UNIVERSIDAD NACIONAL AUTONOMA
DE MEXICO

FACULTAD DE INGENIERIA

DISEÑO DE UN PROGRAMADOR INTELIGENTE DE DISPOSITIVOS LOGICOS PROGRAMABLES

T E S I S

QUE PARA OBTENER EL TITULO DE:

Y PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A:

M I G U E L R A M O S R E Y E S

DIRECTOR DE TESIS: ING EDMUNDO ROSALES VALDERRABANO

CIUDAD UNIVERSITARIA 1998

DISEÑO DE UN PROGRAMADOR INTELIGENTE DE DISPOSITIVOS LÓGICOS PROGRAMABLES

CONTENIDO

AGRADECIMIENTOS

Agradezco la ayuda de todas aquellas personas que confiaron en mi y me brindaron su apoyo incondicional, pero mi mayor muestra de gratitud es a **DIOS**.

JOSEFINA

¡GRACIAS A DIOS!

MIGUEL

Gracias a Octavio donde quiera que esté y a Miguel Angel porque siempre confiaste en nosotros.

Gracias papás

JOSEFINA Y MIGUEL

INTRODUCCIÓN

De acuerdo a la situación actual, el empleo de la tecnología de punta es el fundamento para el avance económico de los países. Hoy en día se llevan a cabo procesos industriales utilizando un sinnúmero de unidades automatizadas que minimizan costos, personal y tiempo. Con base al panorama anterior, en México surge la necesidad de preparar profesionales aptos para desarrollar proyectos que respondan a tales hechos; uno de los perfiles orientado a ello es a través de las instalaciones y apoyo académico que proporciona la Máxima Casa de Estudios, en particular la Facultad de Ingeniería. Un problema común que se presenta durante el desenvolvimiento académico al querer aplicar los conocimientos adquiridos, es la carencia o mal funcionamiento de equipo *que permita maniobrar la tecnología de vanguardia, de tal forma que* fomente la habilidad creativa del estudiante para que así obtenga una mayor calidad profesional.

Esta peculiar situación se observa en áreas donde se requiere programar dispositivos lógicos tales como: memorias, pal's y microcontroladores; ahora bien para efectuar tal hecho se necesita de un soporte lógico y físico que reúna ciertas características.

De las unidades disponibles en los laboratorios de la Facultad se aprecia que cuentan con programa ejecutable, su respectiva ayuda y además de una serie de archivos estáticos que contienen las especificaciones de los elementos que pueden ser programados. Cabe destacar que para aprovechar el potencial de estos programadores es preciso que manifiesten transigencia; es decir, que en cuanto al manejo de archivos concedan la actualización para incluir nuevos circuitos integrados que salen al mercado y conjuntamente modificar su estructura interna. Sin embargo, no es factible llevar a cabo estas posturas debido a que no se dispone del programa fuente así como del respaldo general del software. *Por otro lado, se ubica al hardware cuyo mantenimiento es mínimo dado a* que no se posee la información técnica del equipo. Tales circunstancias conducen a plantear dos conjeturas, la primera es la de tratar de cambiar

la organización de un sistema ya establecido, pero con la limitante de no tener la documentación estratégica y si a esto se le agrega que el emprender una reconstrucción implica una inversión de tiempo y dinero, entonces como premisa se descarta esta idea; en el segundo término se perfila el diseño de un programador argumentando un mejor rendimiento.

El presente trabajo describe el desarrollo del software y hardware de un Sistema de nombre PR-2000 que programa dispositivos lógicos de 1 a 40 pin's introduciendo patrones estandarizados de lectura/escritura proporcionados por el fabricante; además de poder actualizar sus archivos base. Cabe aclarar que para efectuar el diseño del equipo se tomó como referencia la unidad UP600, la cual se encuentra en la aulas de trabajo de la Facultad de Ingeniería.

Ahora bien, en cuanto a la distribución de este libro se contemplan los siguientes apartados:

En el primer capítulo se hace una recopilación de algunas características que manifiestan los sistemas que emplean software y hardware en su funcionamiento y que además se manejan dentro del ambiente estudiantil. Por ello encabezan la lista de opciones sobre las cuales se trazan persectivas que pudiesen ser integradas en el equipo PR-2000.

En el capítulo dos se hace una reseña sobre las singularidades que presentan los dispositivos lógicos programables tanto a nivel físico como operacional; además se entabla una descripción sobre las facultades que posee la unidad, así como lo relacionado al soporte lógico, desde su creación por medio del lenguaje de programación C++, hasta la elección de las herramientas necesarias para la implementación de las principales funciones que realiza la unidad durante una sesión de trabajo. También en este apartado se enfatiza el desarrollo de módulos que hacen más efectiva la tarea de mantenimiento y actualización; asimismo dada su naturaleza se decidió extender su uso en la parte complementaria del software.

El capítulo tres ofrece una descripción de los elementos que intervienen en una operación de lectura/escritura, y una vez definidos explica de que manera se fueron seleccionando los dispositivos electrónicos que en conjunto con el programa hacen posible la tarea encomendada. De ellos destaca el MC68HC11, puesto que es el centro de control y administrador de las etapas de trabajo.

La estructura modular del diseño permitio aplicar una serie de pruebas, las cuales arrojaron datos y observaciones de carácter técnico y económico; tales aspectos se encuentran detallados en el capítulo cuatro.

El último capítulo manifiesta la conclusión del desarrollo del Sistema PR-2000.

PREÁMBULO

Independientemente de los capítulos en que se encuentra dividido este libro, se ha creido conveniente hacer una pequeña pausa para mostrar el Sistema UP600, que como anteriormente se citó, fué la referencia que se tomó para armar el escenario del nuevo diseño.

Lo que aquí se presenta es una síntesis de su actividad, así como de sus principales integrantes.

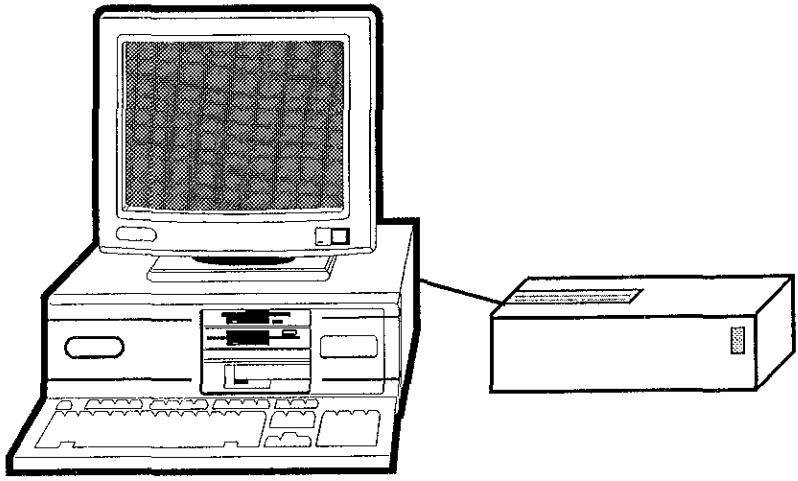
UN SISTEMA ACTUAL

La unidad tiene como enfoque global el verificar el borrado, la grabación y la comprobación de los datos introducidos en dispositivos de 1 a 40 pin's. De acuerdo a su funcionamiento y para comprenderlo mejor, se obró de una manera pertinente para lograr una división en módulos que hacen posible la ejecución de las tareas anteriormente mencionadas:

Módulo Informático

Módulo Auxiliar

ver Ilustración No. 1.



MÓDULO INFORMÁTICO *MÓDULO AUXILIAR*

Ilustración No. 1

Módulo Informático

A esta división del Sistema se le atribuye este nombre por el hecho de que trata de manera racional y automática la información, de esta forma se liga a la interfaz computadora-usuario. En una revisión sin pormenores se distingue que es amigable, realiza procesos específicos y se ocupa de la comunicación al exterior.

A partir de ahora se presenta una simple descripción de la operación; comienza con una pantalla de selección que da acceso a diversas bibliotecas:

MENÚ PRINCIPAL:

En este punto se hace referencia a las opciones que lo conforman.

SELECCIÓN DEL DISPOSITIVO: antes de realizar cualquier operación con el paquete, primero se debe elegir el dispositivo empleando este comando; el cual se encuentra dividido en dos partes: Lista de marcas y Número de elementos del fabricante.

COPIA RÁPIDA: se emplea para hacer la transferencia de datos de un circuito previamente grabado.

CARGA DE DATOS: realiza la captura de datos desde disco o de un dispositivo hacia la RAM.

PROGRAMACIÓN: permite copiar la información existente en RAM o en disco hacia el circuito integrado a programar.

VERIFICACIÓN: hace la comparación entre la fuente y el destino.

