printf("%3.2f-%3.2f\t  ¦\t%2d  ¦\t%2d\t¦   %3.2f\t ¦   %3.1f\n",mlc1,mlc2,frec[i],freca[i],porc[i],M);
fprintf(hisoj,"%3.2f-%3.2f\t  ¦\t%2d  ¦\t%2d\t¦   %3.2f\t ¦   %3.1f\n",mlc1,mlc2,frec[i],freca[i],porc[i],M);
    sumf +=frec[i];
 }
   printf("\n");
   fprintf(hisoj,"\n");
   printf("TOTAL DE DATOS =\t%d\t\t    %s\n",sumf,"100%");
   fprintf(hisoj,"TOTAL DE DATOS =\t%d\t\t   %s\n",sumf,"100%");
return;
}


void histo(frcl,frx)
int frcl[],frx;
{
int graphdriver=DETECT;
int graphmode;
int x,y,k,tp,res=2;
char cam ='y';
do{
 x=-20;
 y=0;
 clrscr();
 initgraph(&graphdriver,&graphmode,"c:\\turboc");
```

```
 setlinestyle(SOLID_LINE,0xf0f,NORM_WIDTH);
 setviewport(0,0,600,600,1);
 rectangle(100,0,550,300);
 for (k=0;k<=9;k++)
  {
  x +=40;
  if (k==9)
     tp = frx;
  else
     tp = frcl[k];
  y=300-tp*res;
 rectangle(x+100,y,(x+140),300);
 outtextxy(97,y-3,"-");
 outtextxy(x+116,301,"¦");
outtextxy(105,310,"   7.5 22.5 37.5 52.5 67.5 82.5 97.5 112.5 127.7 142.5");
outtextxy(250,getmaxy()-150,"MARCAS DE CLASE");
outtextxy(150,380,"HISTOGRAMA DE LA TEMPERATURA DEL SISTEMA");
  }
outtextxy(150,getmaxy()-50,"DESEA CAMBIAR LA RESOLUCION (y-n)?");
 cam=getche();
 if (cam=='y')
 {
 do{
       cleardevice();
 printf("\nELIJA EL TIPO DE RESOLUCION (1,2,3,4,5): ");
 scanf("%d",&res);
   }while((res>=1)&&(res<=5)!=1);
 }
 else
 cleardevice();
}while(cam!='n');
outtextxy(150,getmaxy()-20,"PRESIONE CUALQUIER TECLA PARA CONTINUAR");
 getche();
closegraph();
return;
}



polfrc(y)
int y[];
{
int  graphdriver =DETECT,graphmode,i;
int py,yf[12],m,res=2;
char cam='y';
do{
 initgraph(&graphdriver,&graphmode,"c:\\turboc");
 setlinestyle(SOLID_LINE,0xf0f,NORM_WIDTH);
 yf[0]=0;
 yf[11]=0;
 for (i=1;i<11;++i)
     yf[i] = y[i-1];
 setviewport(0,0,600,600,1);
```

```
 rectangle(100,0,600,300);
 m=-40;
 for (i=0;i<=11;++i)
      {
  m +=40;
  outtextxy(97+m,301,"|");
      }
 m =-40;
 moveto(100,300);
 for (i=0;i<=11;++i)
  {
  m +=40;
  lineto(100+m,300-res*yf[i]);
  outtextxy(94,297-res*yf[i],"-");
  moveto(100+m,300-res*yf[i]);
  }
outtextxy(97,310,"0  7.5 22.5 37.5 52.5 67.5 82.5 97.5 112.5 127.7 142.5 157.5");
outtextxy(250,getmaxy()-150,"MARCAS DE CLASE");
outtextxy(150,380,"POLINOMIO DE FRECUENCIAS DE LA TEMPERATURA");
outtextxy(150,getmaxy()-50,"DESEA CAMBIAR AL RESOLUCION (y-n)?");
 cam=getche();
 if (cam=='y')
 {
 do{
 cleardevice();
 printf("\nELIJA EL TIPO DE RESOLUCION (1,2,3,4,5): ");
 scanf("%d",&res);
   }while((res>=1)&&(res<=5)!=1);
 }
 else
 cleardevice();
}while(cam!='n');
outtextxy(150,getmaxy()-20,"PRESIONE CUALQUIER TECLA PARA CONTINUAR");
 getche();
closegraph();
return;
}


ojiv(y)
int y[];
{
int  graphdriver =DETECT,graphmode,i;
int py,yf[12],m,res=2;
char cam='y';
do{
 initgraph(&graphdriver,&graphmode,"c:\\turboc");
 setlinestyle(SOLID_LINE,0xf0f,NORM_WIDTH);
 yf[0]=0;
 yf[11]=0;
 for (i=1;i<11;++i)
     yf[i] = y[i-1];
 setviewport(0,0,600,600,1);
```

```
 rectangle(100,0,600,300);
 m=-40;
 for (i=0;i<11;++i)
      {
  m +=40;
  outtextxy(97+m,301,"|");
      }
 m =-40;
 moveto(100,300);
 for (i=0;i<11;++i)
  {
  m +=40;
  lineto(100+m,300-res*yf[i]);
  outtextxy(96,297-res*yf[i],"-");
  moveto(100+m,300-res*yf[i]);
  }
outtextxy(97,310,"0   15   30   45   60   75   90   105  120  135  140");
outtextxy(250,getmaxy()-150,"FRONTERAS DE CLASE");
outtextxy(150,380,"OJIVA DE LA TEMPERATURA DEL SISTEMA");
outtextxy(150,getmaxy()-50,"DESEA CAMBIAR LA RESOLUCION (y-n)?");
 cam=getche();
 if (cam=='y')
 {
 do{
 cleardevice();
 printf("\nELIJA EL TIPO DE RESOLUCION (1,2,3,4,5): ");
 scanf("%d",&res);
   }while((res>=1)&&(res<=5)!=1);
 }
 else
 cleardevice();
}while(cam!='n');
outtextxy(150,getmaxy()-20,"PRESIONE CUALQUIER TECLA PARA CONTINUAR");
 getche();
closegraph();
return;
}



grafaprox(dexp,yexp,ne)
float dexp[],yexp[];
int ne;
{
float maxdexp,maxyexp,res=2;
int graphdriver= DETECT,graphmode,i,numpt=0,xscale,yscale;
 initgraph(&graphdriver,&graphmode,"c:\\turboc");
 setlinestyle(SOLID_LINE,0xf0f,NORM_WIDTH);
 maxdexp=maximo(dexp,ne);
 maxyexp=maximo(yexp,ne);
 yscale=300/(res*maxyexp);
 xscale=500/(res*maxdexp);
 setviewport(0,0,600,600,1);
```

```
 rectangle(100,0,600,300);
       numpt=ne;
 moveto(100,300);
    for (i=0;i<numpt;++i)
    {
    lineto(100+xscale*dexp[i],300-yscale*yexp[i]);
    outtextxy(96,298-yscale*yexp[i],"-");
    outtextxy(97+xscale*dexp[i],301,"¦");
    moveto(100+xscale*dexp[i],300-yscale*yexp[i]);
    }
 outtextxy(150,380,"GRAFICO DE LOS PUNTOS EXPERIMENTALES");
     outtextxy(150,getmaxy()-20,"PRESIONE CUALQUIER TECLA PARA CONTINUAR");
 getche();
     closegraph();
return;
}


void histojiv(hisoj)
FILE *hisoj;
{
 FILE *lectura,*fopen();
 float dato[500];
 char horai[8],horaf[8],hora[8];
 int fc[12],fa[12],fx;
 int n=0,i;
 char c;
 hisoj = fopen("A:\HISTOJIV.R","w+");
 printf("\n\tDISTRIBUCIONES DE FRECUENCIA\n\n");
 fprintf(hisoj,"\t\tDISTRIBUCIONES DE FRECUENCIA\n\n");
printf("INGRESE EL INTERVALO DE TIEMPO DE LA MUESTRA\n\n");
rango(horai,horaf);
 if((lectura=fopen("a:\TEMPHORA.DAT","r"))==NULL)
 {
 printf("archivo vacio-fallo apertura");
 exit(0);
 }
do{
  fscanf(lectura,"%s%f",hora,&dato[n]);
  }while(strcmp(hora,horai)!=0);
do{
   n+=1;
   fscanf(lectura,"%s%f",hora,&dato[n]);
   }while(strcmp(hora,horaf)!=0);
fprintf(hisoj,"\n\tDATOS DEL INTERVALO DE TIEMPO TOMADO\n\n");
 for (i=0;i<n;++i)
  {
  fprintf(hisoj,"X[%d]=%3.1f ",i,dato[i]);
     if (i%5==0)
   fprintf(hisoj,"\n");
  }
 fprintf(hisoj,"\n\n");
printf("\n");
```

```
 fprintf(hisoj,"RANGO DE LA MUESTRA :%s - %s\n",horai,horaf);
 fprintf(hisoj,"NUMERO DE DATOS :%d\n\n",n);
 marcas(dato,n,fc,fa);
 clrscr();
 muesfrec(fc,fa,n,hisoj);
printf("\nSI NO DESEA VER EL HISTOGRAMA Y LA OJIVA PRESIONE @\n");
printf("PARA CONTINUAR PRESIONE CUALQUIER OTRA TECLA\n");
 c=getche();
 if (c!='@')
 {
    fx = fc[9];
    histo(fc,fx);
    polfrc(fc);
    ojiv(fa);
 }
printf("\n\nLOS DATOS PUEDEN SER OBSERVADOS EN EL ARCHIVO HISTOJIV.R\n\n");
return;
}



 /*ESTO CALCULA LAS SUMAS PARCIALES DE Xi y Yi*/

void sumas(ndatos,grado,xdat,ydat,Ex,Eyx)
int ndatos,grado;
float xdat[],ydat[],Ex[],Eyx[];
{
 int i,p;
 float k=0;
 for (p=0;p<=2*grado;++p)
  Ex[p] =0;
 for (p=0;p<=grado;++p)
  Eyx[p] =0;
  for (p=0;p<=2*grado;++p)
   for(i=0;i<ndatos;++i)
    {
    k = potencia(p,xdat[i]);
    Ex[p] += k;
    if (p<= grado)
     Eyx[p] += ydat[i]*k;
    }
 return;
}



 /*ESTA PARTE OBTIENE LOS COEFICIENTES DEL POLINOMIO*/

void resolvedor(matx,nx,cor,appol)
float matx[][6];
FILE *appol;
float cor[];
int nx;
```

```
{
int j,i,k;
float suma;
cor[nx]=(matx[nx][nx+1]/matx[nx][nx]);
for (i=nx-1;i>=0;--i)
 {
 suma=0;
 for (j=nx;j>i;--j)
  suma +=cor[j] * matx[i][j];
 cor[i]=(matx[i][nx+1]-suma)/matx[i][i];
 }
printf("\n");
fprintf(appol,"\n\t\tPOLINOMIO DE APROXIMACION\n\n");
printf("\n\nEL POLINOMIO DE APROXIMACION ES EL SIGUIENTE :\n\n");
for(i=0;i<=nx;++i)
{
printf("+(%4.4f)%X^%d",cor[i],i);
fprintf(appol,"+(%4.4f)%X^%d",cor[i],i);
}
printf("\n\n");
fprintf(appol,"\n\n");
return;
}

   /* EVALUACION POLINOMICA */

int evaluador(cov,nv,appol,dexp,yexp)
float dexp[],yexp[];
float cov[];
FILE *appol;
int nv;
{
int nd,i,j;
float k,en;
 do{
     printf ("\nINGRESE EL NUMERO DE DATOS A SER EVALUADOS =");
     scanf("%d",&nd);
     printf("\n");
     if (nd<0)
     printf("\nEL NUMERO DE DATOS DEBEN SER MAYOR A 0\n");
 }while(nd<0);
printf("INGRESE LOS DATOS A EVALUAR\n\n");
for(i=0;i<nd;++i)
 {
 printf("X[%d] =",i+1);
 scanf("%f",&dexp[i]);

 }
for(i=0;i<nd;++i)
 yexp[i]=0;
for(i=0;i<nd;++i)
 for (j=0;j<=nv;++j)
  {
```

```
  k=potencia(j,dexp[i]);
  yexp[i]+=cov[j]*k;
  }
printf("\nLOS DATOS EVALUADOS APROXIMADAMENTE SON :\n");
fprintf(appol,"\tLOS DATOS EVALUADOS APROXIMADAMENTE SON :");
fprintf(appol,"\n\n");
fprintf(appol," X[i]\t  Y[i]\n");
for(i=0;i<nd;++i)
 {
 printf("%10.2f\t%10.2f\n",dexp[i],yexp[i]);
 fprintf(appol,"%10.2f\t%10.2f\n",dexp[i],yexp[i]);
 }
return(nd);
}



   /*MAIN DE LA APROX. POLINOMICA*/

void aproxpol(appol)
FILE *appol;
{
float dexp[50],yexp[50];
char ok='@',o='@',g;
int M=0,n=0,d,ne;
float x[100],y[100],sumax[10],sumayx[10],cop[5];
float aum[10][10],A[5][6];
while (o=='@')
  {
 appol=fopen("A:\APROXPOL.R","w+");
 clrscr();
 ok='@';
printf("\n\t\tAPROXIMACION POLINOMIAL POR MINIMOS CUADRADOS\n");
 while (ok=='@')
 {
 do{
 printf("\nINGRESE EL NUMERO DE DATOS = ");
 scanf("%d",&M);
 if (M<=2)
  printf("\nEL NUMERO DE DATOS TIENE QUE SER MAYOR A 2\n");
 }while(M<=2);
 do{
printf("INGRESE EL GRADO DEL POLINOMIO DE APROXIMACION = ");
 scanf("%d",&n);
 if((n<0)||(n>=5))
  printf("\nEL GRADO TIENE QUE SER MAYOR A 0 y MENOR A 5\n");
  }while((n<=0)||(n>=5));
fprintf(appol,"\n\n\t\tAPROXIMACION POLINOMIAL POR MINIMOS CUADRADOS\n\n");
fprintf(appol,"GRADO DEL POLINOMIO = %d",n);
 fprintf(appol,"\n\n\t\tVALORES EXPERIMENTALES\n\n");
 fprintf(appol,"  X[i]\t  Y[i]\n");
 printf("\nINGRESE LOS VALORES Xi,Yi\n");
 for (d=0;d<M;++d)
```

```
  {
  printf("X[%d]=",d);
  scanf("%f",&x[d]);
  printf("Y[%d]=",d);
  scanf("%f",&y[d]);
  }
  printf("\nLOS DATOS QUE INGRESO SON :\n");
  printf("\tXi\t\tYi\n");
 for(d=0;d<M;++d)
  printf("%10.2f\t%10.2f\n",x[d],y[d]);
printf("\n");
printf("SI LOS DATOS ESTAN INCORRECTOS DIGITE @ SINO ENTER\n");
 ok=getche();
 if (ok=='@')
   rewind(appol);
 }
 for(d=0;d<M;++d)
  fprintf(appol,"%10.2f\t%10.2f\n",x[d],y[d]);
 sumas(M,n,x,y,sumax,sumayx);
 matrix(A,sumax,sumayx,n);
 resta(A,n);
 resolvedor(A,n,cop,appol);
 ne=evaluador(cop,n,appol,dexp,yexp);
 printf("\nSI DESEA VER UN GRAFICO DE LOS DATOS PRESIONE @\n");
 printf("SINO CUALQUIER OTRA TECLA\n");
 g=getche();
 if(g=='@')
 grafaprox(dexp,yexp,ne);
 clrscr();
 printf("\nSI DESEA CONTINUAR PRESIONE @ SINO CUALQUIER TECLA\n");
  o=getche();
  }
printf("\n\nLOS DATOS PUEDEN SER OBSERVADOS EN EL ARCHIVO APROXPOL.R\n\n");
return;
}
```

```
  /*ESTO CALCULA LA MEDIA Y LA MEDIANA */

void proceso(x,y,z,p,sump,scp)
int p;
float *x,*y,*sump,*scp;
float z[];
{
 int i;
 float suma,sumcuad,a,b;
 suma=0;
 sumcuad=0;
 for(i=0;i<p;++i)
```

```
  {
  suma += z[i];
  sumcuad +=z[i]*z[i];
  }
 *x=suma/p;                /*MEDIA*/

 *sump=suma;
 *scp=sumcuad;
 if(p%2==0)                /*MEDIANA*/
 {
 a=z[p/2];
 b=z[(p+2)/2];
 *y=(a +b)/2;
 }
 else
  *y=z[(p+1)/2];
 return;
}

 /*ESTE PROCEDIMIENTO CALCULA LA DESVIACION ESTANDAR*/

float variaz(k,sumpv,scpv)
float *sumpv,*scpv;
int k;
{
float sp,s1,sc1,parc;
 s1=*sumpv;
 sc1=*scpv;
 parc=((k*sc1)-(s1*s1))/(k*(k-1));
 sp=sqrt(parc);
 return(sp);
}



void datostotal(datol)
FILE *datol;
{
FILE *xavier;
tabla muestra[100];
FILE *lectura;
float ptemp[20];
float sum=0,media,mediana,s,sc=0;
char hora[8],horai[8]="08:00",horaf[8];
int i,n,h,j=0;
printf("\n\n\t\t\tESPERE UN MOMENTO\n");
 datol=fopen("A:\EVALDIA.R","w+");
 xavier=fopen("A:\datoxav.R","w+");
 if((lectura=fopen("a:\TEMPHORA.DAT","r"))==NULL)
 {
 printf("archivo vacio-fallo apertura");
 exit(0);
```

```
 }
while(j<14)
{
      n=0;
      do{
    fscanf(lectura,"%s%f",hora,&ptemp[n]);
   }while(strcmp(hora,horai)!=0);
 strcpy(muestra[j].horai,hora);
 for(i=0;i<12;++i)
    {
    n+=1;
    fscanf(lectura,"%s%f",hora,&ptemp[n]);
    strcpy(horai,hora);
    }
 ordena(ptemp,n);
 proceso(&media,&mediana,ptemp,n,&sum,&sc);
 s=variaz(n,&sum,&sc);        /*MUESTRA DATOS*/
 muestra[j].xbar=media;
     muestra[j].med=mediana;
 muestra[j].var=s;
       rewind(lectura);
  j+=1;
 }
   clrscr();
 printf("\n\nHORA\tMEDIA  MEDIANA\tDES.ESTD\n\n");
fprintf(datol,"\n\nHORA\tMEDIA  MEDIANA\tDES.ESTD\n\n");
 for (i=0,h=8;i<j;++i,h+=1)
 {
printf("%s\t%3.2f\t%3.2f\t%3.2f\n",muestra[i].horai,muestra[i].xbar,muestra[i].med,muestra[i].var);
fprintf(datol,"%s\t%3.2f\t%3.2f\t%3.2f\n",muestra[i].horai,muestra[i].xbar,muestra[i].med,muestra[i].var);
fprintf(xavier,"%d\t%3.2f\n",h,muestra[i].xbar);
 }
fclose(lectura);
fclose(xavier);
printf("\n\nLOS DATOS PUEDEN SER OBSERVADOS EN EL ARCHIVO EVALDIA.R\n\n");
return;
}



/*ESTA PARTE HACE LA ADQUICICION DE LOS DATOS DE LA BASE*/

void adqmuestra(tprom)
FILE *tprom;
{
     tabla muestra[100];
     FILE *lectura,*fopen();
 int nmuestra;
 float ptemp[500];
 char hora[8];
 char nm;
 int i=0,j=0,n;
```

```
float sum=0,media,mediana,s,sc=0;
tprom=fopen("a:\ESTIMADORES.R","w+");
do{
   clrscr();
   fprintf(tprom,"\n\t\tESTIMADORES DE LAS MUESTRAS DEL PROCESO\n\n");
   printf("\nINGRESE EL NUMERO DE MUESTRAS A TOMAR (MAX=5):");
   scanf("%d",&nmuestra);
}while(((nmuestra>=1)&&(nmuestra<=5))!=1);
fprintf(tprom,"NUMERO TOTAL DE MUESTRAS : %d\n\n",nmuestra);
if((lectura=fopen("a:\TEMPHORA.DAT","r"))==NULL)
 {
 printf("archivo vacio-fallo apertura");
 exit(0);
 }
for(j=0; j<nmuestra; j++)
  {
do{
  n=0;
   printf("\nINGRESE EL RANGO DE LA MUESTRA %d\n",j+1);
   printf("\n VALORES ENTRE 08:00 Y 22:00 SOLAMENTE\n\n");
   rango(muestra[j].horai,muestra[j].horaf);
 do{
   fscanf(lectura,"%s%f",hora,&ptemp[n]);
  }while(strcmp(hora,muestra[j].horai)!=0);
 do{
  n+=1;
  fscanf(lectura,"%s%f",hora,&ptemp[n]);
  }while(strcmp(hora,muestra[j].horaf)!=0);
  if (n==1)
      {
 printf("\nTIENEN QUE EXISTIR MAS DE UN DATO\n");
 rewind(lectura);
      }
 }while(n==1);
 ordena(ptemp,n);
 proceso(&media,&mediana,ptemp,n,&sum,&sc);
 s=variaz(n,&sum,&sc);        /*MUESTRA DATOS*/
 muestra[j].xbar=media;
     muestra[j].med=mediana;
 muestra[j].var=s;
 rewind(lectura);
  }
  /*ESTA PARTE MUESTRA LOS DATOS*/
 clrscr();
 printf("\n MEDIA\t\tMEDIANA\t\tDES.ESTD\t   RANGO\n\n");
 fprintf(tprom,"MEDIA\t\tMEDIANA\t\tDES.ESTD\t   RANGO\n\n");
 for (i=0;i<nmuestra;++i)
 {
printf("%f\t%f\t%f\t",muestra[i].xbar,muestra[i].med,muestra[i].var);
fprintf(tprom,"%f\t%f\t%f\t",muestra[i].xbar,muestra[i].med,muestra[i].var);
 printf("%s",muestra[i].horai);
 printf(" - ");
 printf("%s",muestra[i].horaf);
```

```
 printf("\n");
 fprintf(tprom,"%s",muestra[i].horai);
 fprintf(tprom," - ");
 fprintf(tprom,"%s",muestra[i].horaf);
 fprintf(tprom,"\n");
 }
 fclose(lectura);
printf("\n\nLOS DATOS LOS PUEDE OBSERVAR EN EL ARCHIVO PROMEDIO.R\n\n");
return;
}


void continua()
{
 printf("\n\tPRESIONE CUALQUIER TECLA PARA CONTINUAR\n");
 getch();
 return;
}


void menu2()
{
 clrscr();
 printf("\n\t\tESCOJA UNA DE LAS OPCIONES\n\n");
 printf("1.-TABLA DE MEDIA,MEDIANA Y VARIANZA.\n\n");
 printf("2.-APROXIMACION POLINOMIAL.\n\n");
 printf("3.-HISTOGRAMA - OJIVA.\n\n");
 printf("4.-EVALUACION DEL DIA.\n\n");
 printf("5.-FIN DE EJECUCION\n\n");
return;
}


void main()
{
char opcion;
FILE *appol1=NULL,*hisoj1=NULL,*prom=NULL,*datol=NULL;
 clrscr();
 do
 {
 menu2();
 printf("OPCION =");
 opcion=getche();
 getch();
 clrscr();
 switch (opcion)
  {
 case '1': { adqmuestra(prom);
  fclose(prom);

  continua();
   break;}
 case '2':{ aproxpol(appol1);
```

```
 fclose(appol1);
 continua();
  break;}
case '3': { histojiv(hisoj1);
 fclose(hisoj1);
 continua();
  break;}
case '4':{ datostotal(datol);
 fclose(datol);
 continua();
   break;}
case '5': exit();
default : break;
 }
}
while (opcion!='5');
return;
}
```

```
   /*ESTO INGRESA LA HORA*/

void rango(hoi,hof)
char hoi[],hof[];
{
 int i,j,err;
 char hi[20],hf[20];
 char *ini,*fini;
do{
 err=0;
  printf("Hora inicial (HH:MM) =");
  hi[0]=6;
 ini=cgets(hi);
  printf("\nHora final (HH:MM) =");
  hf[0]=6;
  fini=cgets(hf);
  for (i=2,j=0;j<6;++i,++j)
  {
  hoi[j]=hi[i];
  hof[j]=hf[i];
  }
   if(((hoi[0]>='0')&&(hoi[0]<='2'))==0)
  err=1;
   if(((hof[1]>='0')&&(hof[1]<='9'))==0)
        err=1;
  if (hoi[2]!=':')
  err=1;
   if(((hoi[3]>='0')&&(hoi[3]<='5'))==0)
  err=1;
   if(((hoi[4]==5)||(hoi[4]==0))!=0)
    {
    err=1;
    printf("\nLA ULTIMA CIFRA DEBE SER 0 O 5\n");
    printf("DIGITE CUALQUIER TECLA PARA CONTINUAR");
    getche();
    clrscr();
    }
  if(strcmp(hoi,hof)>0)
  {
  printf("\nLA HORA INICIAL DEBE SER menor A LA final\n");
  printf("DIGITE CUALQUIER TECLA PARA CONTINUAR");
  getche();
  err=1;
  clrscr();
  }
  if(((strcmp(hoi,"08:00"))<0)||((strcmp(hof,"22:00"))>0))
  {
  printf("\nHORA FUERA DE RANGO\n");
  printf("DIGITE CUALQUIER TECLA PARA CONTINUAR");
  getche();
```

```
  err=1;
  clrscr();
  }
   if (err==1)
   clrscr();
  }while(err!=0);
  printf("\n");
  *ini=*fini='\0';
 return;
}


   /*ESTO CALCULA LA POTENCIA*/

float potencia(pot,dato)
int pot;
float dato;
{
 int i;
 float vpot=1;
 for (i=0;i<pot;++i)
  vpot *=dato;
 return(vpot);
}


    /*GUARDA LAS SUMAS PARCIALES EN LA MATRIZ*/

void matrix(ap,sx,syx,np)
 float ap[][6];
 float sx[],syx[];
 int np;
{
 int c,f,r=0,k=0;
 for (f=0;f<=np;++f,++r)
  {
  k=r;
  for(c=0;c<=np+1;++c,++k)
   if (c<np+1)
    ap[f][c] = sx[k];
   else
    ap[f][c] = syx[f];
  }
 printf("\n");
 return;
}

   /*REDUCCION GAUSSIANA*/

void resta(apr,npr)
int npr;
```

```
float apr[][6];
{
 int u,i,j,x=0;
 float red;
 for (u=0;u<=npr-1;++u)
 {
  for (i=u+1;i<=npr;++i)
  {
   if(apr[u][u] !=0)
   {
   red=apr[i][u];
    for (j=u;j<=npr +1;++j)
   apr[i][j]-=(red/apr[u][u])*apr[u][j];
   }
   else
   {
    x=cambio(apr,u,i,npr);
   if (x==1){
   printf("\ninfinitas soluciones\n");}
   }
  }
 }
 printf("\n");
 return;
}

 /*INTERCALA LAS FILAS SI EL PIVOTE ES 0*/

int cambio(mat,up,ip,n)
float mat[][6];
int up,ip,n;
{
 float temp;
 int k=0,f=ip,c,prob=0;
 while ((k==0) && (f<=n))
 {
 if (mat[f][up]!=0)
  {
  k=1;
  for (c=0;c<=n+1;++c)
    {
    temp=mat[f][c];
    mat[f][c]=mat[up][c];
    mat[up][c]=temp;
    }
  }
 f+=1;
 }
 if ((k==0)&&(f==n))
  prob=1;
 return(prob);
}
```

```
void ordena(dat,n)
float dat[];
int n;
{
int i,j;
float menor,tp;
for (i=0;i<n;++i)
  for (j=i+1;j<n;++j)
 if(dat[j]<dat[i])
   {
   tp=dat[i];
   dat[i]=dat[j];
   dat[j]=tp;
   }
return;
}


float maximo(dat,n)
float dat[];
int n;
{
int i;
float mayor;
mayor=dat[0];
for (i=1;i<n;++i)
 if (dat[i]>mayor)
    mayor=dat[i];
return(mayor);
}


void getdate(hoi)
char hoi[];
{

 int i,j,err;
 char hf[20];
 char *fecha;
do{
 err=0;
  printf("INGRESE LA FECHA (MM/DD/AA) =");
  hf[0]=9;
 fecha=cgets(hf);
  for (i=2,j=0;j<9;++i,++j)
    hoi[j]=hf[i];
  if(((hoi[2]!='/')||(hoi[5]!='/'))==1)
  err=1;
  else if(((hoi[0]<'0')&&(hoi[0]>'1'))==1)
  err=1;
        else if(((hoi[1]<'0')&&(hoi[1]>'9'))==1)
```

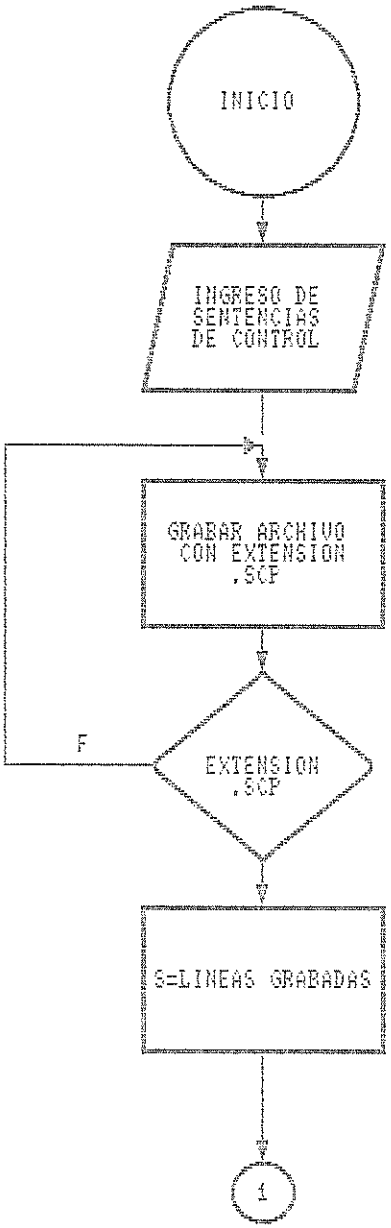
```
  err=1;
  if(((hoi[3]<'0')&&(hoi[3]>'3'))==1)
  err=1;
        else if(((hoi[4]<'0')&&(hoi[4]>'9'))==1)
  err=1;
   }while(err==1);
 *fecha='\0';
return;
}