CHEQUEO DE BLANCOS: revisa que el chip esté limpio. En caso contrario emite un mensaje de error.

MÁS COMANDOS: incluye los siguientes menús:

Editor de datos
Transferencia de datos
Prueba de dispositivo
Calibración de la unidad.

Para que el sistema pueda realizar la carga de datos desde disco hacia la RAM es necesario que se auxilie de protocolos de conversión de datos.

El paquete contiene una interfaz de comunicación; ésta permite establecer y terminar una conexión entre dos estaciones (en este caso se trata de la computadora y el grabador), identifica el transmisor y el receptor, asegura que todos los mensajes se pasen correctamente sin

errores, y maneja las funciones de control involucradas en una secuencia de transferencia de datos.

Para establecer la vía de comunicación entre el equipo de cómputo y la unidad grabadora se utiliza una tarjeta controladora, ésta se encarga de preparar el ambiente propicio para que envíe la información en cadenas de bits con el siguiente formato: encabezado, datos y tren de datos.

Otra característica común que ofrece el sistema UP600 es la contar con un archivo de ayuda.

Una situación adversa que se estima dentro de la organización interna del paquete es lo estático de las relaciones esenciales que manejan los parámetros propios de los dispositivos lógicos programables.

Módulo Auxiliar

Como complemento al módulo informático el paquete UP600 dispone de una unidad que contiene circuitos que pueden realizar las siguientes acciones: grabación de los dispositivos, verificación de los mismos y carga de datos; para ejecutar dichas funciones se procede a capturar información referente a las características intrínsecas de los circuitos programables, una vez que se cuenta con los datos necesarios se establecen las áreas de trabajo, las cuales generan y manejan niveles de voltaje, tiempos, capacidades y señales.

A fín de que cierto aparato electrónico rinda al máximo, algunas veces proporciona referencia técnica para su sustento, pero desafortunadamente el paquete UP600 no dispone de ella, lo cual provoca deficiencias, que repercuten en su desempeño.

1
GENERALIDADES DE UN SISTEMA

Para poder estructurar el cuerpo de la unidad PR-2000 se hizo una revisión de las peculiaridades más comunes que suelen presentar los sistemas que regularmente se manejan dentro del ámbito estudiantil; ésto se llevó a cabo, con el fin de que algunas pudiesen ser incluidas en la organización del proyecto, sobretodo aquellas que ofrecieran accesibilidad a lo siguiente: colaboración para crear un ambiente de trabajo amigable, operatividad de archivos; es decir, que se permita la actualización y maniobras de acceso en el momento que se requieran, buena comunicación al exterior y tolerancia a la integración de una interfaz de soporte electrónico.

1.1. A NIVEL PROGRAMA

En la actualidad, el campo de la computación ha fructificado tanto que es capaz de brindar un sinnúmero de modalidades dinámicas como auxiliar en la creación de sistemas, por lo que comunmente al diseñarlos procuran evitar el tedio y rebuscamiento por medio de estructuras y funciones viables a la comprensión del usuario. Para ello normalmente recurren a los siguientes parámetros:

PANTALLAS: como son el primer contacto con el usuario debe contemplar aspectos visuales agradables que implementen un producto que combine:

Presentación
Diálogos con el usuario
Ayudas y
Salida.

ARCHIVOS: es un auxiliar al que más se recurre cuando se necesita considerar datos durante la ejecución de algún programa, debido a la versatilidad con que éstos manejan la información, es decir; como puede ser procesada-recuperada, almacenada o escrita; ya sea en un formato binario o en ASCII según se lo soliciten los resultados a la salida. Algunos formatos permiten llevar a cabo la operación de darles mantenimiento y/o actualización. La manera en que operan normalmente requiere de los siguientes pasos:

Nombre del archivo
Apertura
Escritura
Procesamiento
Lectura y
Cierre.

PROCESOS: es una parte medular de los lenguajes, porque ellos son capaces de organizar los componentes necesarios para realizar una tarea específica.

COMUNICACIÓN: es importante este aspecto debido a que se debe considerar la forma en que se transfieren los datos de la computadora hacia otros dispositivos.

Después de este recuento acerca de lo que un usuario generalmente encuentra en un sistema, se tomarán en cuenta esos aspectos como soporte básico dada la magnitud que éstos representan. Otra presencia de gran relevancia lo es sin duda el lenguaje, ya que sin él no se podría implementar ningún procedimiento; hay que tomar en cuenta que en estos momentos en el entorno computacional se encuentran una gran infinidad de ellos, pero de acuerdo a las razones que motiven a efectuar cualquier trabajo, siempre se va a buscar el lenguaje más adecuado que cubra las expectativas previstas.

Un porcentaje apreciativo tanto para el usuario como para el diseñador lo constituye la organización, su importancia radica en el hecho de que permite determinar los procedimientos a seguir e indica una jerarquía operacional. El desarrollo de la disposición del programa contempla niveles guiados hacia contextos de ahorro de recursos; como tiempo de procesamiento, ejecución y comprensión que se puede ver como una consecuencia del punto anterior. Todo ello es una proyección específica y por lo tanto con frecuencia agregan algún tipo de verificación de aceptación sobre los datos de entrada en módulos decisivos del paquete, con el fin de detectar posibles fallas.

1.2. A NIVEL CIRCUITO

Los circuitos integrados han ampliado por completo su uso y se han convertido en un producto cuya función y propósitos básicos son comprendidos en casi todas las facetas de la electrónica, incluyendo comunicaciones, control, instrumentación y computación; este extendido empleo se debe a su bajo costo, su tamaño reducido y sobre todo a que realizan operaciones elementales.

De acuerdo a las pretensiones de la aplicación se pueden crear sistemas construidos por medio de los circuitos MSI, de tal manera que tienen la ventaja de que se ajustan a las necesidades de una aplicación particular, sin embargo, podría requerir de un gran número de elementos.

Algunos diseños pueden realizarse usando LSI y VLSI, los cuales pueden ser microprogramados para adecuarse a especificaciones solicitadas; su organización usa menos CI que la configuración anterior.

Anteriormente se citó el empleo de tarjetas controladoras, no obstante que su uso se limita a la computadora en donde se encuentra instalada y si por alguna circunstancia el usuario desea trabajar en otro equipo lo obliga a realizar la maniobra de desinstalación/instalación de la tarjeta en la máquina, arriesgándose a encontrar incompatibilidad de

componentes además de posible falta de pericia en el manejo de la misma.

1.3. PROPUESTAS

Para lograr un buen principio en el planteamiento y construcción de proyectos, el diseñador deberá evaluar con todo cuidado qué es exactamente lo que va a hacer el sistema y cómo lo va a efectuar, para alcanzar esta meta nada resulta más conveniente que la descomposición modular o diseño estructurado; al realizarlo de esta manera se anteponen las bases de una organización adecuada. Para conseguir la unificación de esta idea, a continuación se expondrán las partes del sistema que se pretenden abarcar.

1.3.1. Organización

Durante el desarrollo del software se procurará mantener una línea de trabajo que se apegue a condiciones tales como: la administración de recursos, la efectividad de las tareas ejecutadas y la prevención de posibles errores. Racionalmente también se le adicionará lo siguiente: en relación a la composición interna del programa no hay nada mejor que eludir extravagancias que son prescindibles y sólo tienen un efecto negativo en su estructura. Los parámetros operacionales que se establezcan se acatarán con responsabilidad y precisión.

Con una buena documentación es permisible coordinar apropiadamente el proceso de operación y a su vez simplificar instrumentos que eleven la capacidad del programa.

Por último, para poder alcanzar una integración total de todo lo anterior, resulta oportuno buscar entre el soporte de cómputo existente lo más acorde a las necesidades planteadas.

1.3.2. Lenguaje

En cuanto a la estructura del programa se supone que lo constituirán una serie de elementos necesarios que contribuyan a realizar un ambiente de trabajo placentero. Para poder disponer de ello se recurre a estos factores:

Presentación :

Dentro de las expectativas que debe contemplar ésta se puede mencionar lo subsecuente: que sea sencilla, amena, con objetivo y explícita. Con el fín de evitar hastío y confusión se recurren a auxiliares de tipo visual; es decir, diversidad de formatos de letras, efectos de pantalla, sonidos y figuras. Todo ésto en conjunto representa un importante complemento en cualquier tipo de presentación.

Manejo de archivos :

Se pretende almacenar los datos mediante un manejo de listas; también se busca aptitud para modificaciones; es decir, que se puedan llevar a cabo actualizaciones. En el caso de la ayuda, comunmente se guarda en archivos de tipo texto, esto se hace con el fin de obtener una copia impresa del contenido del mismo.

Detección de errores :

A manera de resguardo de la información que maneja un programa, éste debe contemplar un mecanismo para la detección de errores, con ello evitará redundancias y posibles pérdidas.