/*main ()
{
char hoi[9];
getdate(hoi);
printf("\n%s\n",hoi);
return;
} */
```

C .- F I G U R A S .

Diagrama de flujo del proceso de Compilacion

Fig. 2.1.



Continuacion Fig.2.1.

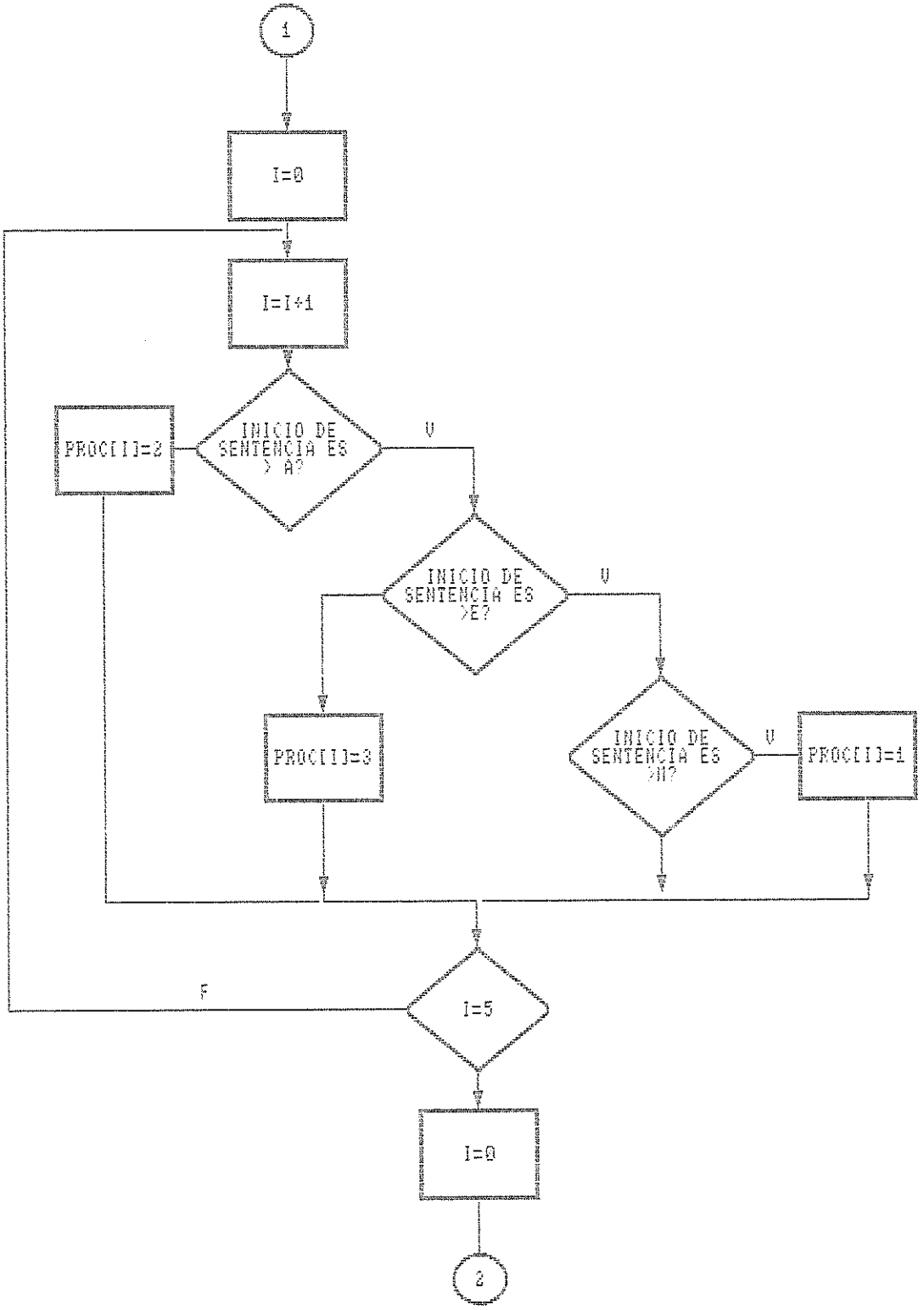
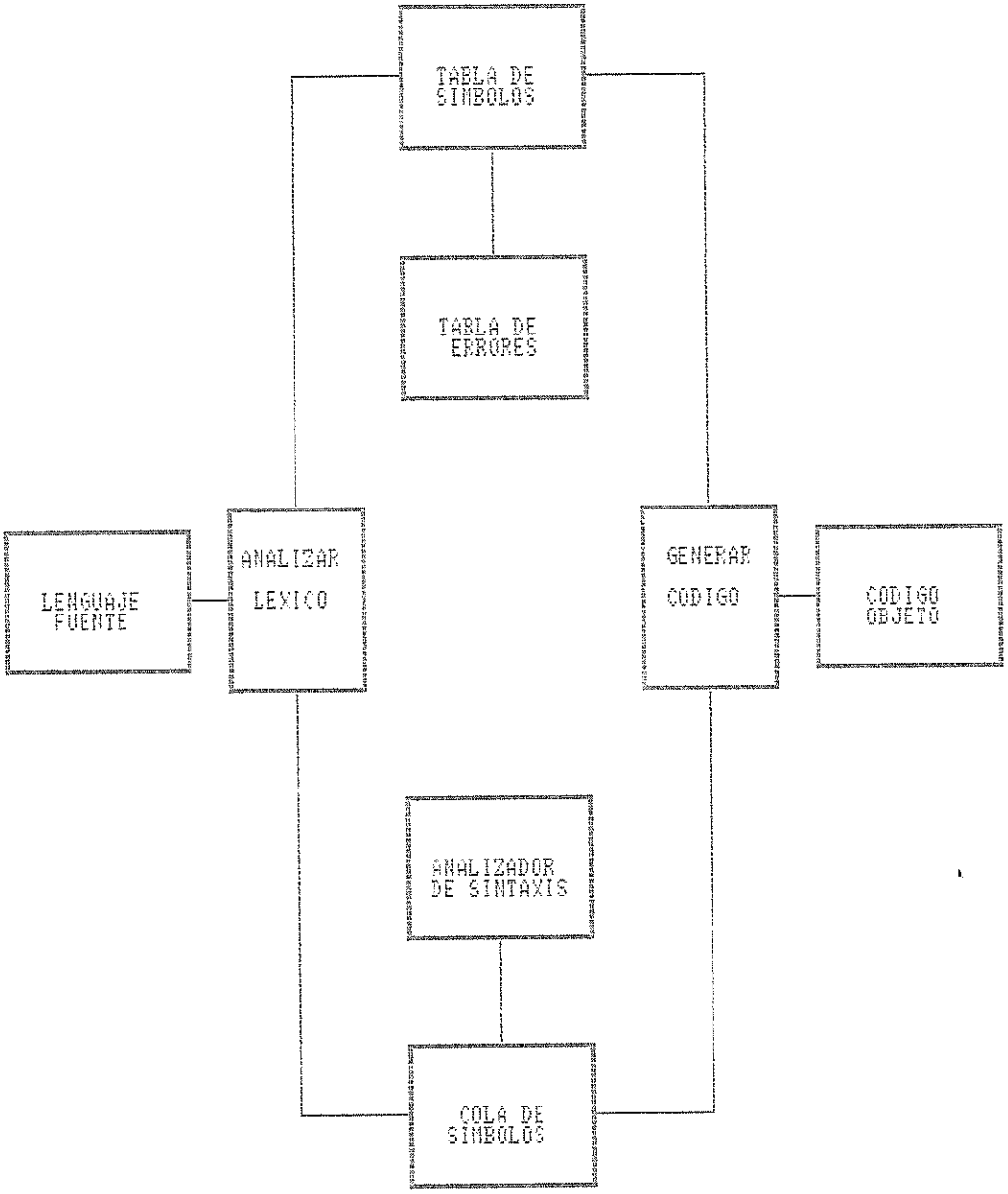


Diagrama del Proceso de Compilacion

Fig. 2.2



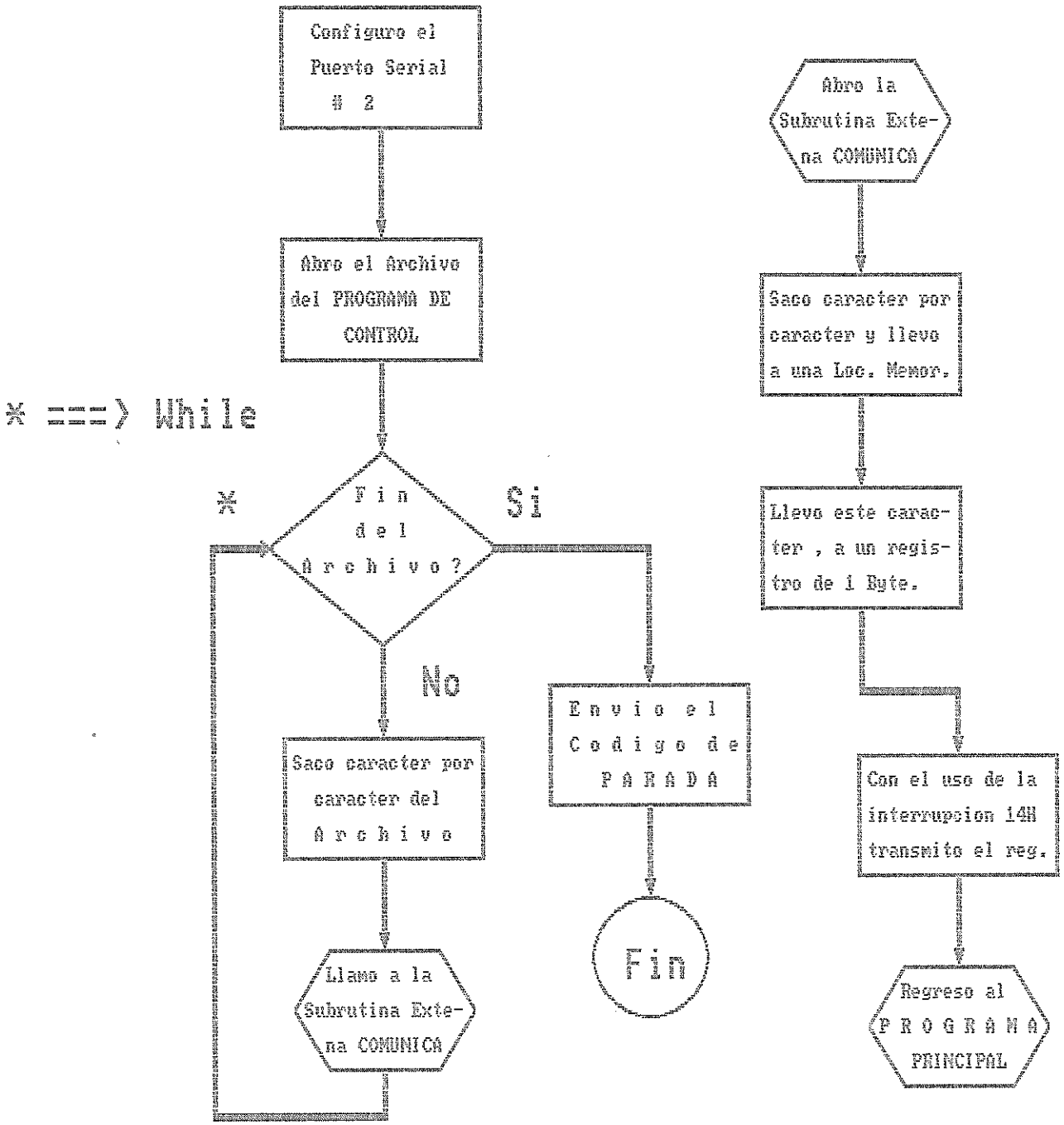
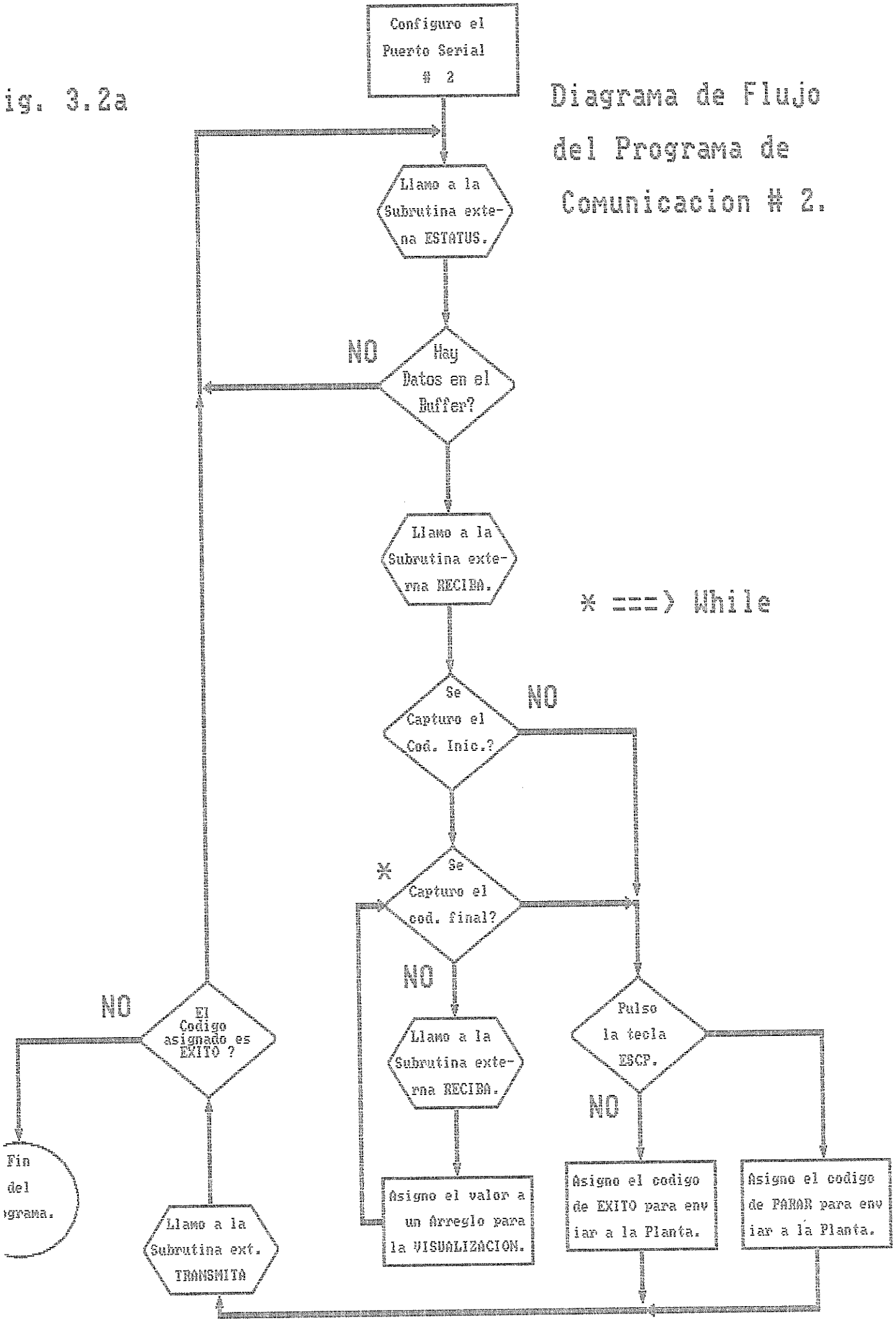


Fig. 3.1

DIAGRAMA DE FLUJO DEL PROGRAMA DE COMUNICACION # 1 .

Fig. 3.2a

Diagrama de Flujo del Programa de Comunicacion # 2.



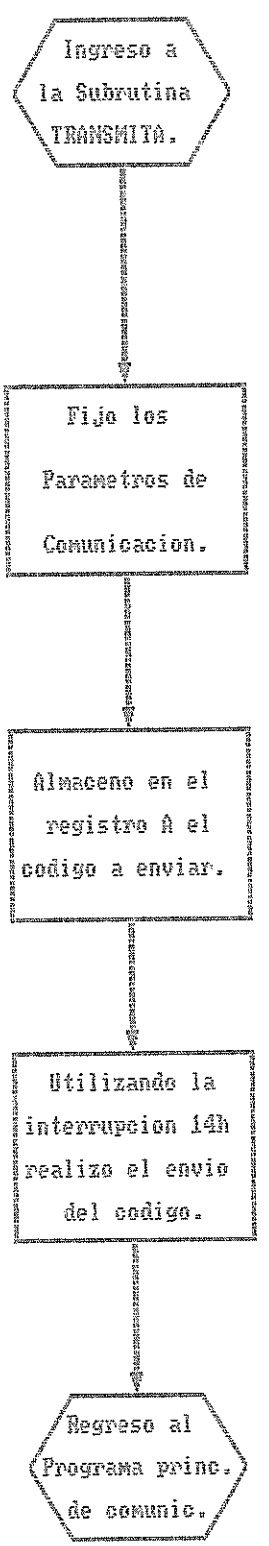


Fig. 3.2b

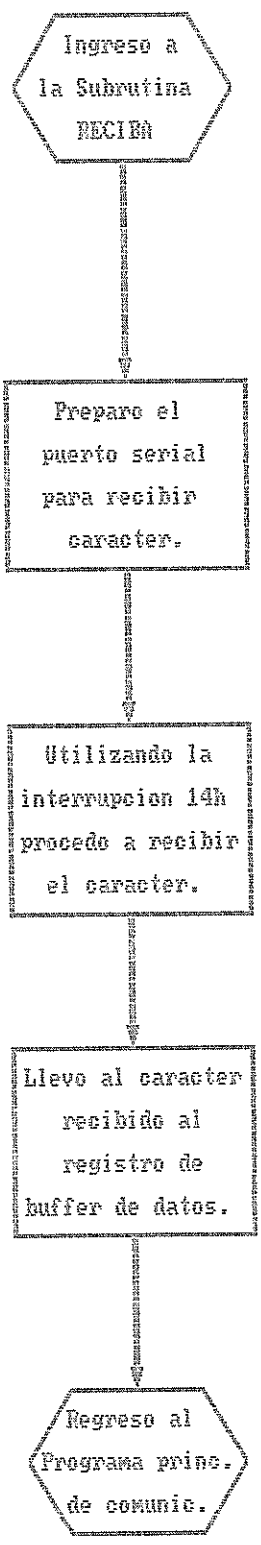


Fig. 3.2c

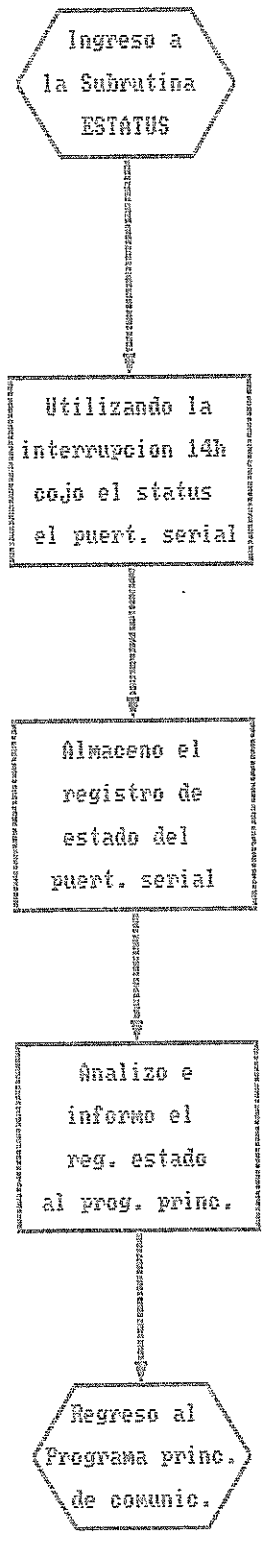


Fig. 3.2d

Diagrama de flujo de las Subrutinas del

Programa de Comunicacion # 2

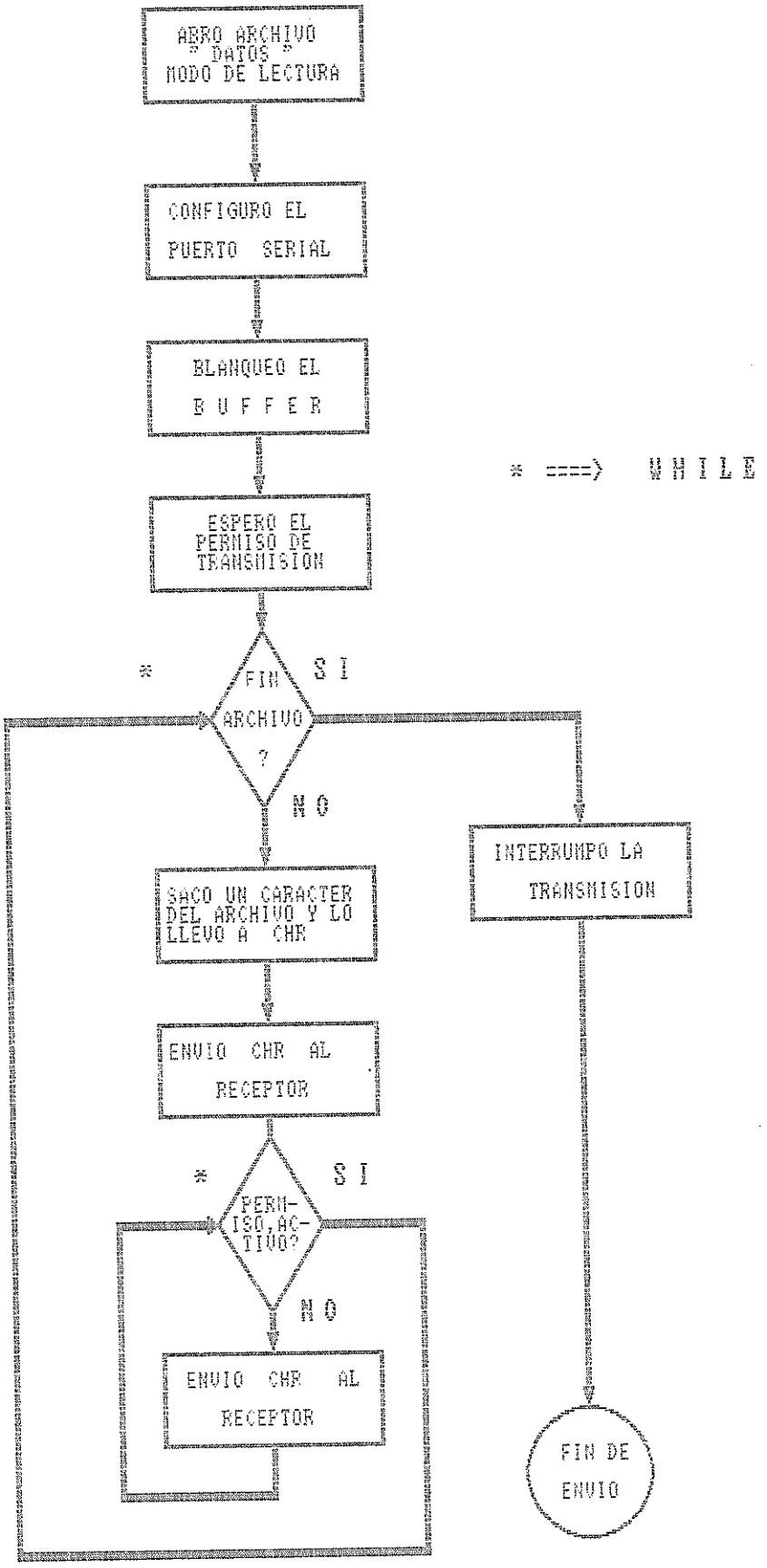


Fig. 3.3

DIAGRAMA DE BLOQUES DEL PROGRAMA EMISOR

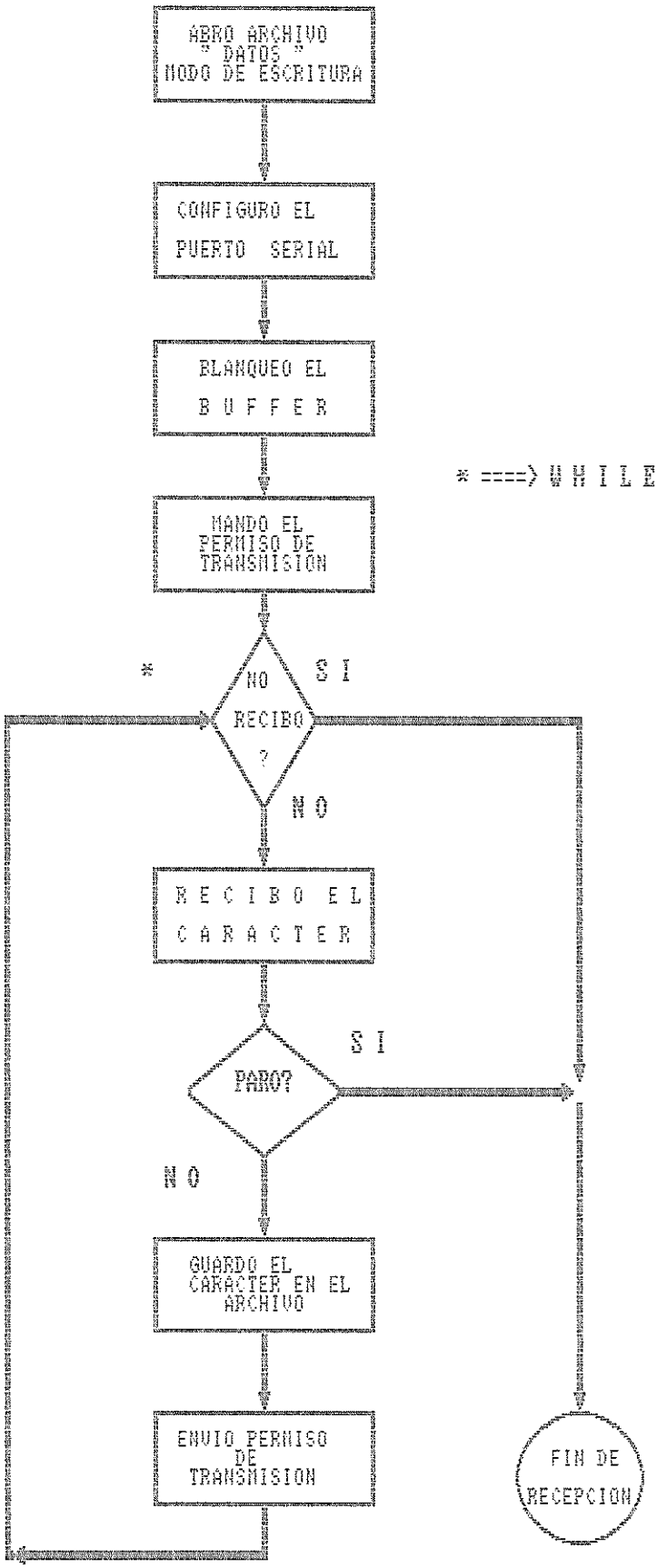


Fig. 3.4

DIAGRAMA DE BLOQUES DEL PROGRAMA RECEPTOR

Fig. 4.1

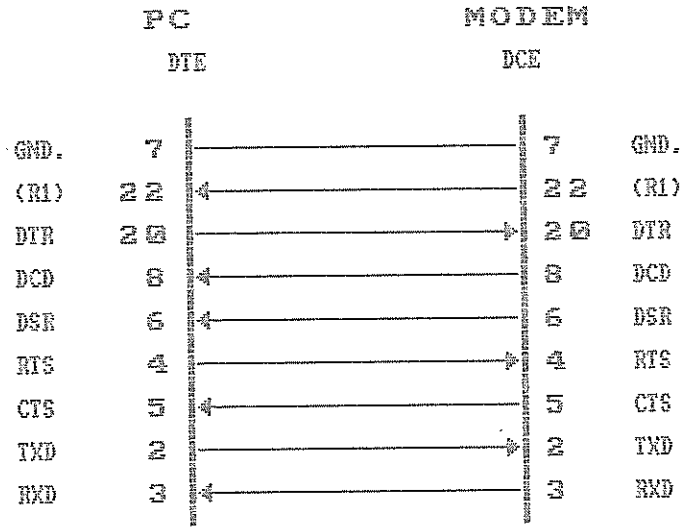


Fig. 4.2

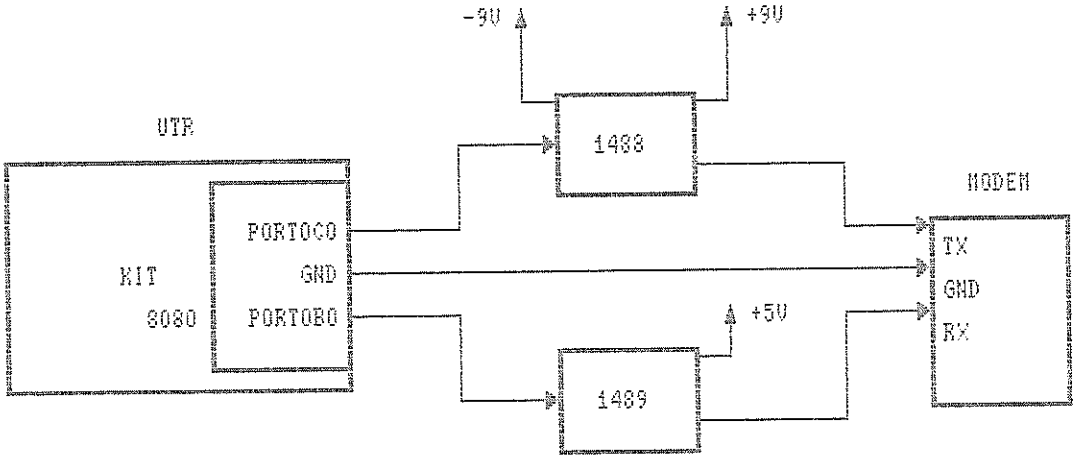


Fig. 4.3

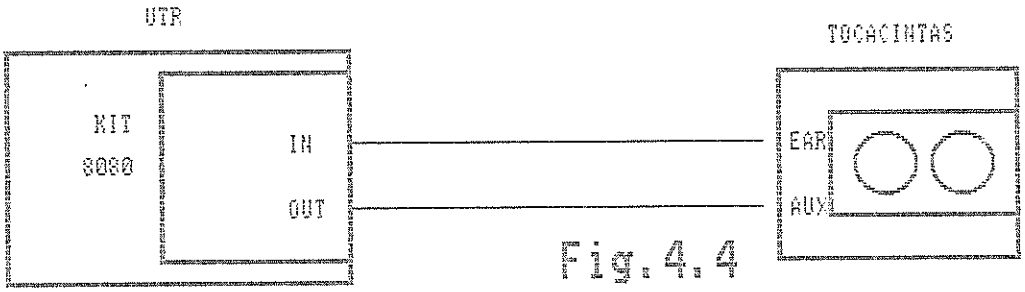


Fig.4.4

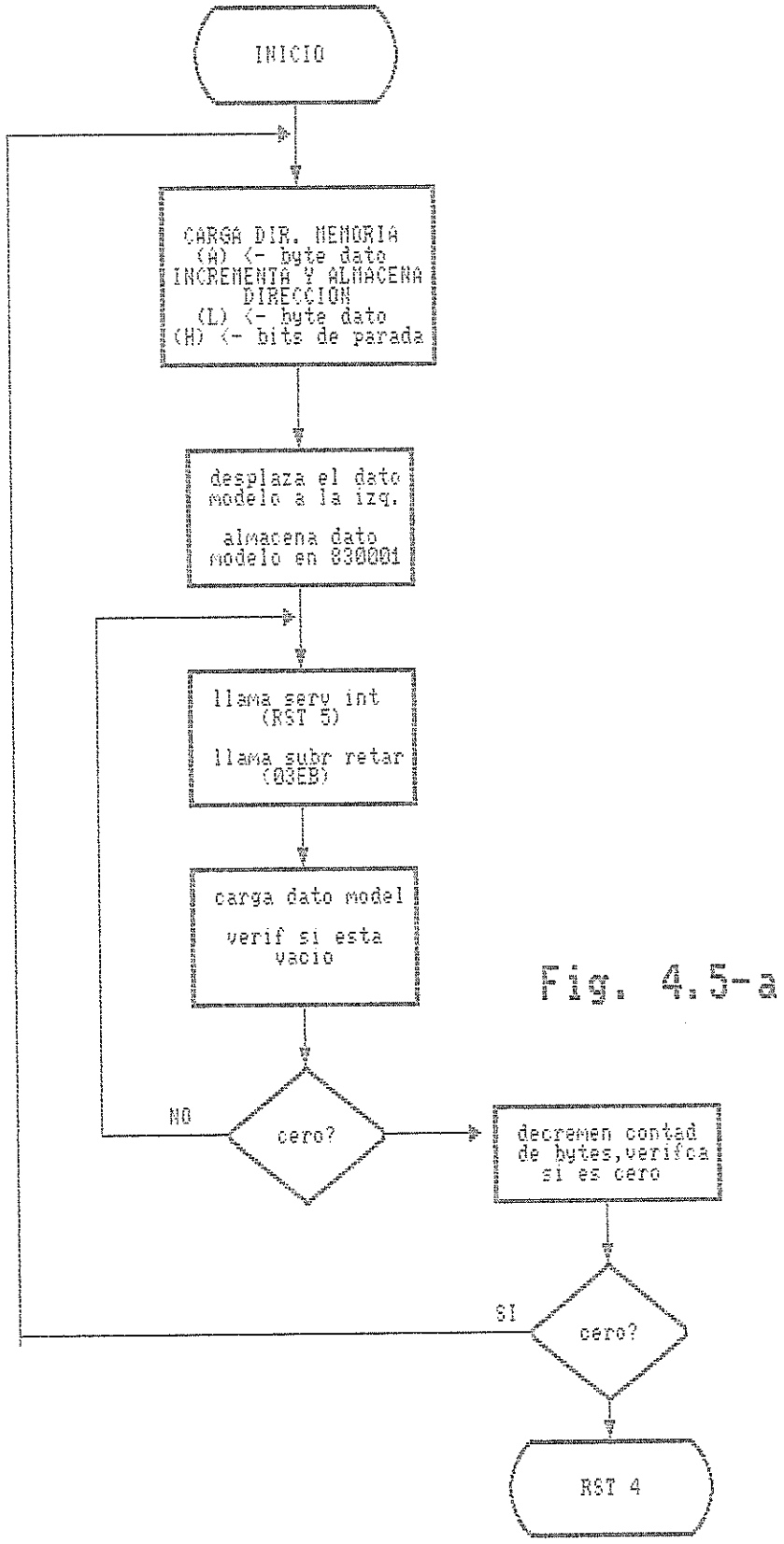


Fig. 4.5-a

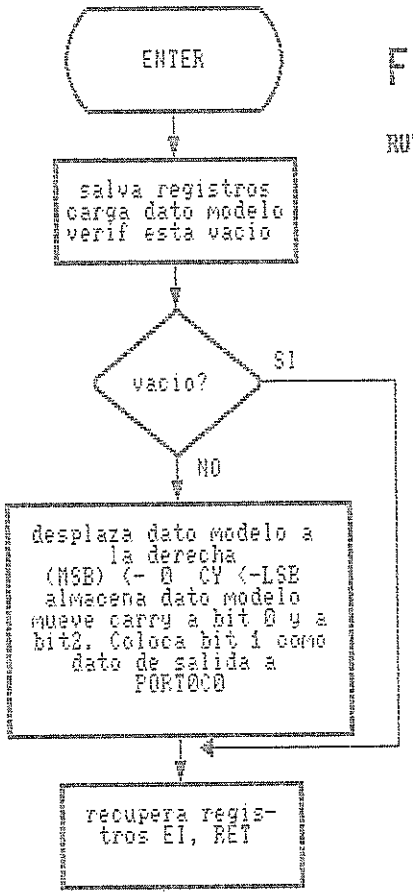


Fig. 4.5-b

RUTINA DE SERVICIO DE INTERRUPCION
TRANSMISION SERIAL

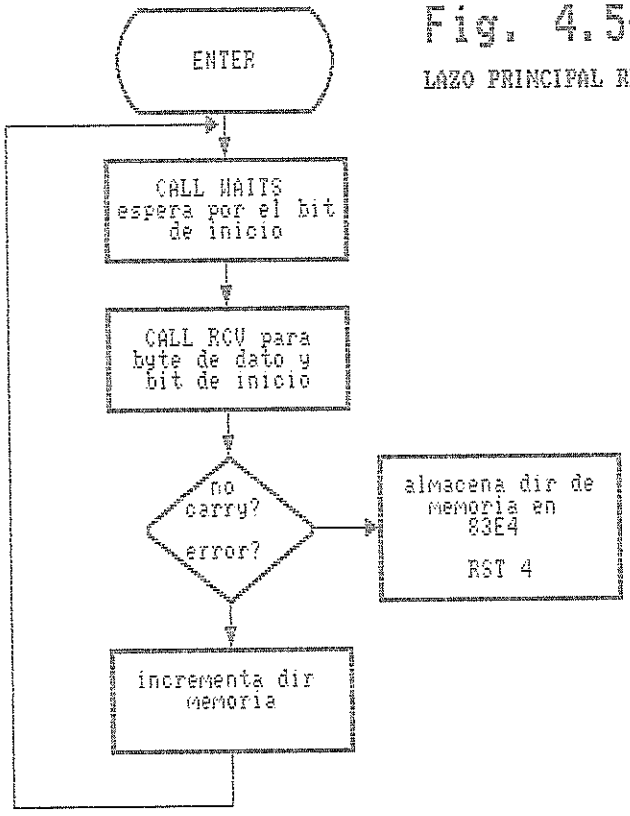


Fig. 4.5-c

LAZO PRINCIPAL RECEPCION SERIAL

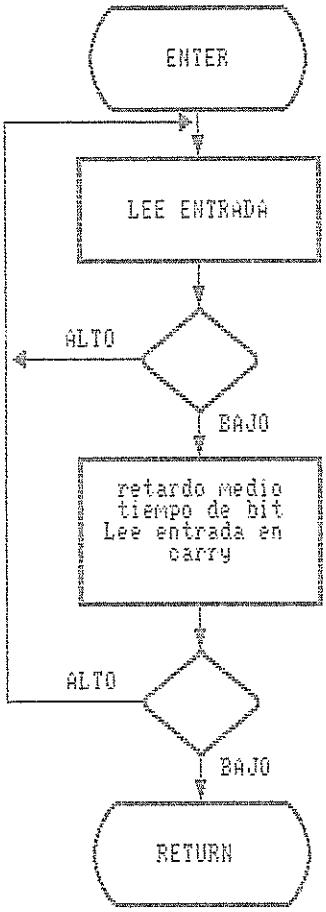


Fig. 4.5.d

SUBRUTINA WAITS

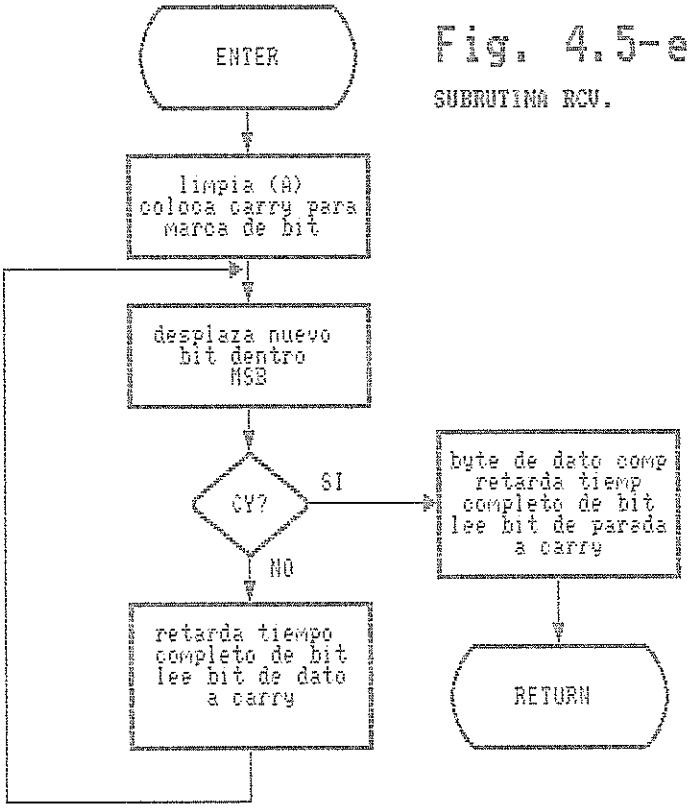


Fig. 4.5-e

SUBRUTINA RCV.

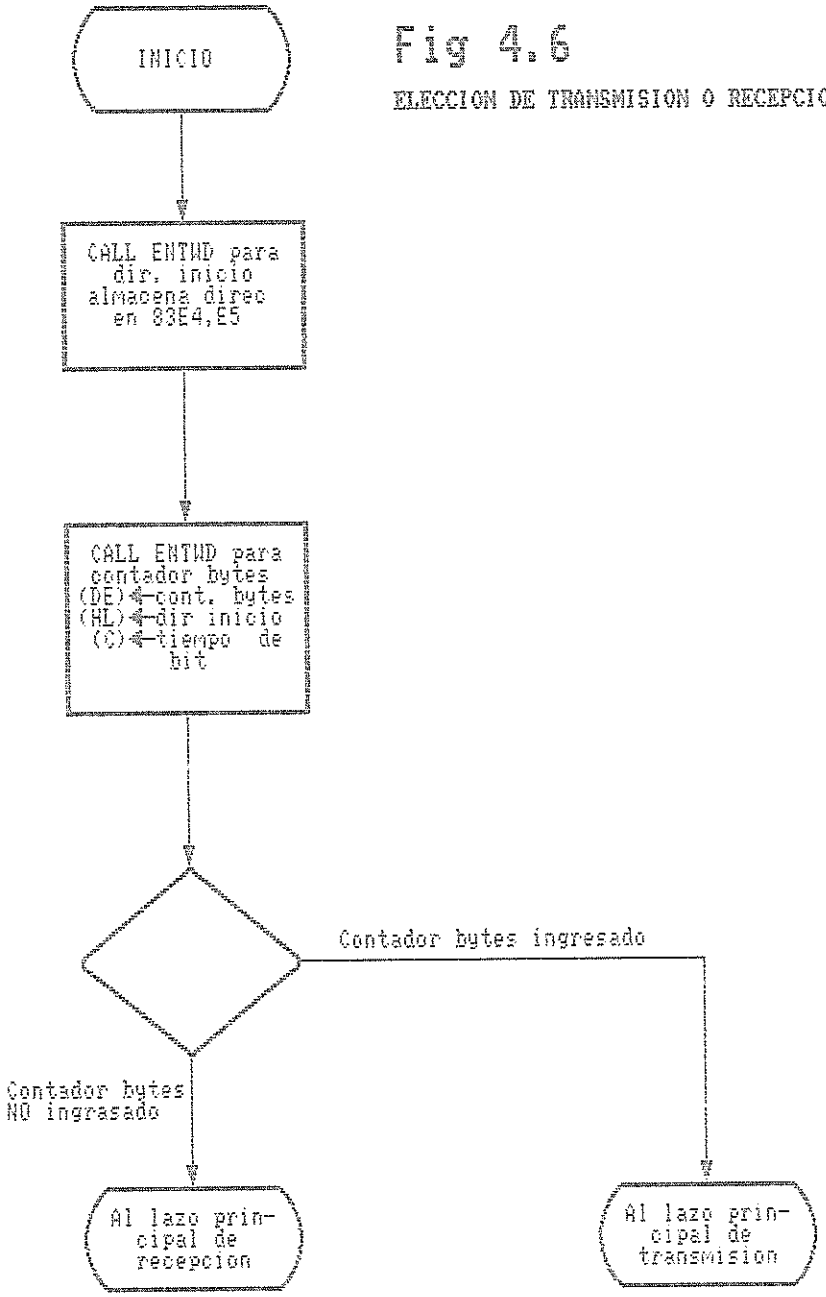


Fig 4.6

ELECCION DE TRANSMISION O RECEPCION

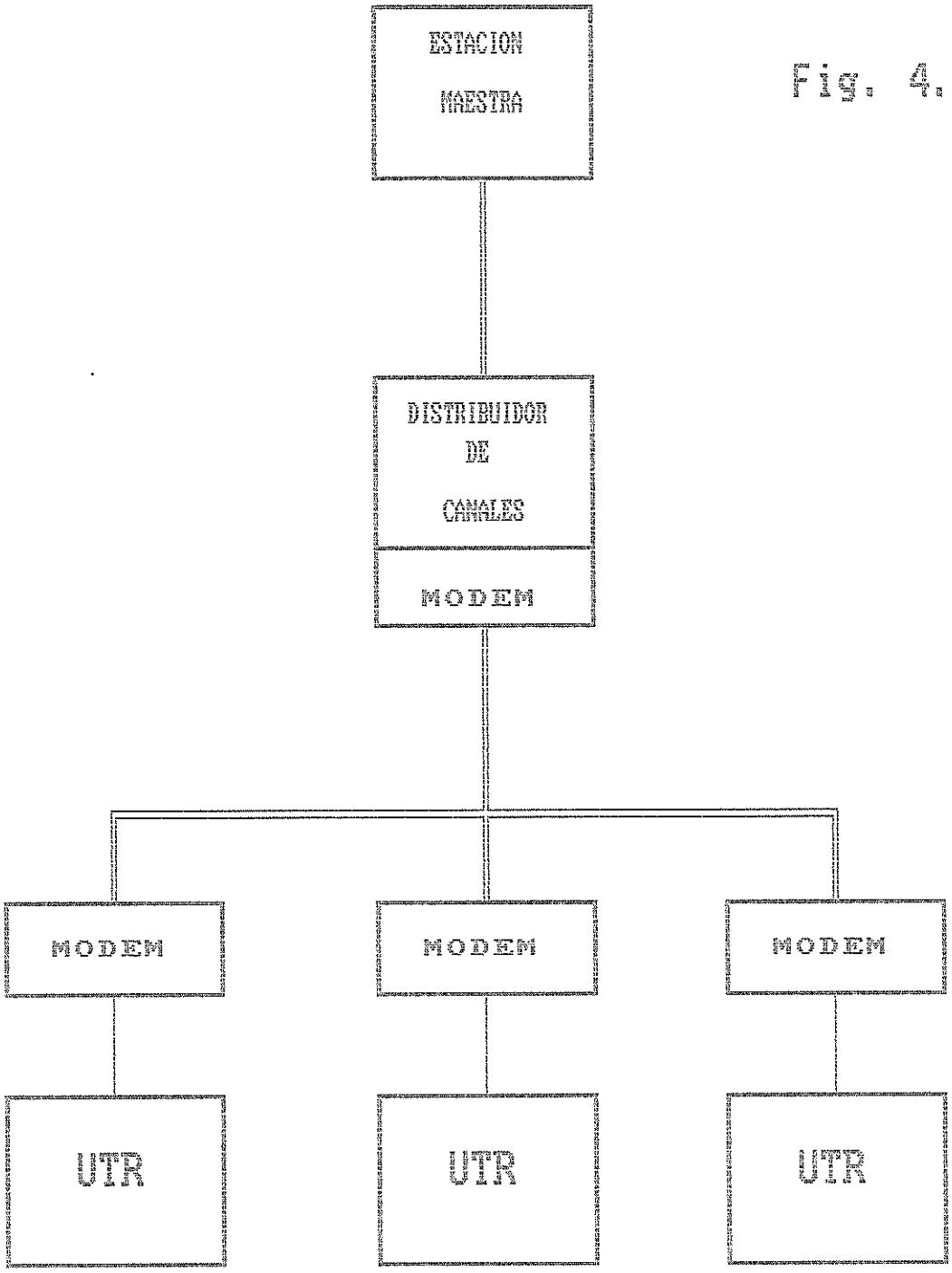

ESTACION
MAESTRA
DISTRIBUIDOR
DE
CANALES
MODEM
MODEM
MODEM
MODEM
UTR
UTR
UTR

Fig. 4.7

Configuracion tipica de una Unidad Programable

Fig. 5.2

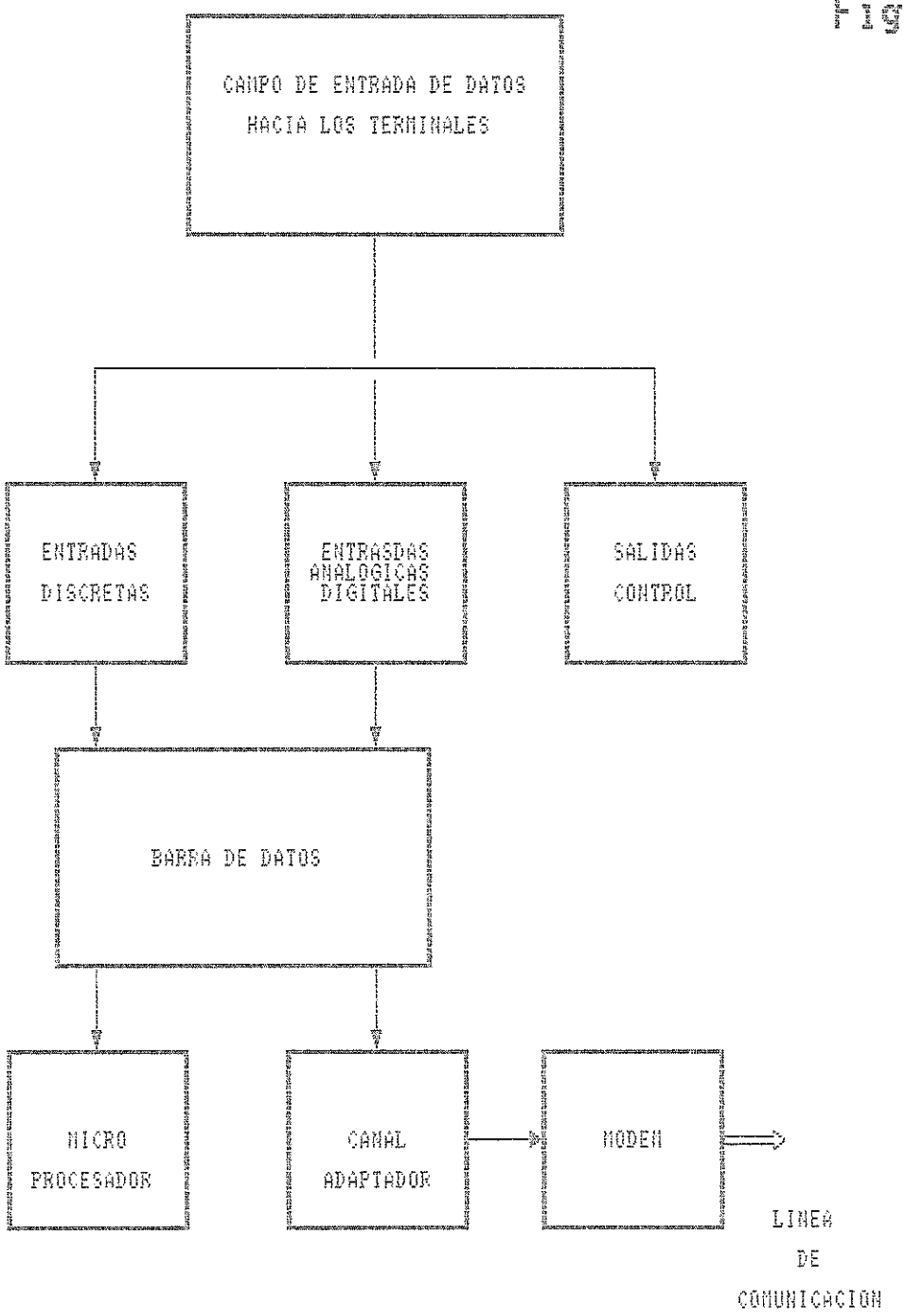


Diagrama del Control Principal .

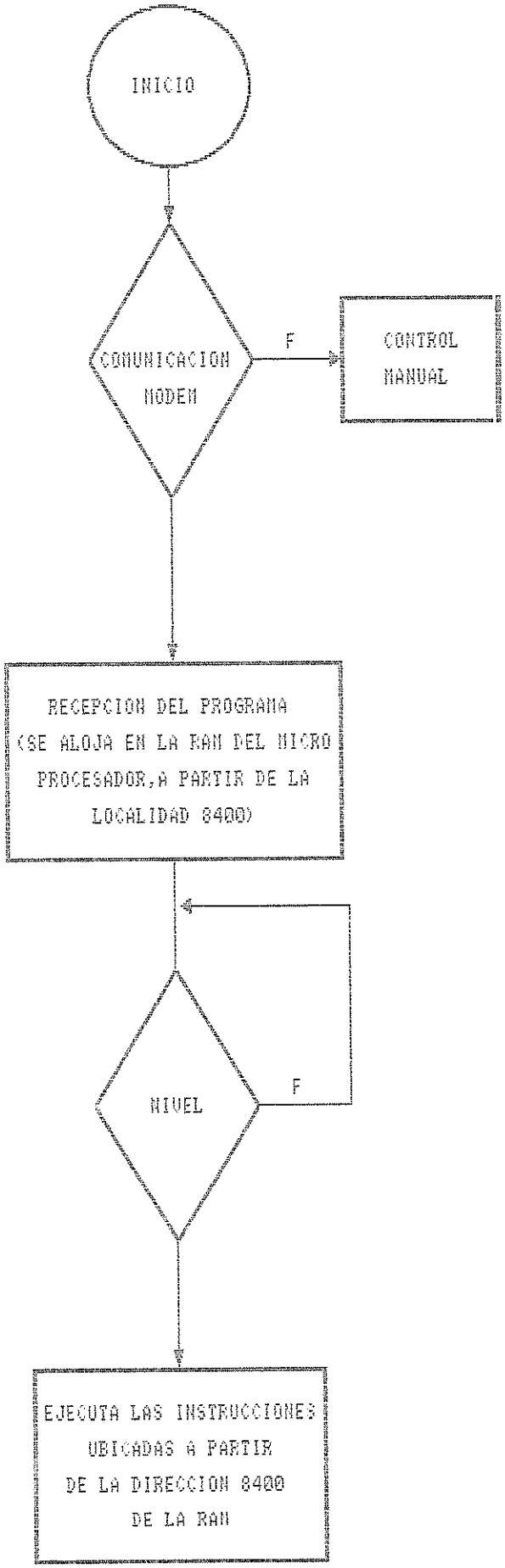


Fig. 5.3

Subrutinas de apagado y encendido

Fig. 5.4.

Subrutina de espera

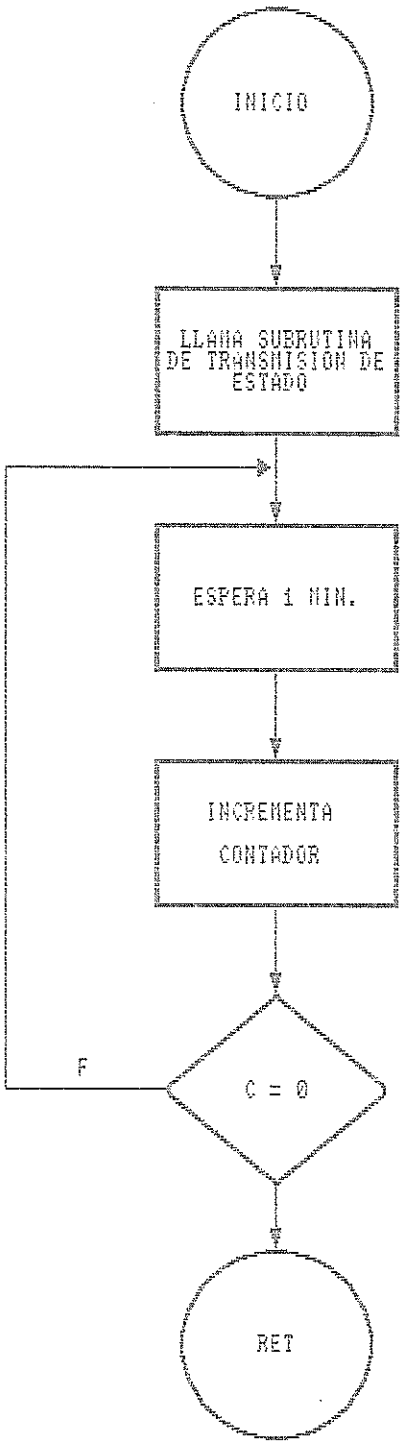


Fig. 5.5

Subrutina de fijacion de temperatura

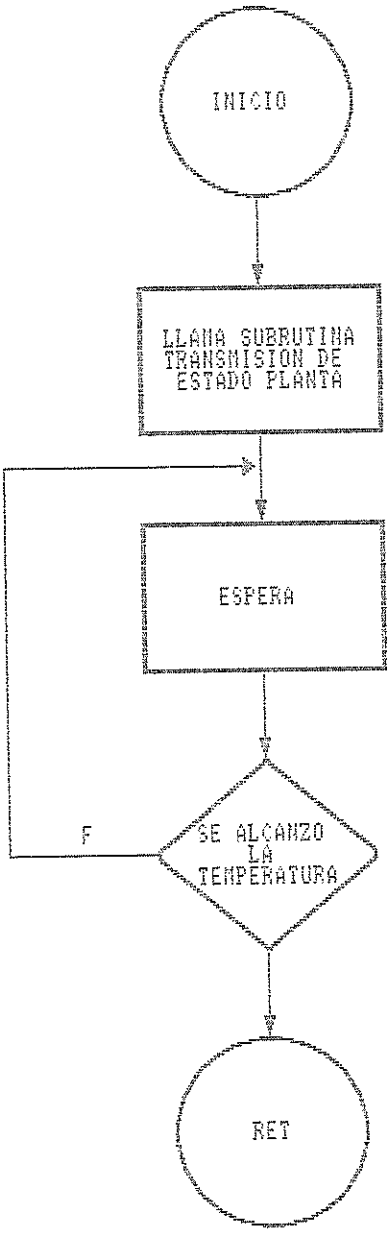
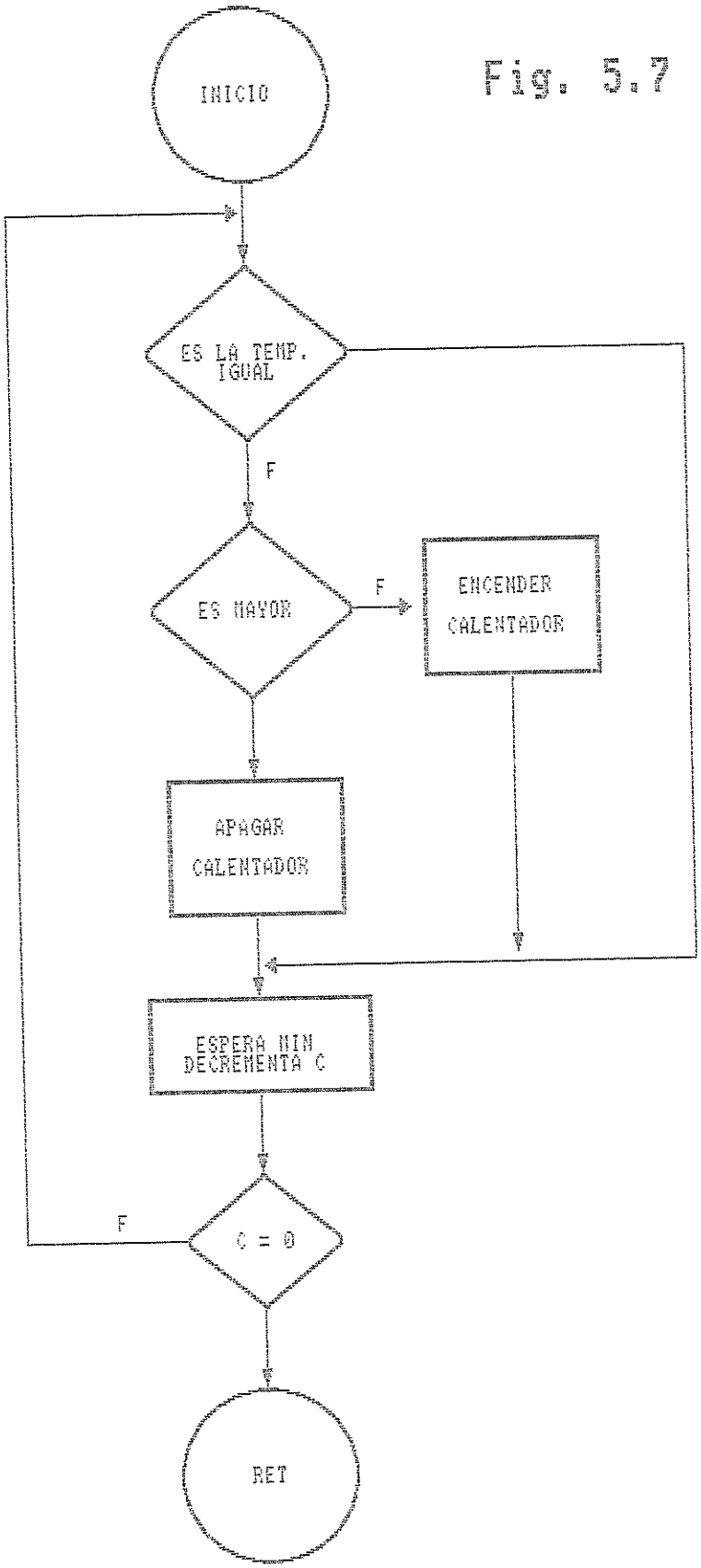