Protección :

Los programas deben contener dispositivos que rastreen anomalías en los datos que se estan procesando para proteger tanto al usuario como al equipo.

Después de que se presentó el esqueleto fundamental, se volcará la atención hacia algunos lenguajes de programación y paquetes que en determinado momento interactuen como apoyo. Tomando en cuenta lo expuesto, es necesario explorar los lenguajes, sobretodo aquellos que

aporten las condiciones más óptimas de implementación de pantallas, menús, cajas de diálogos, comunicación y ventanas de ayuda.

1.3.3. Transmisión de datos

Para entablar el enlace computadora-usuario hoy en día se tienen al alcance una gran variedad de prototipos que lo hacen, pero con la práctica se ha visto que incluir cajas de diálogos contribuyen a la fluidez y vialidad del paquete, así como una buena comprensión del mismo.

Ya se ha visto la comunicación entre la computadora y el usuario ahora bien; toca el turno al enlace entre la computadora y el entorno exterior. Una de las formas es mediante tarjetas controladoras y la otra es empleando el puerto RS-232. Al hacer una conjetura se visualizó la preferencia hacia el puerto ya que reporta circunstancias complacientes al diseño, puesto que es más versátil en cuanto a uso, documentación y adaptación.

Por otro lado, como precepto común la información que se maneja en un programa se encuentra en formato binario, ésto va a repercutir en el modo de transferir los datos entre la computadora y el grabador, por ende es un factor que debe considerarse durante la selección de los componentes electrónicos; dado lo anterior y visando a futuras modificaciones facultativas que se le pudiesen hacer al sistema, se recomienda el empleo de circuitos digitales atendiendo a sus dos clases: elementos de memoria y lógicos.

1.3.4. Implementación con circuitos

Ahora se discutirá sobre el hardware apropiado y en especial aquel que cubra las expectativas del sistema en general, puesto que se piensa dar un tratamiento específico a la información de tipo binario, por lo que

conviene enfocar de una manera cuidadosa el uso de los circuitos electrónicos.

A manera de incorporar las ventajas que ofrece la microelectrónica en equipos actuales, es importante mencionar algunas de ellas: disminuye considerablemente el número de CI que componen una unidad, puede ser programada, expandida para mejoras y dado a que se dispone de la documentación suficiente se le puede dar mantenimiento. Algunos de los protagonistas son el microprocesador y el microcontrolador.

1.4. NOTAS CONCLUYENTES

El material que se presentó en este capítulo deberá de servir como una introducción a los tópicos del diseño del grabador. Se ha hecho hincapié en la diversidad de elementos que quizás formen parte del equipo, pero se les dará una atención especial a aquellos que se encuentren disponibles y con costos accesibles, además de que *respondan a las necesidades planteadas; de igual forma, se anticipa* la utilización de un microprocesador o un microcontrolador.

2
FASE I DEL DISEÑO

Para comprender el ejercicio del UP600 y respondiendo a una serie de inconvenientes de tipo informativo y operacional, se procedió a efectuar una partición de acuerdo al funcionamiento correspondiente a los entornos de software y hardware. Esta acción repercutió de una forma determinante en la manera de cómo comenzar a trabajar sobre el diseño del nuevo equipo de nombre PR-2000; razón por la cual es el nombre de Fase I.

Ahora bien, antes de abundar sobre el software del Sistema, se hará un paréntesis en donde se expondrán los factores y características de los dispositivos lógicos programables que se considerarán en la elaboración del Sistema PR-2000.

2.1. SINGULARIDADES DE LOS CHIP'S PROGRAMABLES

La primer tarea que se llevó a cabo fue el recopilar manuales de especificaciones, en ellos como es bien conocido se encuentran las peculiaridades individuales de los dispositivos electrónicos. Aunque existe una gran gama de empresas que se dedican al ramo de la fabricación de chip's, sólo se tomaron aquellas cuyos compendios se lograron encontrar (Intel, Motorola, Hitachi, Texas Instrument y National); de ahí que se tomó una muestra de 36 dispositivos programables.

Al revisar las hojas de especificaciones correspondientes se efectuó un estudio y una clasificación basada en las operaciones básicas que se pueden ejecutar con los chip's (lectura y escritura). Para iniciar se estructuraron tres campos:

El primero lo integran las propiedades más generales que presentan los chip's en cuestión:

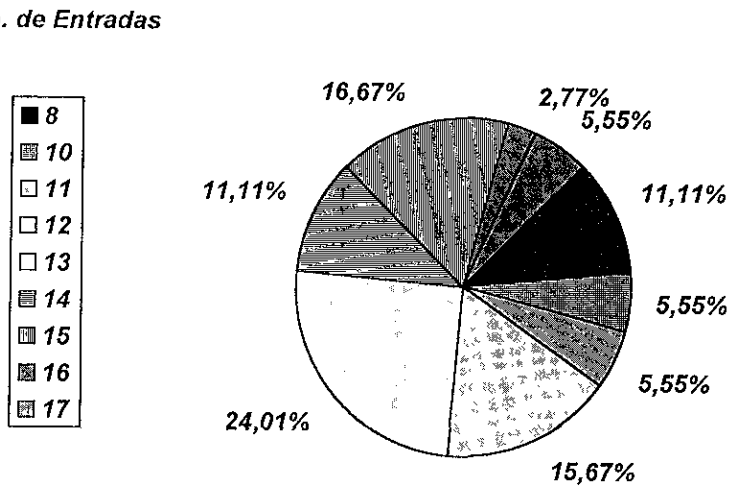
Características Físicas.-

a) Son identificados por una clave que incluye números y letras que a su vez relaciona a la familia y tipo de tecnología a la que pertenecen y capacidad de almacenamiento.

b) El número de pin's más frecuentes son: 20, 24, 28, 32 y 40.

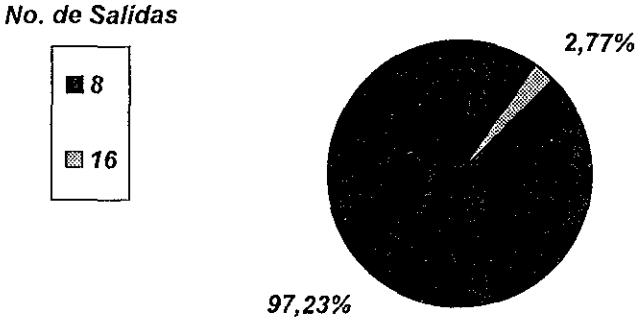
Características internas.-

Los datos de los siguientes incisos estan basados en las gráficas obtenidas de la muestra de los 36 elementos; de ellos se observó que la frecuencia del número de entradas (direcciones) mostraban las proporciones que se indican en el esquema 2.0:



Esquema 2.0 Frecuencia de número de entradas

y el número de salidas mostraban las proporciones señaladas en el esquema 2.1.

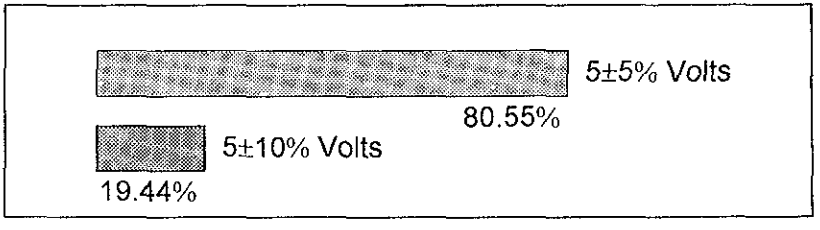


Esquema 2.1 Frecuencia de número de salidas.

Entonces se definió que:
c) Entre 8 a 16 oscila el número de entradas.

d) El número de salidas por lo regular es de 8.

El segundo campo incluye las condiciones que requieren y presentan los chip's durante un proceso de lectura y se obtuvo lo que indica el esquema 2.2.



Esquema 2.2 Frecuencia de niveles de voltaje durante un proceso de lectura.

Por lo que se determinó que:

e) Entre 4.5-5.5 volts oscila el voltaje de polarización durante un proceso de lectura.

Cabe destacar que se necesitan controlar algunas terminales específicas que permiten habilitar al chip para que trabaje en el modo de lectura; por otro lado, el comportamiento que presentan sus entradas para mostrar un valor binario (V_I=-0.5 a 0.8 volts) para un acoplamiento de salida (V_O=0.4 a 3.5 volts) quedan determinados por el tipo de dispositivo.

El tercer campo reune a los elementos que se requieren durante un ciclo de programación:

f) Un voltaje de alimentación de 5 a 6.5 volts.

g) Voltaje de programación de 10 a 25 volts según sea el dispositivo programable.

h) El pulso de programación es muy variado dependiendo del modelo del chip.

Como nota final cabe destacar que todos estos parámetros sirvieron como base para estructurar las fases que conforman el programa y el circuito del grabador. A continuación se explicarán las características, diseño e implementación de todo lo relacionado con el software.

2.2. ESQUEMA GENERAL

En esta sección se explicarán las áreas funcionales del PR-2000. Para comenzar se presentará una visualización de las peculiaridades que éste reporta en comparación con la referencia tomada para la realización de este trabajo:

1. Organización modular

Se puede decir que es un buen principio que hoy en día se ha convertido en la base de cualquier implementación de *software* y hardware, ya que permite en el caso del soporte lógico estructurar el programa de una manera adecuada y generar el código fuente así como el ejecutable; mientras que en el otro aspecto se favorece a la ubicación de los elementos que pertenecen a zonas determinadas; de esta manera cuando se desee modificar algo se recurre directamente al lugar específico.

2. Documentación

La práctica ha demostrado que una unidad que cuente con su respectiva documentación es más rentable; sin embargo, este útil requisito el UP600 no lo reune; por lo que se considera como uno de los aspectos relevantes que debe cubrir este proyecto.

En general, la documentación que se maneja proporciona una descripción completa para ayudar en la fase de operación y mantenimiento. Se anexa a esta idea la particularidad de que es conveniente contar con los antecedentes en lenguaje nativo pues, la traducción de una lengua a otra puede generar ideas encontradas o incomprensión en el tema tratado. El usuario podrá consultar tal escrito con hacer el llamado desde la computadora al programa manual.doc.

Otro factor que puede colaborar en forma determinante, es el hecho de que a partir del diseño de cada faceta del grabador, se iban realizando las pruebas correspondientes por medio de pequeños programas, los cuales estarán incluídos en el manual para poder ajustar o encontrar las posibles fallas a nivel circuito.

3. Diagramas

Este punto es complemento de lo anterior, pero se desligó de él dada su importancia en el momento de presentar información vital del funcionamiento y estructura interna del Sistema.

4. Actualización

Otra particularidad que no cubre la referencia tomada es la de no poder actualizar sus archivos, en especial los que contienen un conjunto de datos y especificaciones imprescindibles para efectuar cualquier operación básica, ya sea de lectura y/o escritura; esta situación se superará con el nuevo diseño.

Aunado a lo anterior se establece una relación con un factor implícito en el desarrollo de un proyecto: el costo. Dada la importancia que éste representa, se consideró la necesidad de economizar los mayores recursos posibles, recurriendo para ello al material que se fue adquiriendo durante la estadía estudiantil así como de la tecnología digital y de otro tipo de material.

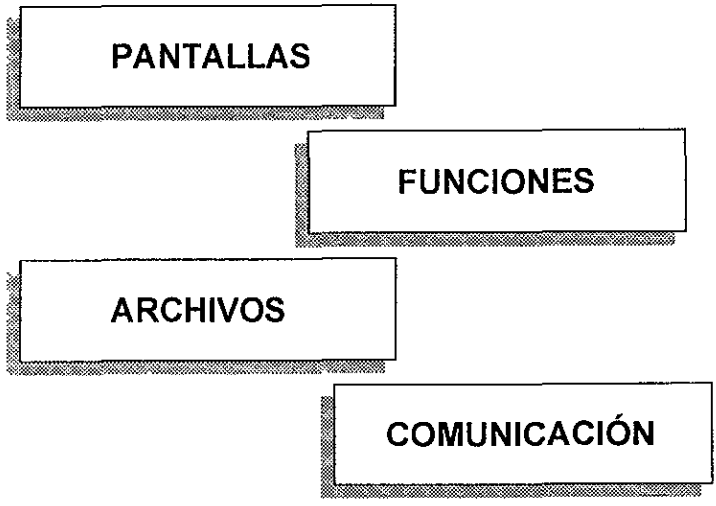
Después de esta revisión de argumentos se dirige ahora la atención al funcionamiento del proyecto, para comenzar; se mencionan las tareas que puede efectuar en los dispositivos lógicos programables, como lo son: carga, lectura, grabación y verificación de datos, además de poder actualizar los archivos.

Para efectuar todo lo anterior, se necesita de un programa que tenga la capacidad de organizar y estructurar todas las funciones y operaciones necesarias además de apoyarse en una interfaz de hardware.

2.2.1. Software

Una vez que se definieron las operaciones que integran el esqueleto del programa, se discernió entre los posibles lenguajes de programación que reunieran ciertas características, tales como: disponibilidad del compilador y sus correspondientes manuales, portabilidad entre equipos, versatilidad en la implementación de funciones y compatibilidad; como resultado de esta búsqueda se encontró que el mejor candidato que conjuntaba todo lo anterior era el lenguaje C++; sin embargo, de las versiones existentes se seleccionó aquella de la marca Borland (Borland

C++ V. 2.0) debido a su amplia biblioteca de gráficos. El esquema 2.4 presenta al ambiente de trabajo representado por bloques.



Esquema 2.4 Ambiente de trabajo del Sistema PR-2000.

2.2. Pantallas

Todas las aplicaciones gráficas tienen en común la propiedad de que son fáciles de leer, así como la de hacer puntos, líneas y cuadros; situación que provoca que éstas se encuentren muy arriba de la lista de cosas excitantes que se pueden llevar a cabo con el apoyo de una computadora. Dado lo anterior las pantallas que conforman el sistema PR-2000, estan creadas por medio de gráficos debido a la facilidad y fluidez con que operan. Los tipos básicos de pantallas que se utilizan son las siguientes:

PRESENTACIÓN:
Como todo sistema requiere de mostrar su nombre y contenido, este programa recurre al uso de diferentes tamaños y formas de letras, figuras geométricas, colores y tonalidades diversos para resaltar títulos y notas importantes para el usuario.

MENSAJE:
El empleo correcto de estas pantallas está orientado hacia el manejo adecuado del paquete, puesto que proporciona al usuario información relevante acerca de la operación de la unidad y del dispositivo a trabajar.

MENÚ:
Es la entrada al entorno de ejecución de las diferentes tareas incluidas en el Sistema.

DIÁLOGO:
Son aquellas que permiten al usuario interactuar con la computadora; por lo general, se enfocan a habilitar etapas de captura de datos o recepción de mensajes que son decodificados como instrucciones, es conveniente incluir estas pantallas en cualquier programa debido a que facilitan el diálogo en el ambiente de trabajo.

2.2.3. Archivos

El archivo representa una de las estructuras de entrada-salida cuyo empleo es imprescindible en cualquier programa, generalmente se utilizan mensajes de ayuda para guiar al usuario en el manejo de algún paquete y una de las formas de guardar información es mediante un archivo; de éstos existen dos formatos de almacenamiento: texto o ASCII y binario.

Una arteria importante para la fluidez del programa la conforman los archivos de tipo texto; en particular los de ayuda, éstos se encargan de auxiliar en un momento determinado al usuario durante la corrida del sistema, también se tomó en consideración la posibilidad de que si alguien desea tener una documentación acerca del programa la puede obtener

por medio de una impresión. En cuanto a campos relacionados con información técnica se vió que la mejor opción son los archivos binarios, puesto que brindan la posibilidad de manejar internamente los datos así como el enviarlos al exterior.

2.2.4. Funciones

Las funciones que realiza cualquier sistema son el resultado de la organización y asignación de las tareas que lo componen; ahora bien, un lenguaje de programación representa el medio por el cual se integran las instrucciones necesarias para que el usuario pueda llevar a cabo su objetivo.

En el proyecto PR-2000 se han establecido una serie de estructuras operacionales que en conjunto formalizan la presentación, operación, actualización y comunicación. Cabe mencionar que toda función responde a una jerarquía previamente establecida, de esta forma se evitan la duplicación de actividades y posibles errores durante la corrida.

2.2.5. Comunicación

Para llevar a cabo el enlace de comunicación el Sistema se apoya en la interfaz RS-232-C, la cual representa el periférico más comercial y de mayor uso, por ende implica el desarrollo de un programa específico para su manejo y operación.

Para realizar este proceso se recurrió al lenguaje ensamblador, éste permite activar y accesar las direcciones de los registros que se involucran directamente así como el establecimiento de los parámetros de comunicación; es decir, la longitud de la palabra (8 bits), el bit de stop (1 bit), el bit de paridad (ninguno) y la velocidad (1200 bps).

Cabe destacar que durante la transmisión se requiere de cierta sincronización entre la emisión y recepción de datos, aspecto que es cubierto por el programa.

2.3. PROGRAMA

El conjuntar los apartados anteriores para lograr un fin determinado, sugiere la creación de programas que sinteticen las acciones que lo integran, por tanto; conviene hacer una pausa para revisar el diagrama 1; el cual representa la columna vertebral que sostiene a todo el sistema, en él se encuentran contenidos los nombres de cada segmento de código lógico así como la secuencia de trabajo.