Fig. 5.6

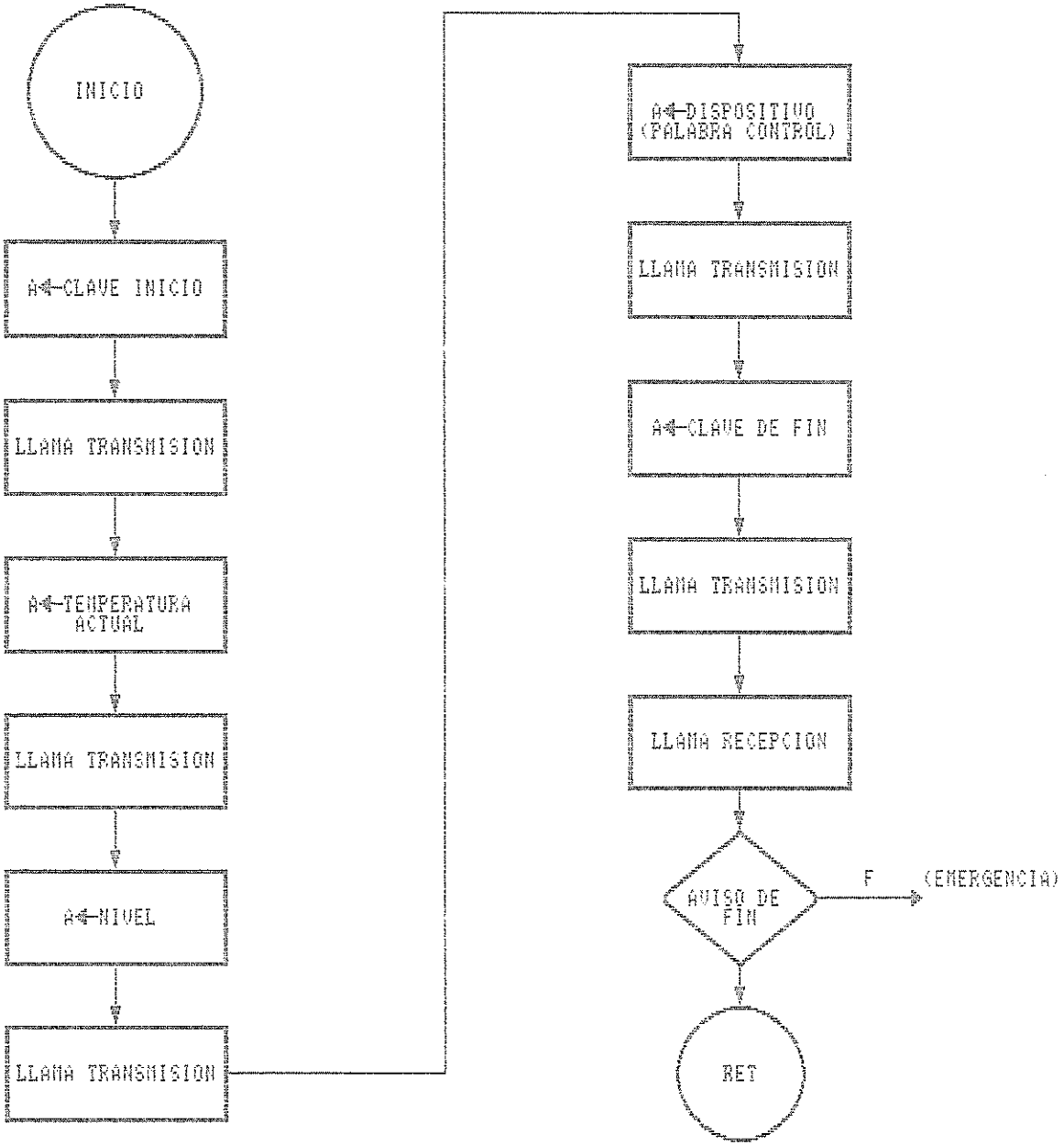
Subrutina de Sostenimiento de Condiciones

Fig. 5.7



Subrutina de transmision de estado de la Planta

Fig. 5.8.



Teclado de Control

Fig. 5.9.

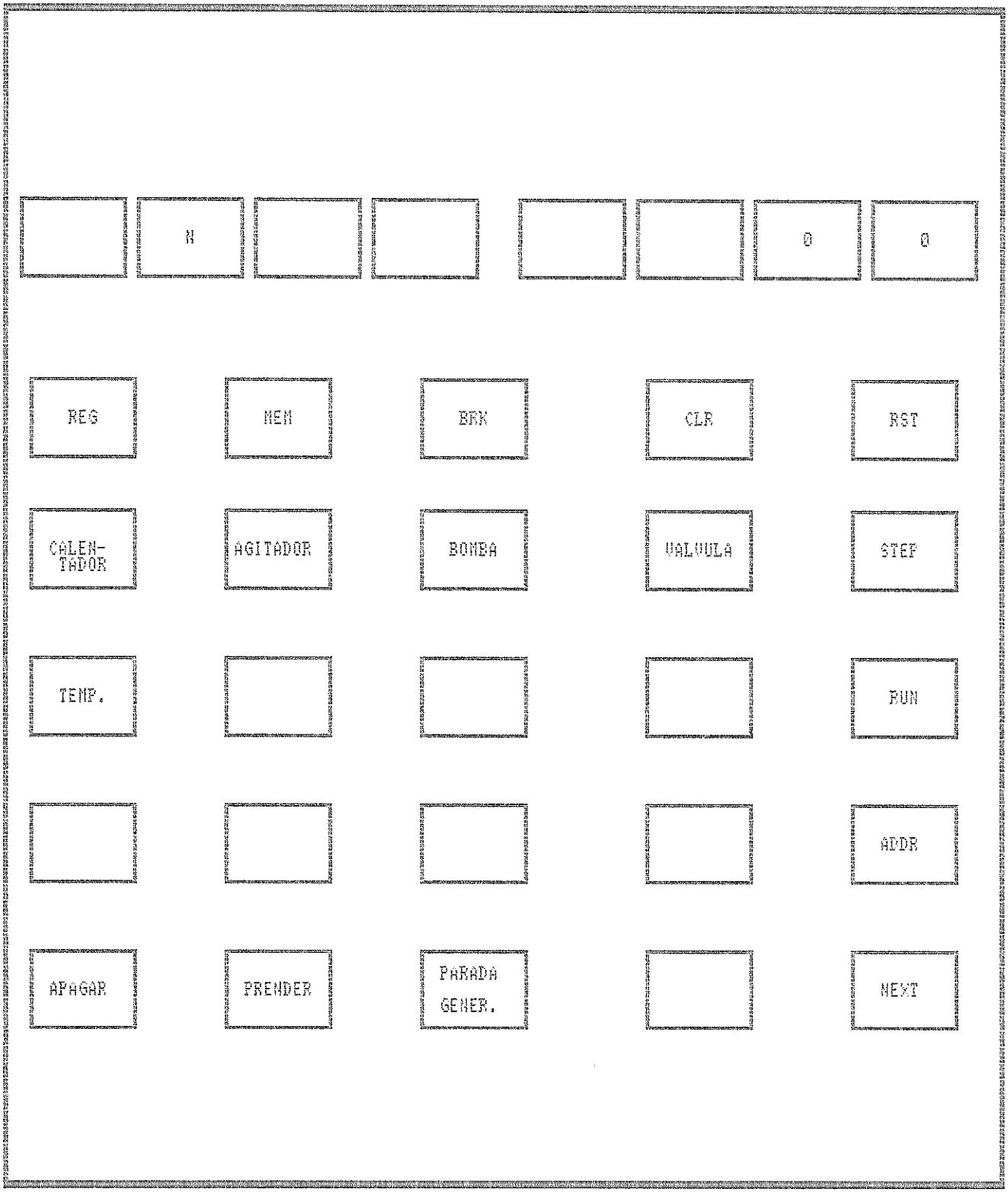
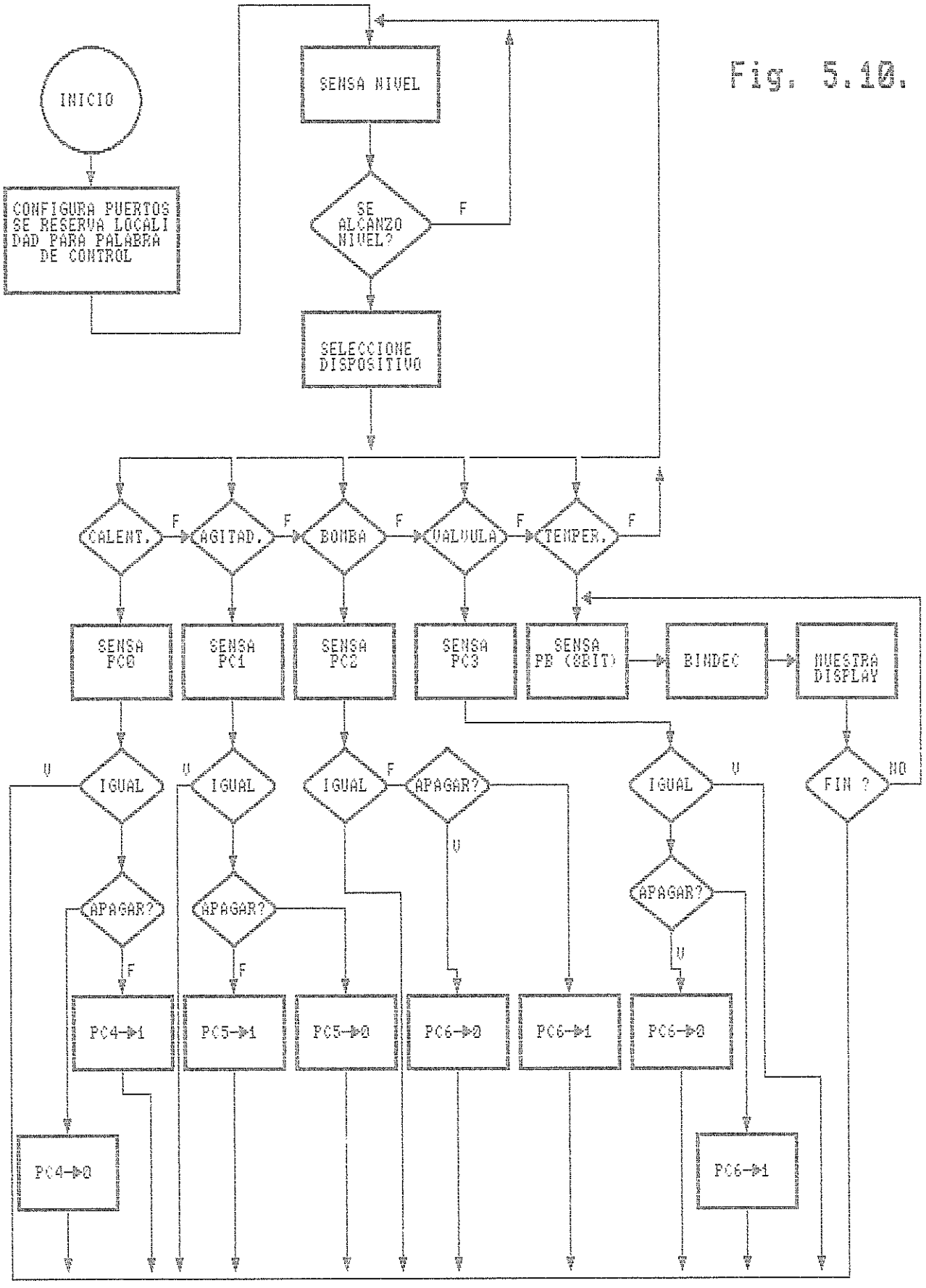


Fig. 5.10.


INICIO
CONFIGURA PUERTOS SE RESERVA LOCALIDAD PARA PALABRA DE CONTROL
SENSA NIVEL
SE ALCANZO NIVEL?
F
SELECCIONE DISPOSITIVO
CALENT.
AGITAD.
BOMBA
VALVULA
TEMPER.
F
SENSA PC0
SENSA PC1
SENSA PC2
SENSA PC3
SENSA PB (8BIT)
BINDEC
MUESTRA DISPLAY
IGUAL
APAGAR?
V
F
FIN ?
NO
PC4→1
PC5→1
PC5→0
PC6→0
PC6→1
PC6→0
PC4→0
PC6→1

Concepto en el Lazo de Retardo

Fig. 5.11

Subrutina de Retardo

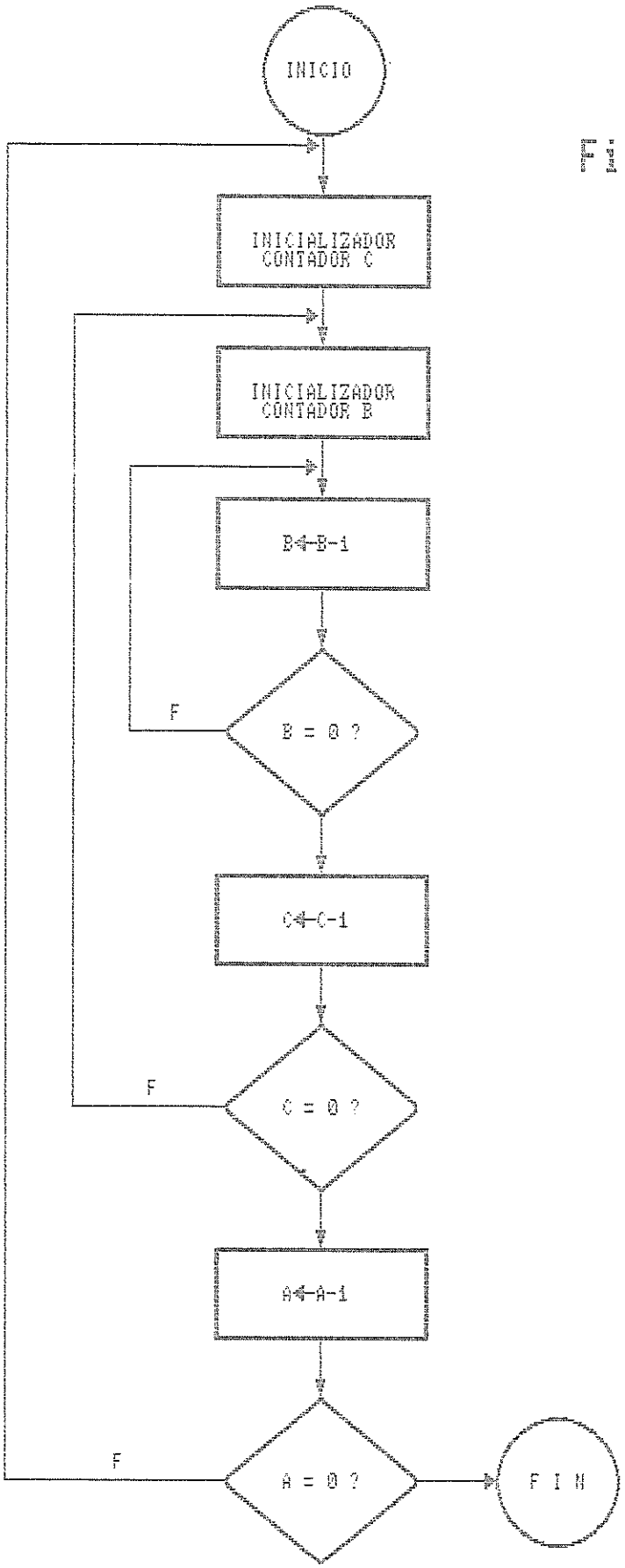


Fig. 5.12

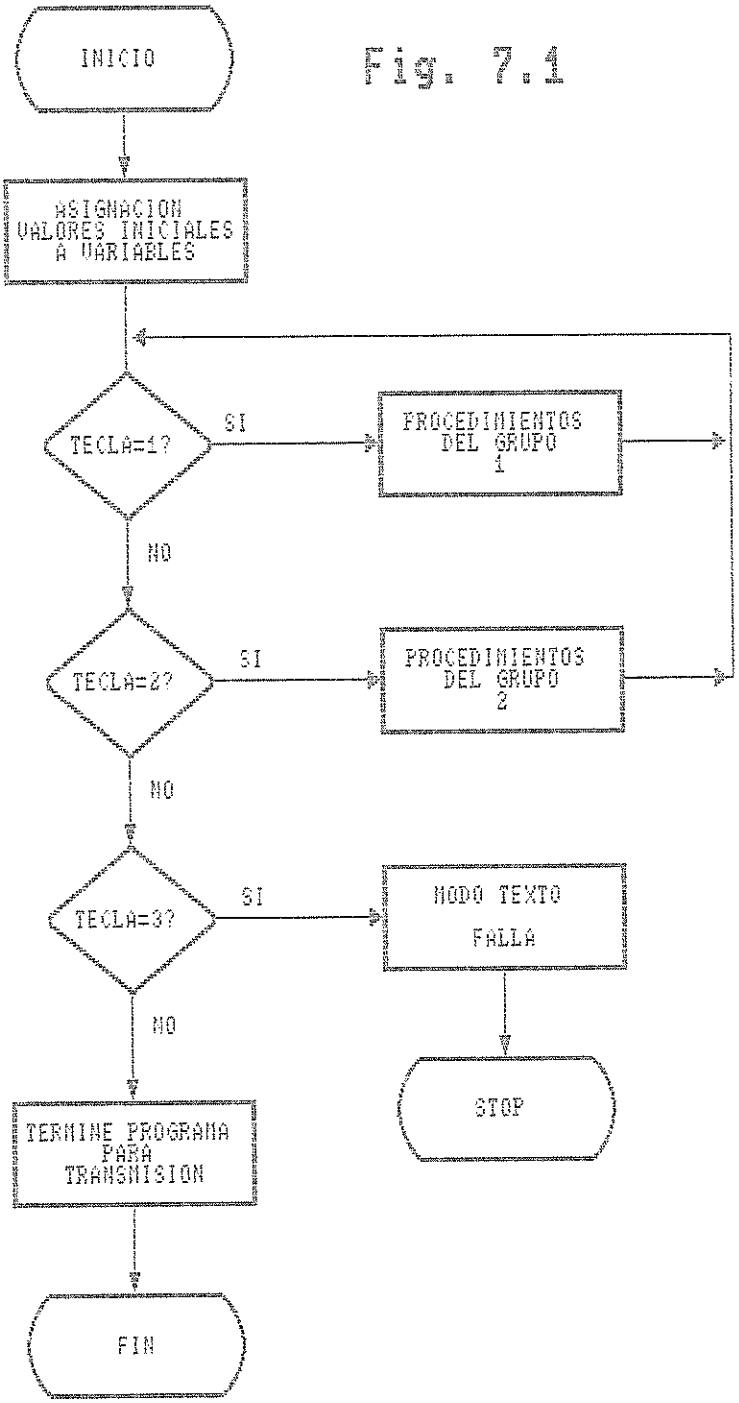


Fig. 7.1

DIAGRAMA DE FLUJO PRINCIPAL
DEL PROGRAMA DE VISUALIZACION

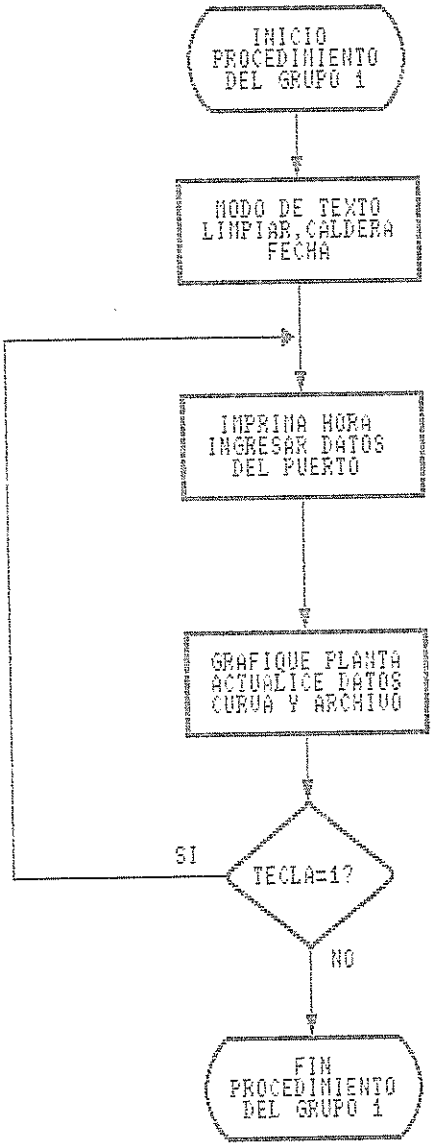


Fig. 7.2

PROCEDIMIENTOS DEL GRUPO 1

DEL PROGRAMA DE VISUALIZACION

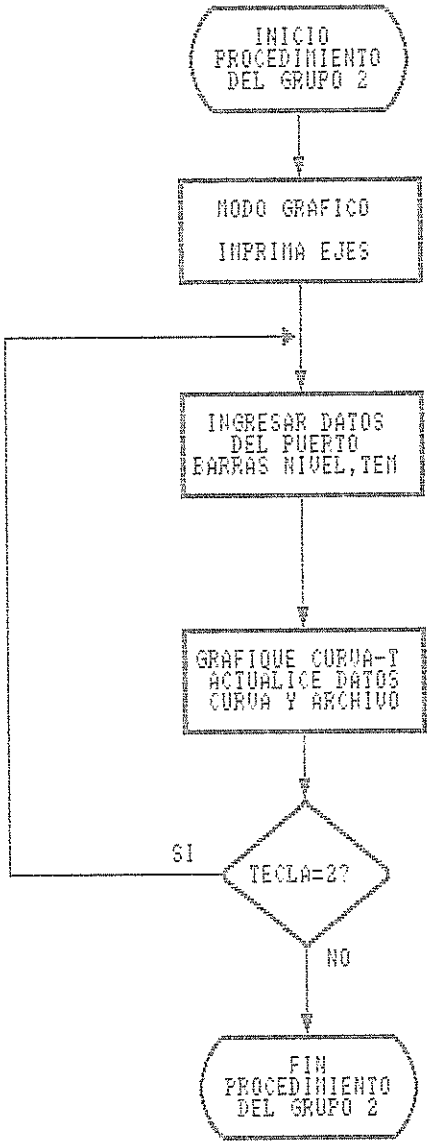


Fig. 7.3

PROCEDIMIENTOS DE GRUPO 2

DEL PROGRAMA DE VISUALIZACION