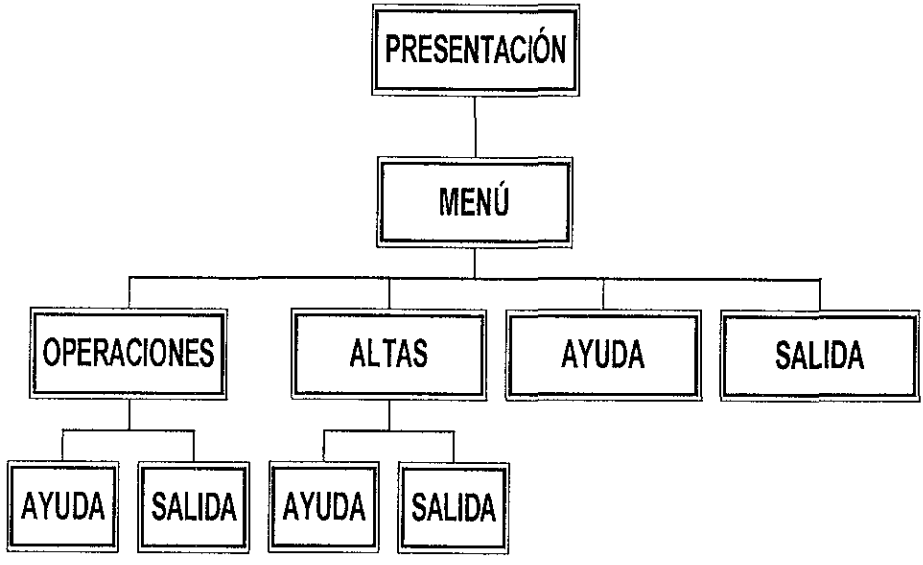


Diagrama 1 Diagrama a bloques del Software del Sistema.

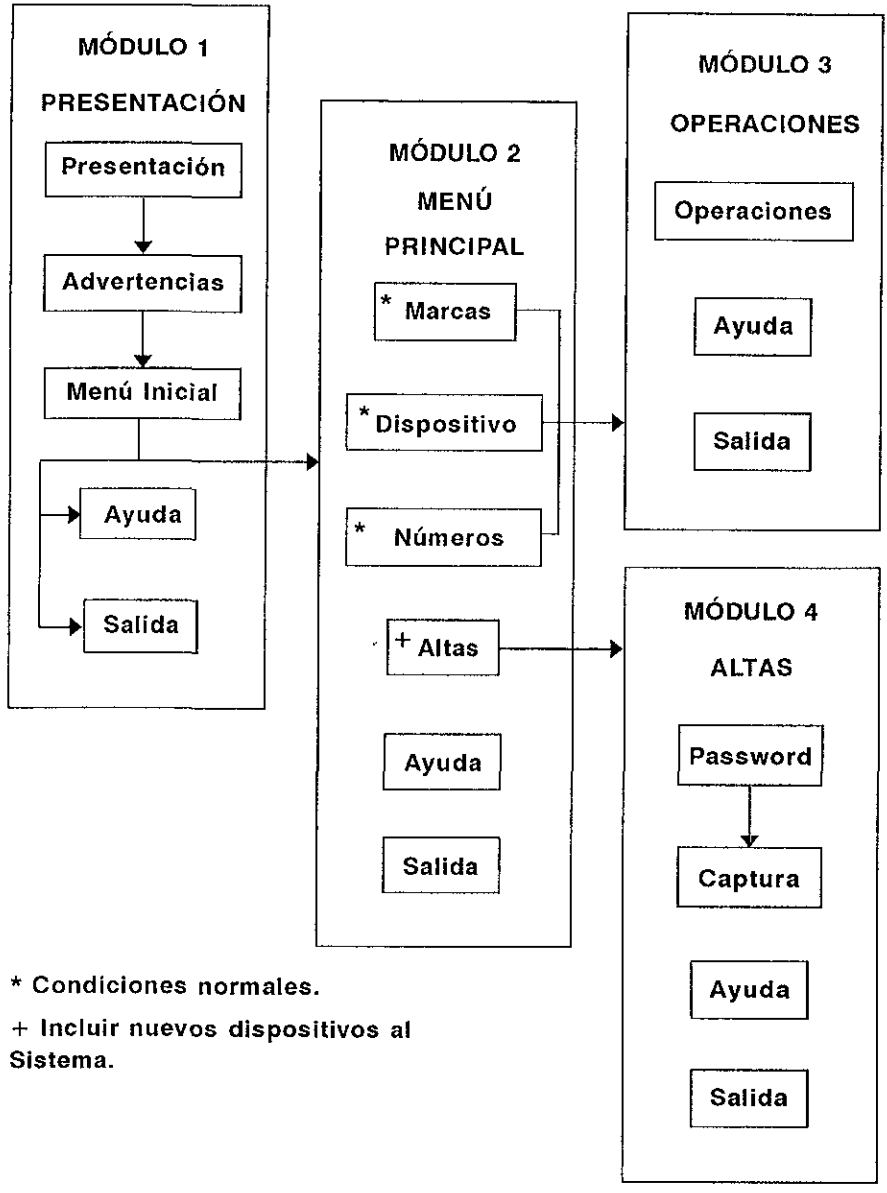
Como puede observarse son 3 etapas básicas que siguen un cierto orden: la primera parte comprende el dar a conocer el nombre y modo de trabajo de la unidad PR-2000, además de permitir la entrada a la segunda fase, la cual consiste en presentar un menú que señala las dos principales arterias y su salida va a depender de la opción que el usuario elija. El tercer nivel lo constituyen los modos en que puede operar el equipo: uno es en condiciones normales; es decir, en el caso de contener todos los datos necesarios para realizar cualquiera de las siguientes labores: carga, lectura, grabación y verificación de datos de un dispositivo lógico programable y la otra forma se refiere a la ausencia de dichas especificaciones, pero con la ventaja de poderlos incluir en los archivos, para así poder trabajar en circunstancias iguales posteriormente.

Para que el lector pueda concebir con mayor claridad el contenido de cada área expuesta, se dispone a indicar los elementos que constituyen a cada programa (esquema 2.5).

2.3.1. Presentación

El Módulo 1 (Presentación) contiene las funciones necesarias para poder entablar la comunicación entre usuario-computadora y computadora-programador; ahora bien, para llevar a cabo la primer interfaz se utilizan implementaciones con pantallas y cajas de diálogos las cuales contienen las leyendas e indicaciones pertinentes. Las instrucciones de la biblioteca de gráficos juegan un papel importante debido a que permiten la generación de código que reditúa en los resultados requeridos.

El segundo tipo de comunicación se encuentra vinculado con el complemento electrónico cuya función es la de indicar el estado del programador (encendido o apagado). Una descripción más detallada del Módulo 1 la encontrará en su correspondiente diagrama de flujo (ver página 20).



Esquema 2.5 Elementos que constituyen a cada programa.

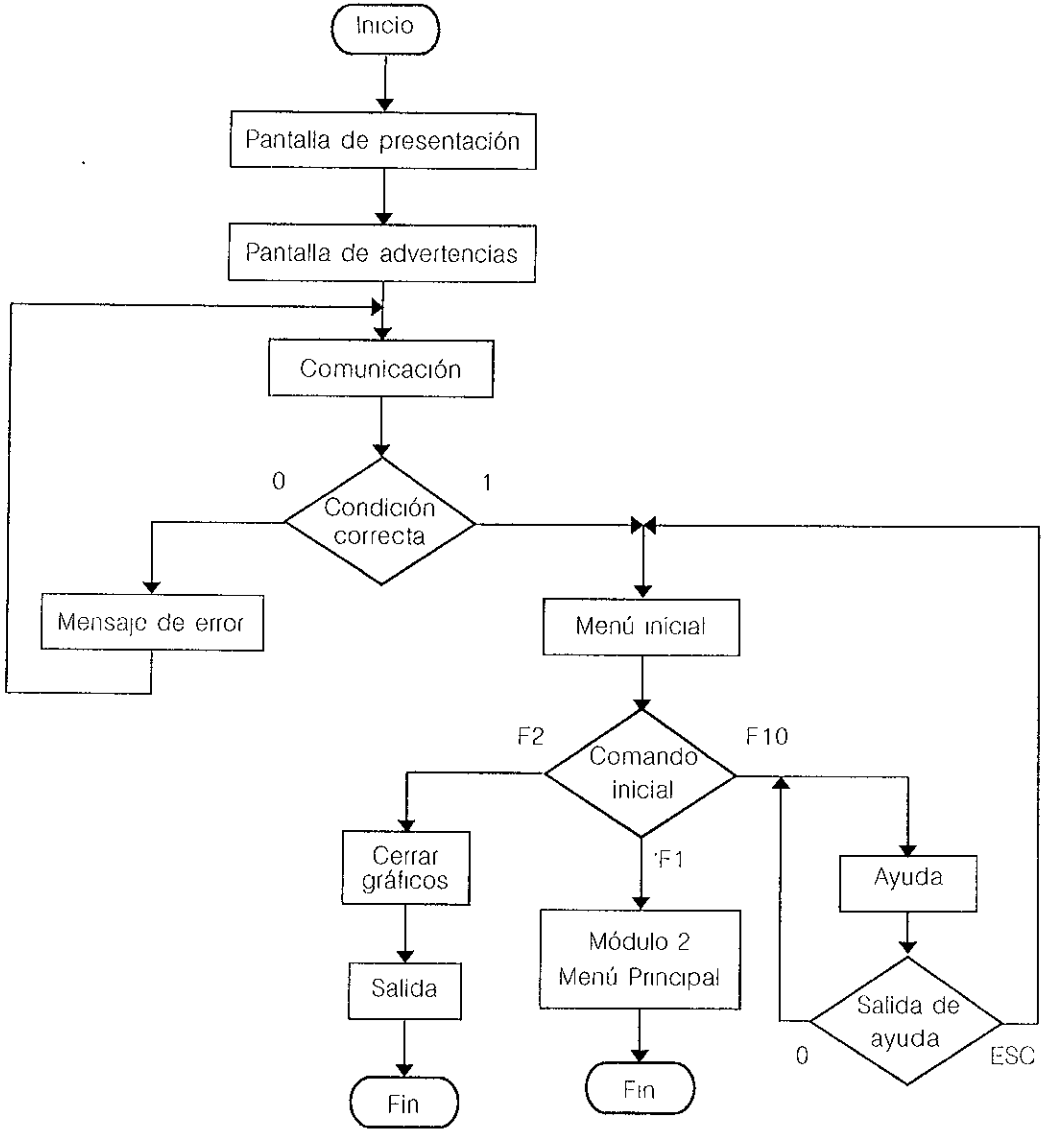


DIAGRAMA DE FLUJO DEL MÓDULO 1.
[PRESENTACIÓN]

PSEUDOCÓDIGO

La Pantalla de Presentación constituye el primer contacto del usuario con el paquete, es por ello que debe contener las funciones adecuadas para lograr en su receptor una sensación de amabilidad y no dar muestra de que se trabajará en un ambiente hostil; la aplicación de estructuras gráficas permite la integración de dos factores importantes para cualquier persona relacionada con el ámbito de la computación; uno, el que no fatigue la vista con la presencia de tonos o colores brillantes y otro, el que no se presenten imagenes monótonas, a fin de lograr este objetivo se cuenta con las funciones initgraph, line, rectangle y settextstyle, todas ellas contenidas en la biblioteca BGI (Borland Graphics Interface) del lenguaje C++ de la compañía Borland, el siguiente bloque implementa la pantalla de presentación:

initgraph(&manejador grafico, modo grafico, "\\BGI");
habilita las funciones de gráficos
rectangle(x, y, x1, y1);
rectangle(x2, y2, x3, y3);
conforman el marco de la pantalla
settextstyle(1, 0, 6);
selecciona el tipo de letra, tamaño y orientación
outtextstyle(150, 200, "PR-2000");
coloca el texto en la pantalla.

De igual forma que la pantalla anterior y recurriendo a las mismas estructuras del lenguaje se implementa la Pantalla de Advertencias. Ambas locaciones sólo consumen tiempo y su duración es determinada por la estructura *delay(5000)* donde 5000 representa 5 segundos.

Posterior a la acción de esta pantalla se cuestiona sobre el complemento del software; es decir, sobre el circuito, es importante que éste se encuentre activado pues en caso contrario se emite un mensaje de error, sólo que ahora la duración se restringe a esperar una señal.

En la Pantalla de Mensaje de Error se estableció una estructura capaz de emitir bits de control hacia el circuito y aguardar una referencia, de esta forma se libera la pantalla y continua la ejecución del programa.

El código que permite el primer contacto entre el grabador y la computadora se resume en las siguientes líneas:

```
# define int 0X83
# define pto 0X0
# define sonido 7
# define archivo "verifica.s19"
  estos parámetro asignan los códigos de inicialización del puerto serie.
envia_puerto(){
  rutina que manejará la comunicación con el MC68HC11
inicializacion();
  abre el puerto serie
leelista();
  guarda el código del archivo S19
envialista();
  envía el código al MCU.
```

En esta etapa se logra que el puerto de comunicación *(COM1)* esté habilitado en la transmisión, el archivo con extensión S19 contiene el programa en ensamblador para HC11 y la última línea *(envialista();)* transmite los datos.

```
if(bandera==1)
  goto otro_intento;
else
  goto continuacion;
```

la bandera permite determinar si la transmisión se ha logrado. En caso afirmativo se repite la acción, en caso negativo se transmite un indicador para hacer referencia a la recepción completa, tal y como se muestra :

```
aviso = 1;
transmite(aviso);
```

```
if(bandera == 1)
  goto otro_intento;
    se pregunta acerca de la recepción en el MCU
else
  goto termina_proceso;
```

se observa que en caso positivo se repite todo el proceso, ésta es una forma de asegurar que la etapa receptora se encuentra habilitada.

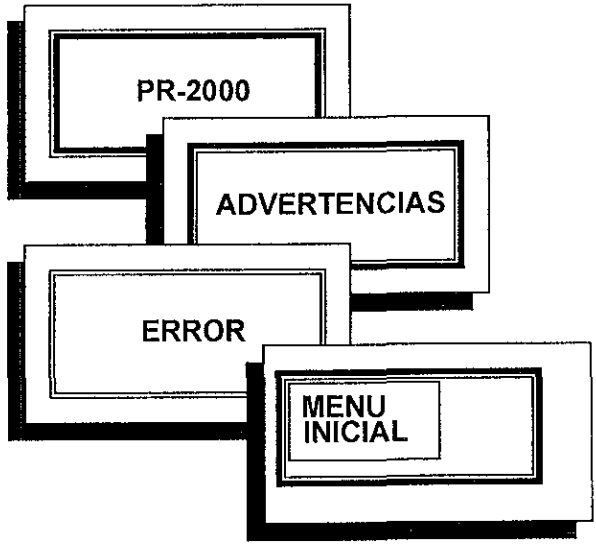
Para finalizar la comunicación se dispone a cerrar el circuito, es decir; si se inicializó y transmitió por el puerto serie, resta recibir un dato; para ello se utiliza lo siguiente:

```
limpia_puerto();
recibe();
if(regs.h.al == 1)
  exit(0);
else
  goto nuevamente;
```

la función *(limpia_puerto();)* permite vaciar el contenido del puerto y dejar la opción de recibir un dato, por otro lado; en *(recibe();)* se lee el puerto serie y en la parte baja del registro A se puede observar el dato transmitido por el microcontrolador.

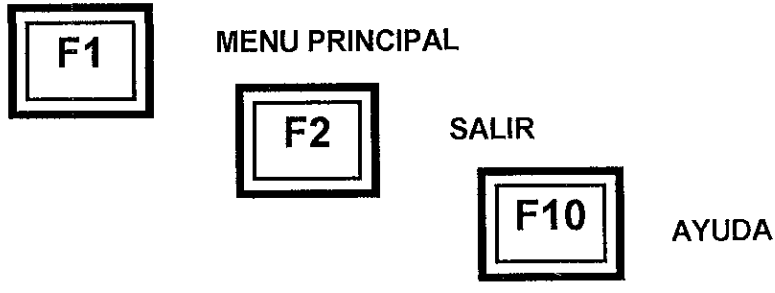
La aplicación de esta estructura es un medio eficaz para proteger al usuario del funcionamiento erróneo del equipo.

Resumiendo la secuencia de las pantallas del Módulo 1, observamos que primero se activa la Pantalla de Presentación del Sistema, posteriormente se le indica al usuario acerca de la necesidad de verificar que las conexiones esten correctas e inclusive se da la oportunidad de corregirlas; y una vez que todo está listo para trabajar se ingresa a la Pantalla de Menú Inicial (ver esquema 2.6).



Esquema 2.6 Pantallas del Módulo 1.

El Menú Inicial esta integrado por figuras geométricas y letreros, en ella se determinan tres opciones (ver esquema 2.7); es decir, constituye la primera etapa de actividades del Sistema PR-2000.