FIGURA 9.1—A

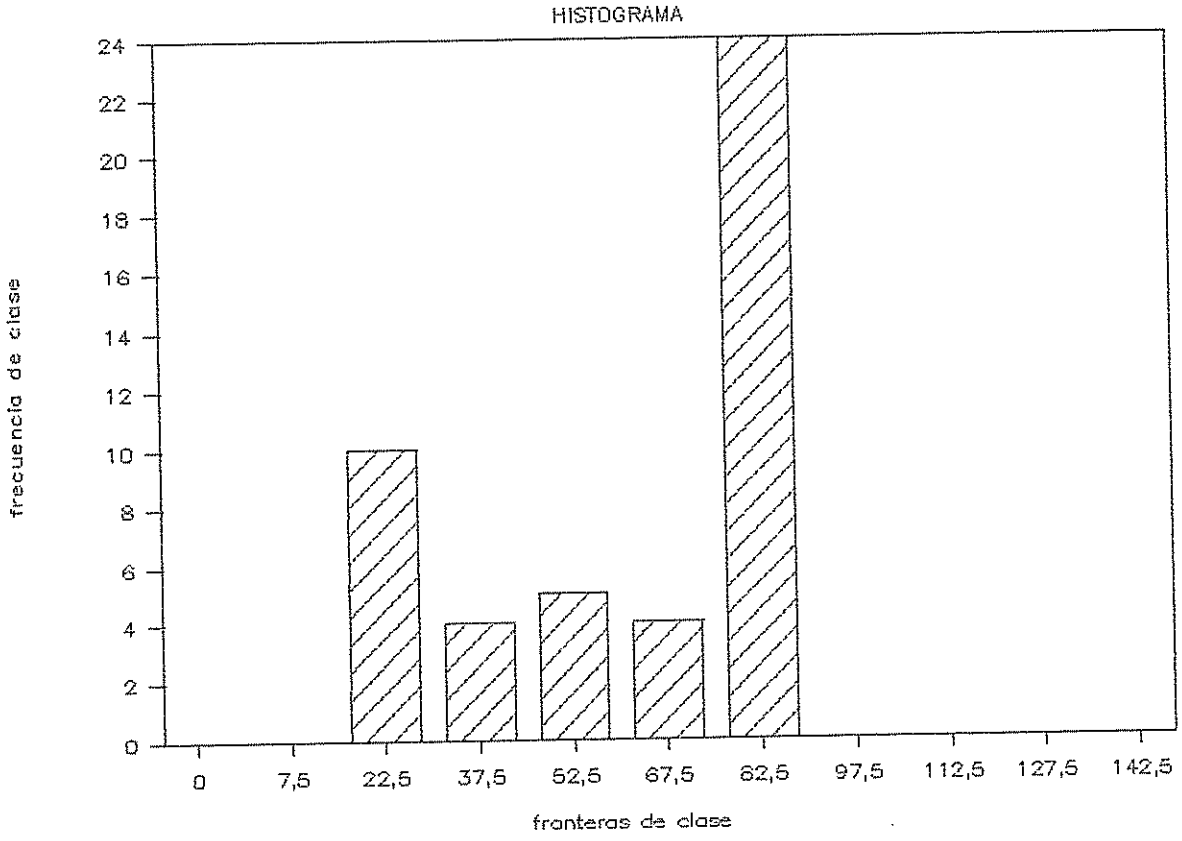

HISTOGRAMA
frecuencia de clase
24
22
20
18
16
14
12
10
8
6
4
2
0
0
7,5
22,5
37,5
52,5
67,5
82,5
97,5
112,5
127,5
142,5
fronteras de clase

FIGURA 9.1—B

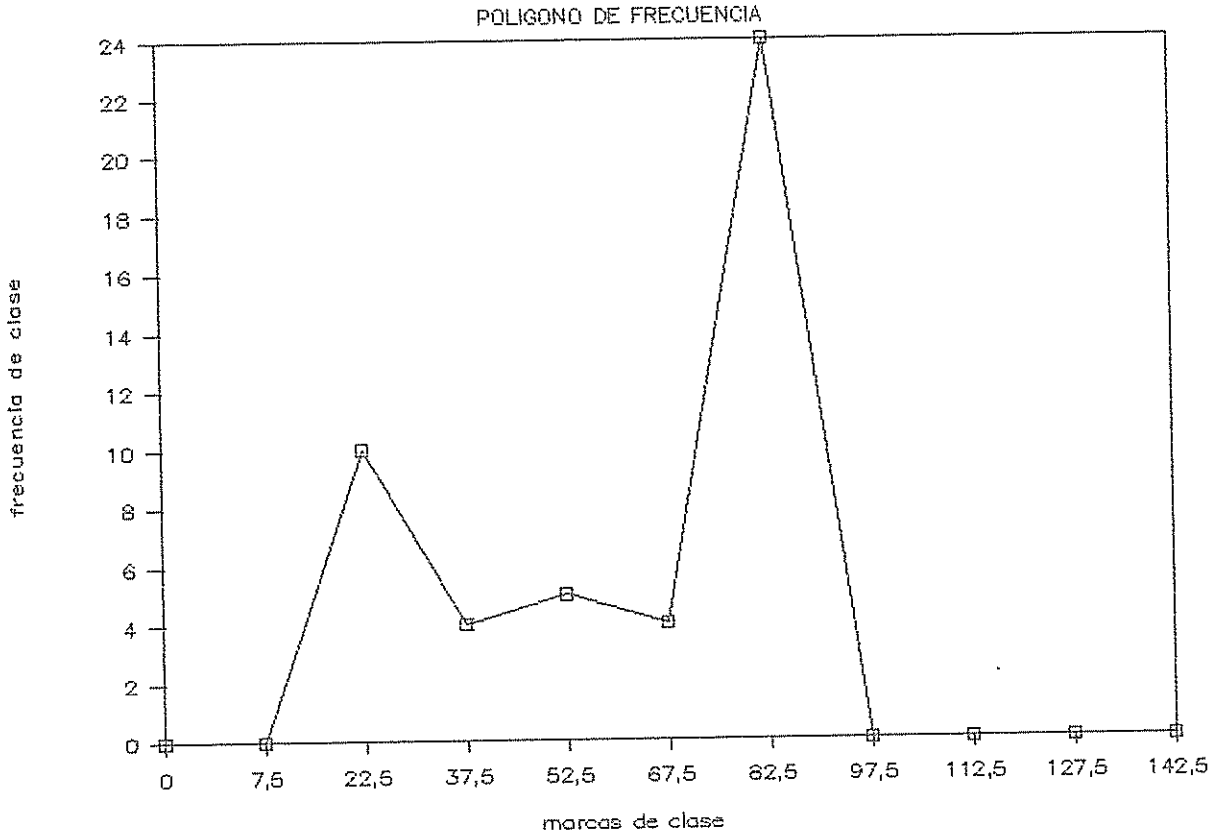
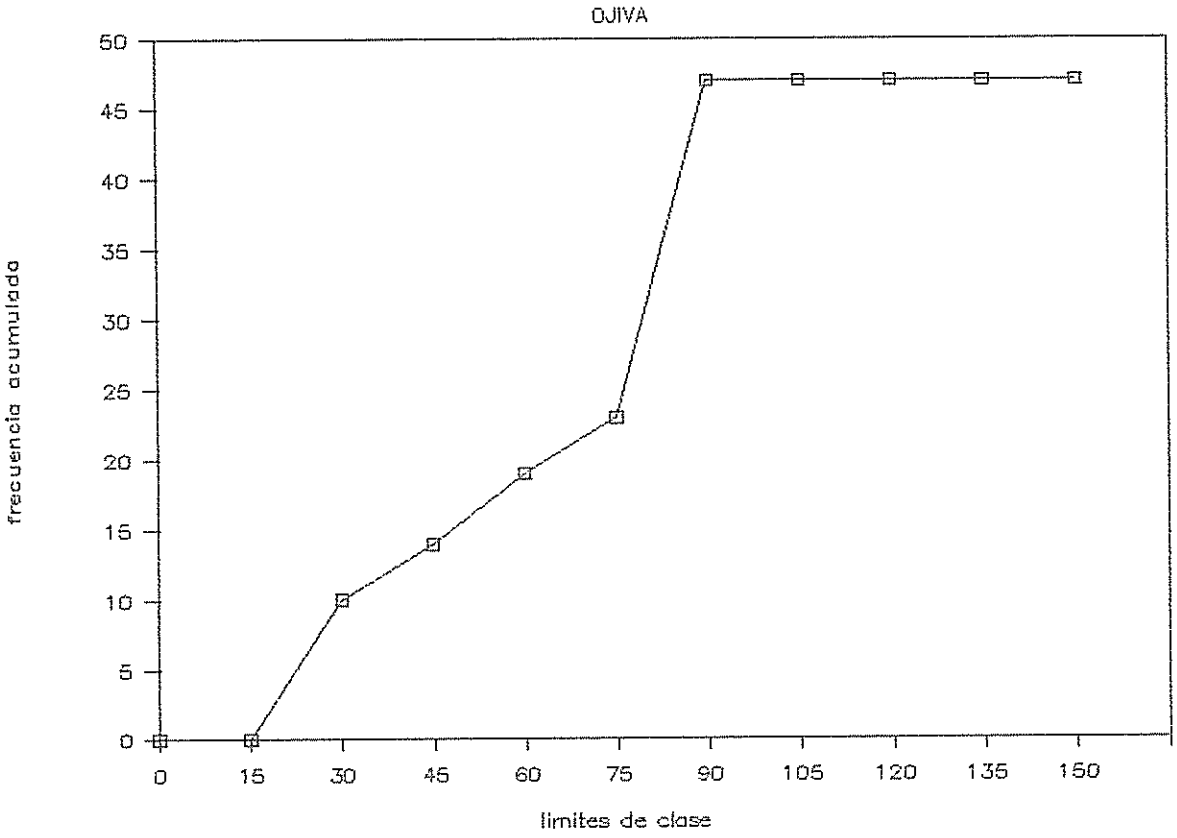
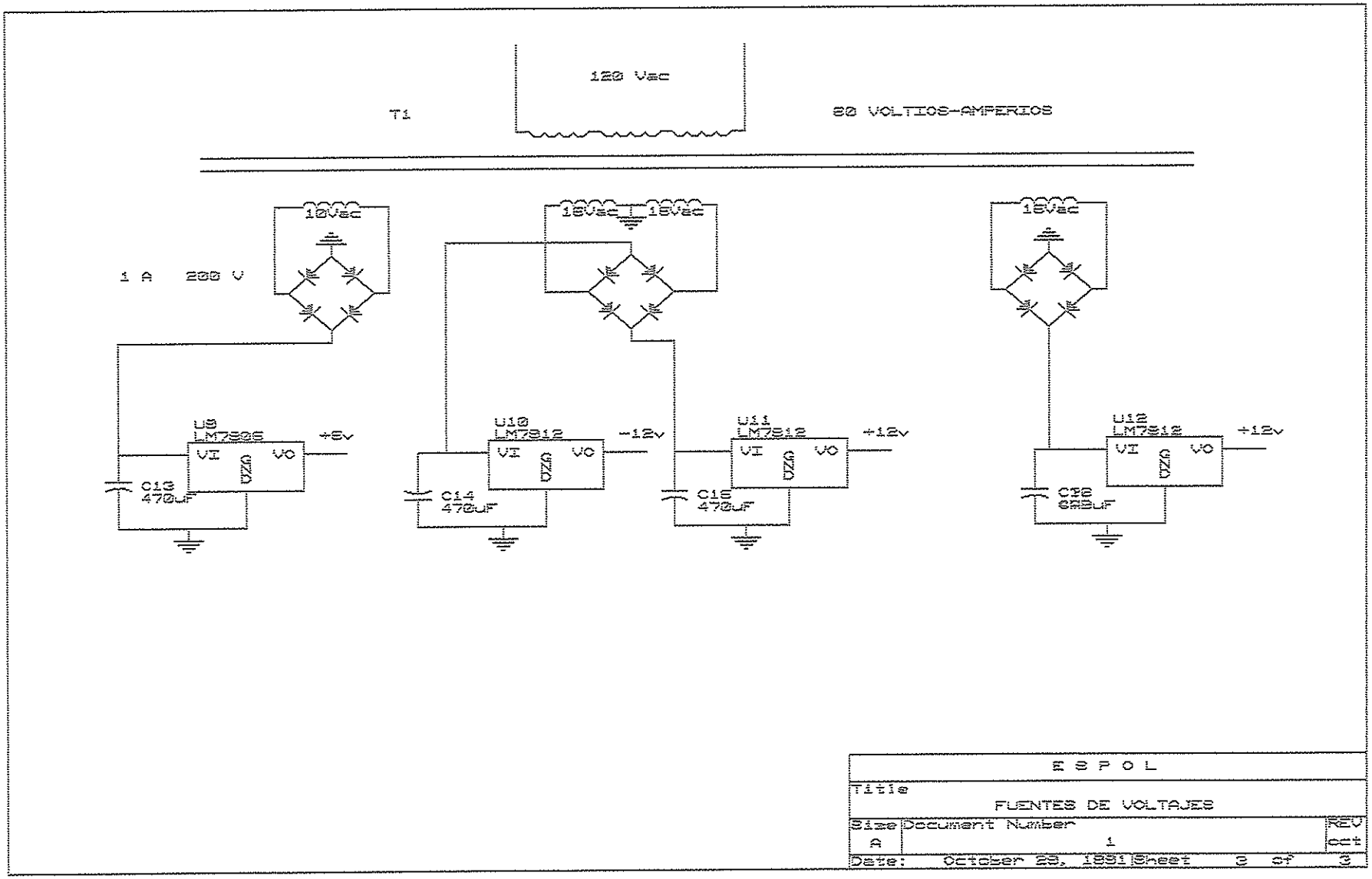

POLIGONO DE FRECUENCIA
frecuencia de clase
24
22
20
18
16
14
12
10
8
6
4
2
0
0
7,5
22,5
37,5
52,5
67,5
82,5
97,5
112,5
127,5
142,5
marcas de clase

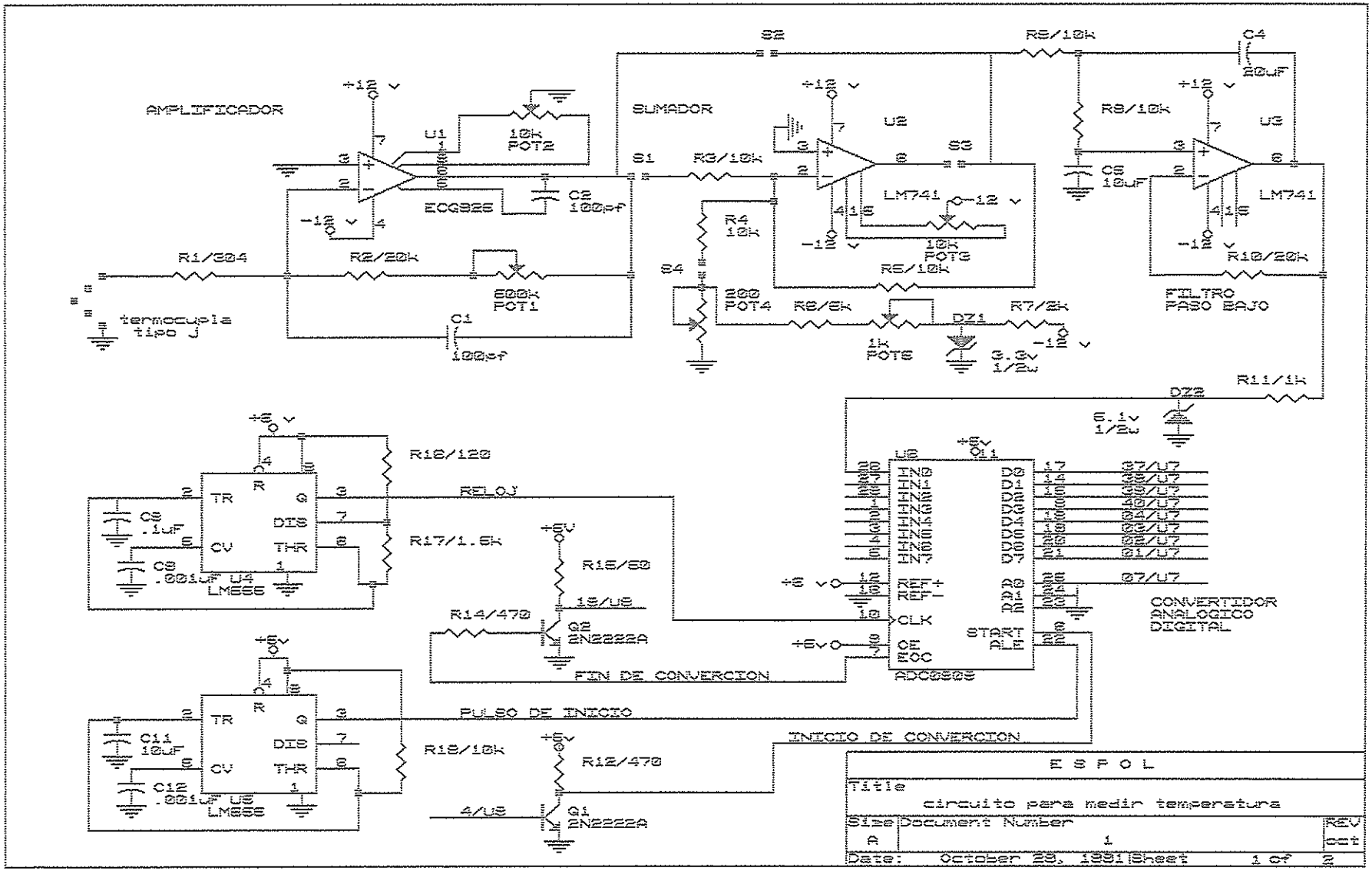
FIGURA 9.1—C


OJIVA
frecuencia acumulada
50
45
40
35
30
25
20
15
10
5
0
0
15
30
45
60
75
90
105
120
135
150
límites de clase

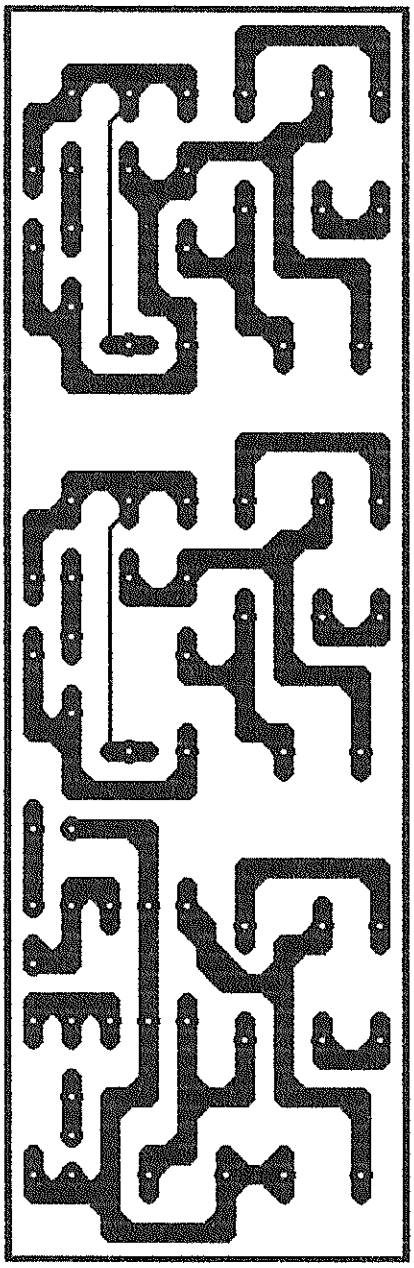


ESPOL		
Title: FUENTES DE VOLTAJES		
Size: A	Document Number: 1	REV: 1.1
Date: October 29, 1991	Sheet 3 of 3	

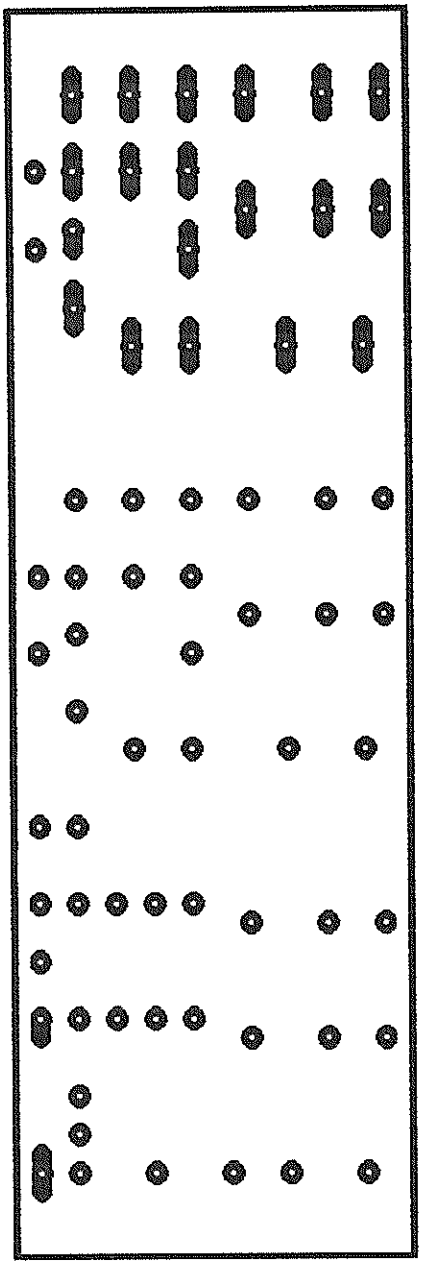
VA
VALVULA
RVA/2K
BA
BOMBA
RBA/2K
AG
AGITADOR
RAG/2K
CA
CALEN
TADOR
RCA/2K
SN1
NIVEL
DE
AGUA
RNI/2K
+5V
D8
D5
D4
D3
R24/470
R23/470
R22/470
R21/470
Q8
Q5
Q4
Q3
2N2222A
220Vac
120Vac
F1
F2
F3
F4
AGITADOR
BOMBA
VALVULA
CB1
15 AMP
CB2
CALENTADOR
FOCO
DE
SENAL
Q1 INICIO DE
CONVERSION
Q2 FIN DE
CONVERSION
U7
U8
ESPOL
Size
A
Document Number
Date: November 2, 1991 Sheet 2 of 3
REV
OCT


AMPLIFICADOR
SUMADOR
termocupla tipo J
R1/680
R2/20K
500K POT1
C1 100pf
U1 ECG926
10K POT2
C2 100pf
S1
R3/10K
S2
R5/10K
C4 20uF
U2 LM741
R4 10K
S4
1K POT4
R6/5K
1K POT5
DZ1 3.6V 1/2w
R7/2K
S3
10K POTG
R8/10K
U3 LM741
C5 10uF
R9/10K
R10/20K
FILTRO PASO BAJO
R11/1K
DZ2 6.1V 1/2w
U4 LM555
C6 .1uF
C9 .001uF
R16/120
R17/1.6K
RELOJ
R15/60
Q2 2N2222A
R14/470
FIN DE CONVERCION
U5 ADC0805
CONVERTIDOR ANALOGICO DIGITAL
CLK
START
ALE
OE
EOC
U6 LM555
C11 10uF
C12 .001uF
R18/10K
PULSO DE INICIO
R12/470
Q1 2N2222A
INICIO DE CONVERCION
E S P O L
Title
circuito para medir temperatura
Size A
Document Number 1
REV 0.01
Date: October 29, 1991 Sheet 1 of 2

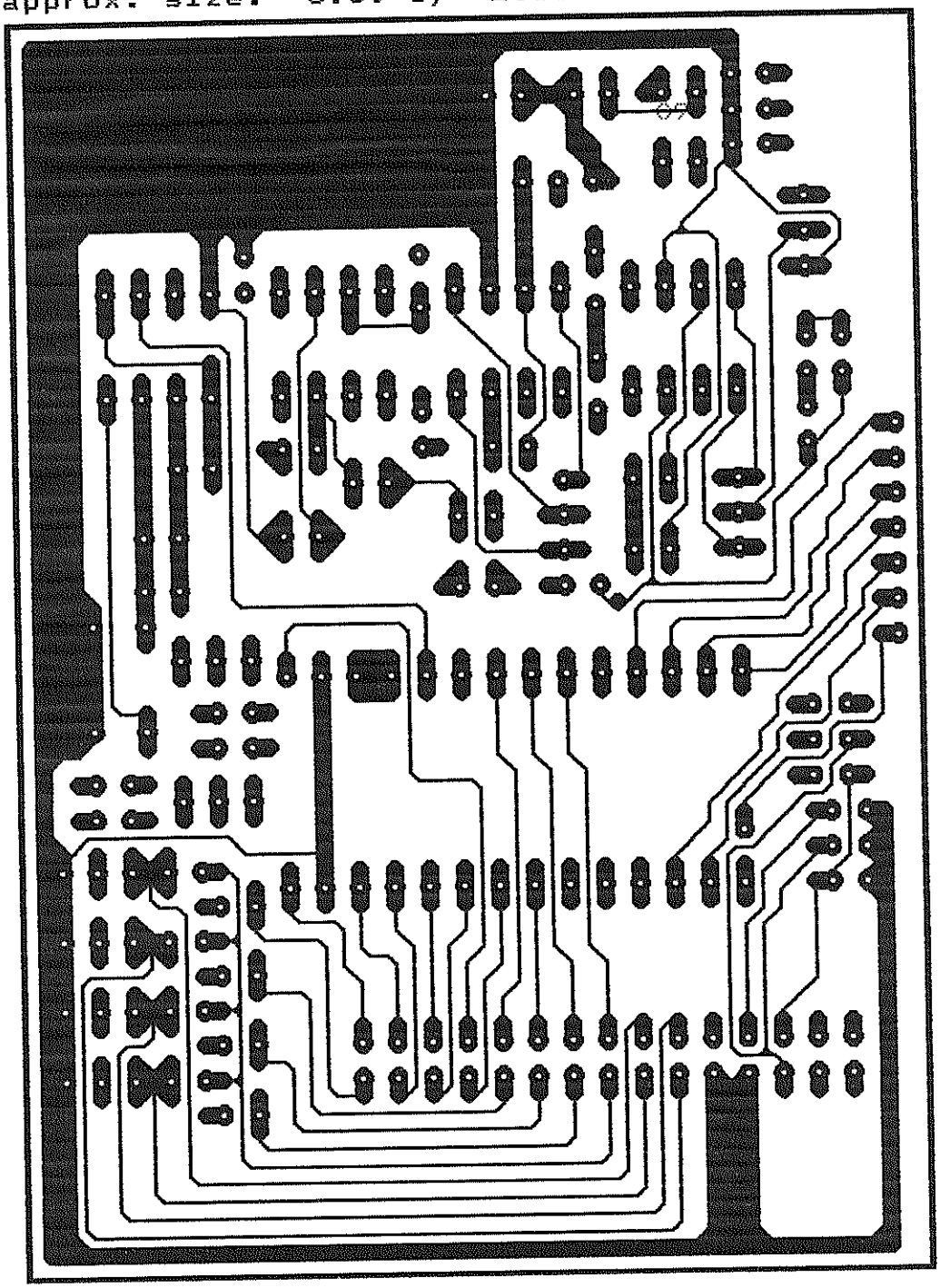
2x artwork v1.0 r1 9 Nov 1991 19:51:40
file: we layer: 0
approx. size: 3.15 by 0.95 in. holes: 69



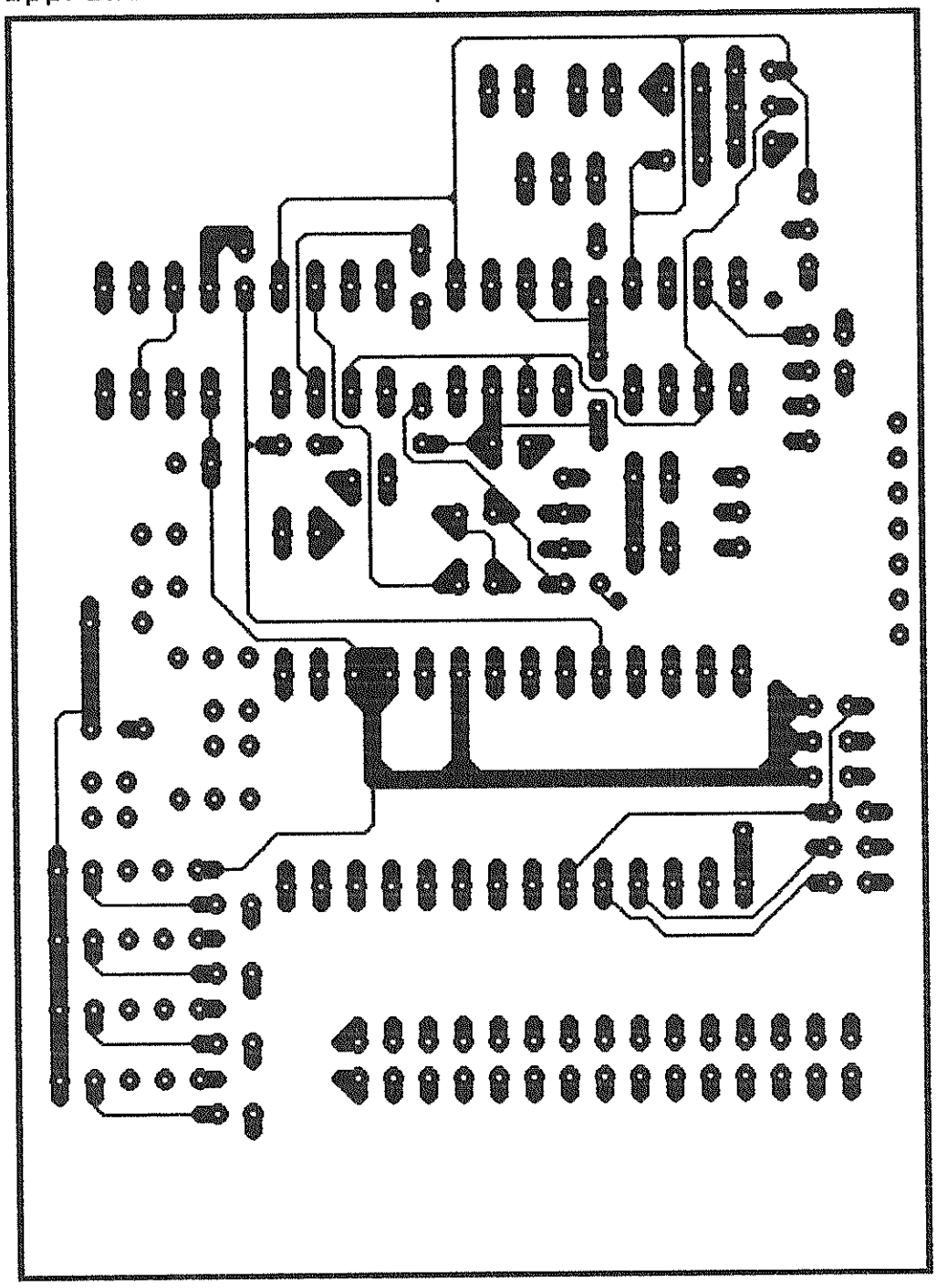
2x artwork v1.0 r1 9 Nov 1991 19:58:49
file: we layer: 1
approx. size: 3.15 by 0.95 in. holes: 69