Esquema 2 7 Teclas que permiten realizar las Operaciones del Menú Inicial

A continuación se anexa un bloque de su código:

```
rectangle(20, 20, 620, 460);
  marco de la pantalla
settextstyle(x, y, "MENU INICIAL");
settextstyle(x, y, "F1  MENU PRINCIPAL");
settextstyle(x, y, "F2  SALIR");
settextstyle(x, y, "F10 AYUDA");
  opciones
switch(opcion){
  case 'F1' : pantalla_menu_principal();
  case 'F2' : exit(1);
  case 'F10' : pantalla_ayuda();
          leer_otra_opcion;
                cuerpo del Menú y acceso al otro bloque del programa.
}
```

Mientras la función espera la tecla de comando el circuito permanece en estado de espera hasta recibir una señal. Cabe destacar el empleo de teclas de comando (F's) las cuales sirven de nemónicos que agilizan la actividad.

Como puede apreciarse la estructura switch permite dirigir la ruta a seguir: en el caso de haber elegido F1, se continua la ejecución del programa mediante la aparición de una nueva pantalla que contiene otro menú (Menú Principal); por otro lado, si el usuario desea consultar la sección de Ayuda deberá accionar la tecla F10.

El definir un segmento de ayuda en este software cubre un aspecto importante debido a que orienta al usuario en su labor, en él se contempla la implementación de ventanas que son el resultado de un manejo conjunto de gráficos y archivos de tipo texto. La estructura básica de la Pantalla de Ayuda es la siguiente:

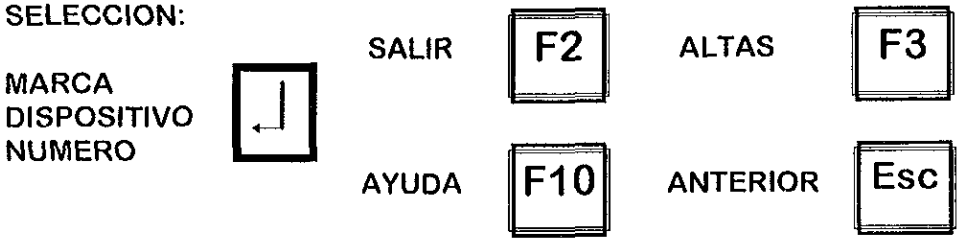
```
rectangle(20, 20, 620, 460);
settextstyle(x, y, "AYUDA");
```

pantalla de ayuda
fp=fopen("ayuda.wp","r");
apertura del archivo
ayuda(opcion);
despliegue del contenido del archivo
fclose(fp);
cierre de archivo.

Este bloque es general y se emplea durante todo el programa. Una vez que el usuario decide salir de la Ayuda, sólo tendrá que pulsar la tecla de Esc para regresar a la pantalla de Menú Inicial. Por último, si elige la opción F2, ésta le dará la salida al sistema MS-DOS concluyendo la sesión de trabajo.

2.3.2. Menú Principal

Una vez que se decidió efectuar una sesión de trabajo se ingresa al Módulo 2 (Menú Principal); en él se encuentran los comandos necesarios para la selección de los dispositivos lógicos programables; así como, la posibilidad de realizar la actualización de los archivos de datos y otros (esquema 2.8). Ver su diagrama de flujo en las páqinas 27 y 28.



Esquema 2.8 Comandos del Menú Principal.

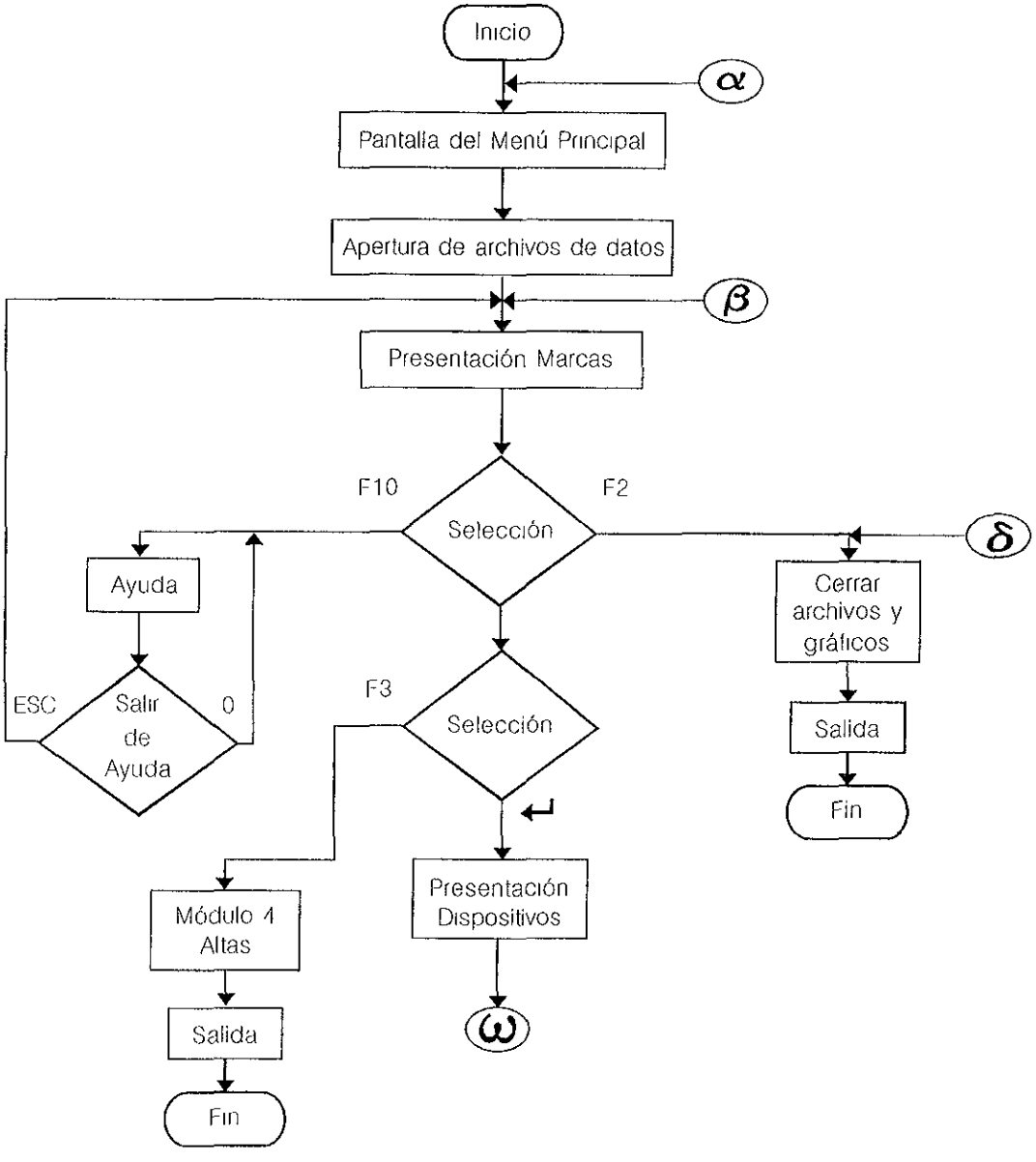


DIAGRAMA DE FLUJO DEL MODULO 2

[MENÚ PRINCIPAL]