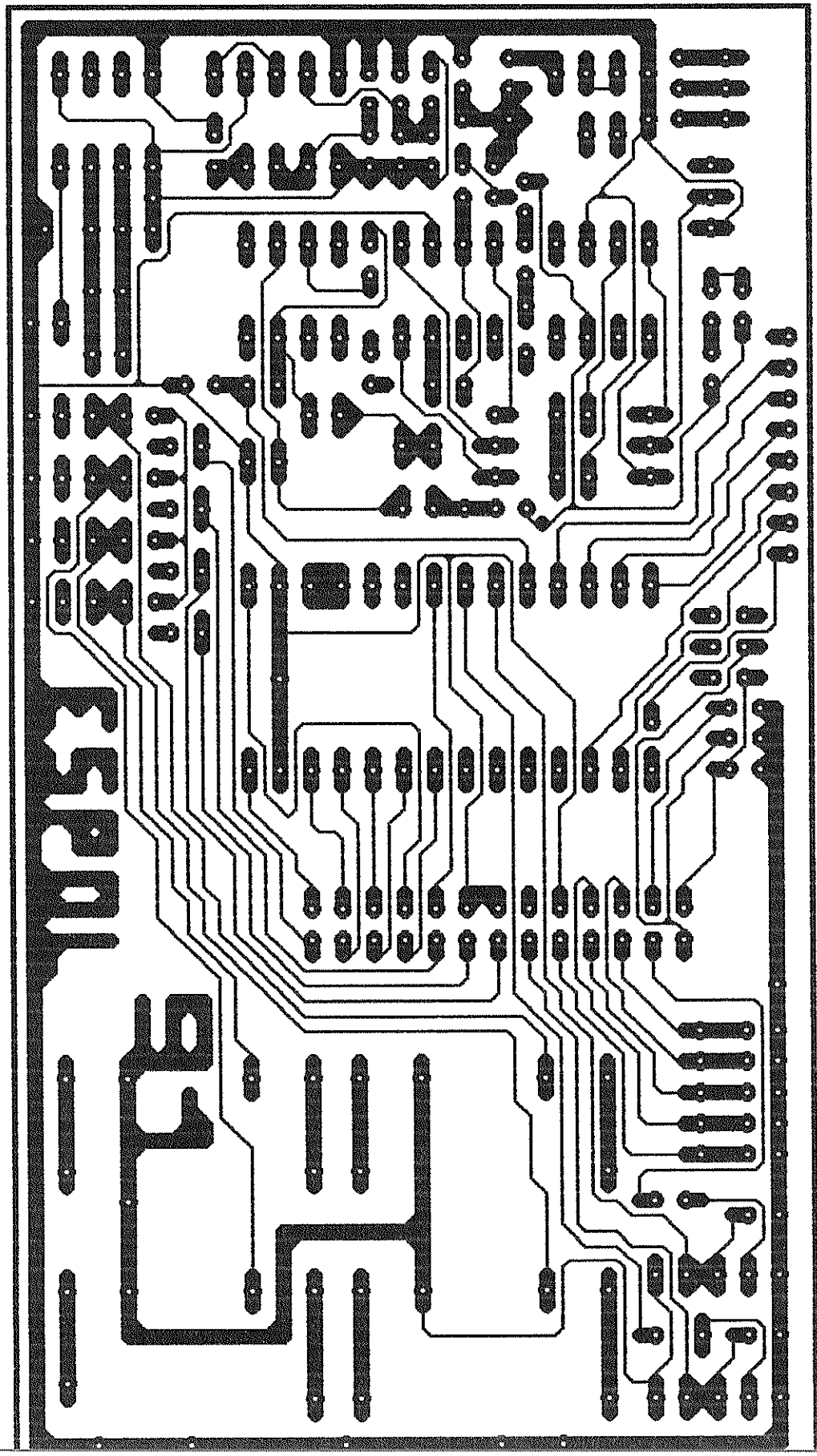
2x artwork v1.0 r1 9 Nov 1991 20:41:19
file: arte1.wrk layer: 0
approx. size: 3.50 by 2.50 in. holes: 222



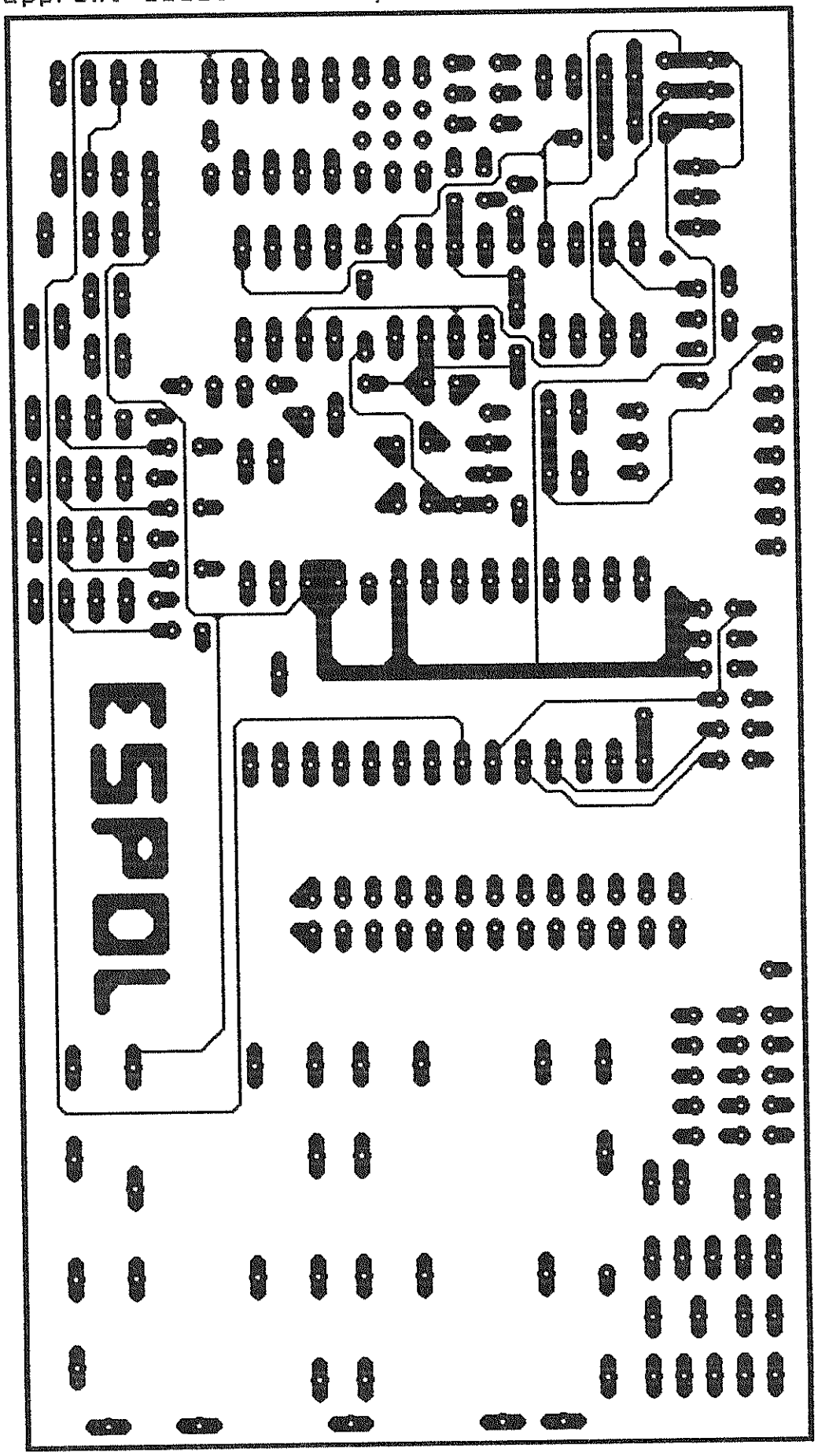
2x artwork v1.0 r1 9 Nov 1991 20:48:15
file: arte1.wrk layer: 1
approx. size: 3.50 by 2.50 in. holes: 222



2x artwork v1.0 r1 9 Nov 1991 18:12:52
file: arte2.wrk layer: 0
approx. size: 4.65 by 2.50 in. holes: 303



2x artwork v1.0 r1 9 Nov 1991 18:22:37
file: arte2.wrk layer: 1
approx. size: 4.65 by 2.50 in. holes: 303



D.- BIBLIOGRAFIA.

BIBLIOGRAFIA.

ASSEMBLER FOR THE IBM PC AND PC-XT	PETER ABEL „
ASSEMBLY LANGUAGE PRIMER FOR THE IBM PC AND XT	ROBERT LAFORE „
TURBO PASCAL	JEFF DUTENMAN „
PASCAL	PAUL CHRISTIAN „
DISENO DE COMPILADORES	AHO ULMAN „
REVISTAS FOX-BORO „	
MANUALES DEL MICROPROCESADOR 8080 „	
MANUAL DEL TURBO PASCAL	BORLAND„
MANUAL DEL TURBO C	MYCROSOFUELT „
MANUAL DEL TURBO C	BORLAND „
MANUAL DE TTL	INTEL „
PROBABILIDAD Y ESTADISTICA PARA INGENIEROS	MILLER FREUND „
ANALISIS NUMERICO	BURDEN FAIRES „
TURBO C BIBLE	NABAJYOTI BARKAKATI „
APUNTES GENERALES	