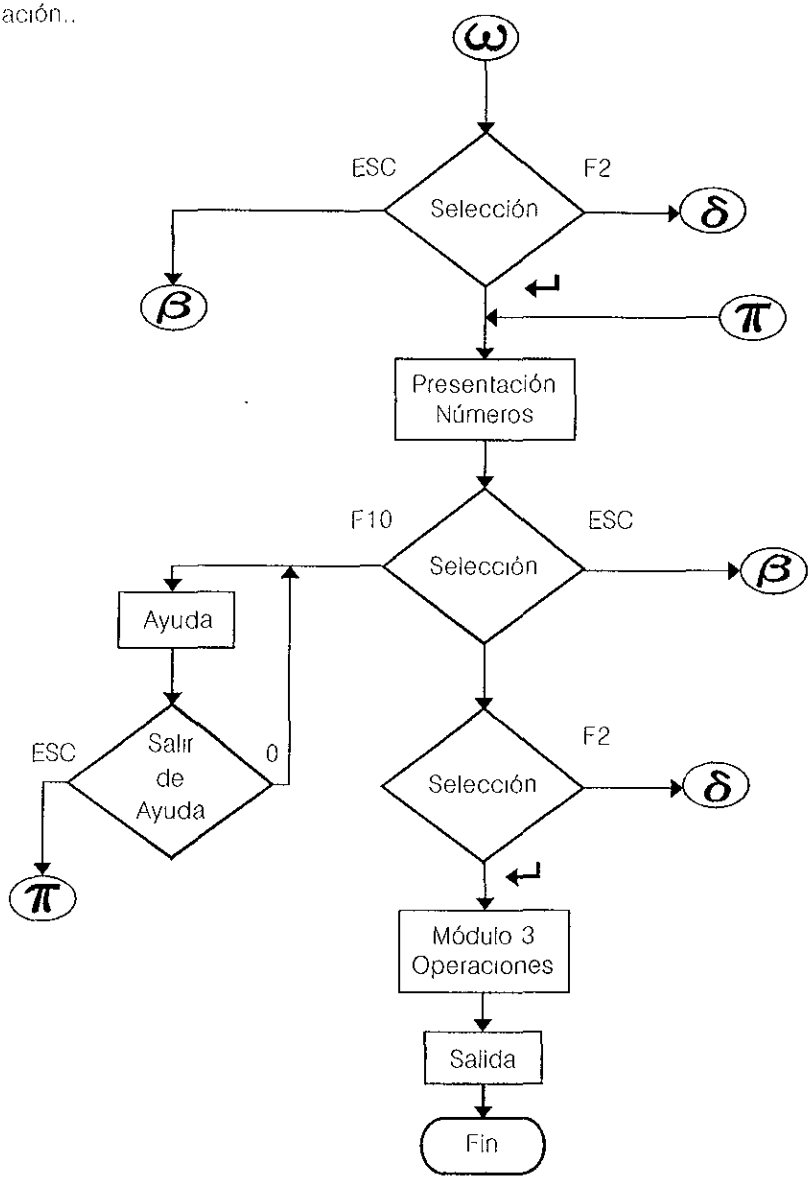


DIAGRAMA DE FLUJO DEL MODULO 2
[MENÚ PRINCIPAL]

PSEUDOCÓDIGO

Las actividades del Menú Principal se agrupan en un bloque de interacción tal y como se observa en el diagrama de flujo, la primera acción la constituye el procedimiento de la creación de la Pantalla del Menú Principal, en la cual se establece el ambiente de trabajo por medio de la biblioteca de gráficos y estructuras básicas de programación (ciclos, iteraciones, selecciones y decisiones).

Una vez que se muestra la pantalla se realiza la lectura del archivo de datos, empleando para ello las siguientes instrucciones:

```
fp=fopen("datos.dat", "r");
```
apertura de archivo para lectura
```
fread(fp, struct lista_datos, 1);
dato1=lista_datos.dato1;
dato2=lista_datos.dato2;
```
lectura de datos y asignación a variables de trabajo
```
fclose(fp);
```
cierre de archivo para evitar pérdidas de información.

Posteriormente la información contenida es importada hacia la pantalla de trabajo. A partir de que son presentados todos los elementos que constituyen el Menú Principal, se procede a elegir la operación deseada. El segmento de código empleado es:

```
switch(opcion){
    case 'F2'    :    cleardevice();
                      closegraph();
                      exit(1);
    case 'F3'    :    altas();
    case 'F10'   :    pantalla_ayuda(opcion);
    case 'return' :   temporal1 = lista_datos.marca;
                      goto seleccion_dispositivo;
```

```
    case 'flecha_abajo':    setfillstyle(1, 2);
                            bar(x1, y1, x2, y2);
                            y1 = y1 + 10;
                            y2 = y2 + 10;
                            creación del cursor y aumento de coordenadas
                            setfillstyle(1, 15);
                            bar(x1, y1, x2, y2);
                            cont = cont + 1;
                            seleccion = getch();
                            captura de la selección
                            if(seleccion==return){
                              temporal1=lista[cont].marca
                                asignación de la marca
                              break;
                            }else
                              goto otra_vez;
    case 'flecha_arriba':   setfillstyle(1, 2);
                            bar(x1, y1, x2, y2);
                            y1 = y1 - 10;
                            y2 = y2 - 10;
                            creación del cursor y aumento de coordenadas
                            setfillstyle(1, 15);
                            bar(x1, y1, x2, y2);
                            cont = cont - 1;
                            seleccion = getch();
                              captura de la selección
                            if(seleccion==return){
                              temporal1=lista[cont].marca
                                asignación de la marca
                              break;
                            }else
                              goto otra_vez;
    default             :   goto otra_vez;
}
```

Si el usuario elige F2 se dan por terminadas las actividades saliendo al Sistema Operativo MS-DOS. F3 constituye el acceso al mayor atractivo del Sistema PR-2000, puesto que permite ingresar al bloque de Altas para poder actualizar los archivos de datos y de esta forma ampliar el campo de acción del programador.

Para escoger una de las Marcas desplegadas en la ventana se auxiliará de los movimientos de cursor (flecha hacia arriba y flecha hacia abajo) y del return; cabe destacar que al ejecutar este acto ya no podrá ingresar al Módulo 4 (Altas).

Después el cursor se posicionará automáticamente en la ventana de selección de Dispositivos. Tanto este bloque como el siguiente (elección del Número de Dispositivo) presentan una estructura muy similar a la anterior.

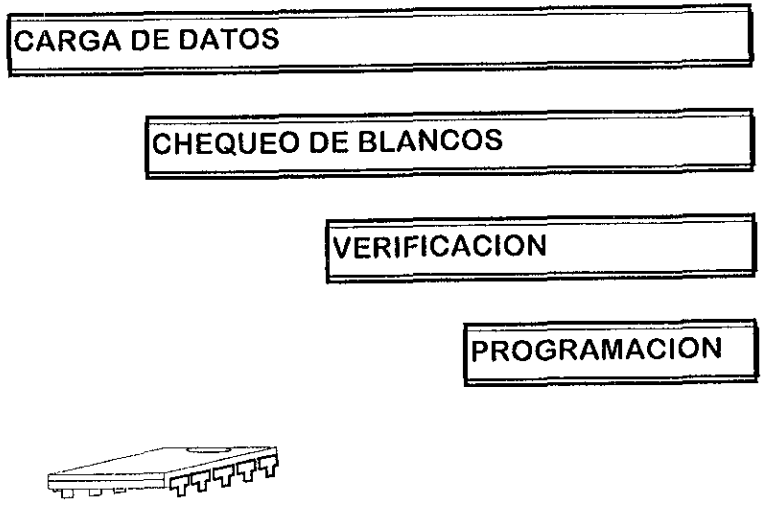
Su correspondiente bloque de programa es el siguiente:

```
switch(opcion){
  case 'F2'            : cleardevice();
                         closegraph();
                         exit(1);
  case 'return'        : temporal2=lista_datos.dispositivo;
                         ventana_dispositivos();
                         selecciona el dispositivo y coloca en la otra
                         ventana
  case 'esc'           : retorno_menu_principal;
  case 'flecha_abajo'  : se crea el cursor de acuerdo a las coordenadas
  case 'flecha_arriba' : de la ventana que contiene los dispositivos
                         empleando el pseudocódigo anterior
                         seleccion = getch();
                           captura de la selección
                         if(seleccion==return){temporal2=lista[cont].dispo-
                         sitivo
                           asignación del dispositivo
                           break;
                         }else
```

```
                    goto otra;
    default         :  goto otra;
}
```

2.2.3. Operaciones

Al ingresar al Módulo 3 (Operaciones), se presentará un ambiente integrado por pantallas de captura, cajas de diálogos y auxiliares gráficos. Para llevar a cabo la implementación de esta parte del Sistema se consideraron los parámetros de lectura/escritura que fueron mencionados al inicio de este capítulo. Esta sección del programa contiene un menú con las operaciones que puede realizar el Sistema (esquema 2.9).



Esquema 2.9 Operaciones del Módulo 3

Carga de datos:
Este comando permite la introducción de datos desde teclado, por lo general se tratarán de programas que el usuario previamente habrá codificado en hexadecimal o binario; para realizar esta actividad se auxilia de una pantalla en forma matricial que admite la información en el formato acorde a cada tipo de dispositivo.

Chequeo de blancos:
Esta opción tiene como propósito hacer una exhaustiva revisión de todas las localidades de memoria para verificar que no existan elementos grabados; en caso contrario, se emite un mensaje al usuario con el objetivo de que tome las medidas correctivas.

Verificación de datos:
Su función básica radica en el hecho de cargar desde el dispositivo en uso la información contenida en él, retenerla en memoria RAM y compararla con la contenida en el archivo de datos, de esta forma es posible detectar si existe un error en el programa y por medio de un mensaje visual y auditivo la persona que se encuentra en la sesión trabajo podrá saber si en realidad los datos mostrados en pantalla están contenidos en su circuito integrado.

Programación:
Es una de las etapas más completa y compleja de la Pantalla de Operaciones, pues en ella se realiza la recopilación de la información capturada, la prepara para su envío, hace el llamado a un manejador de puerto y manipula la comunicación entre la computadora y el grabador.

En este nivel el usuario esta en posibilidad de elegir alguna de las opciones anteriormente mencionadas, dependiendo de su selección el programa transferirá el control a la etapa indicada, por consiguiente el grabador ejercerá su trabajo.

El diagrama de flujo que corresponde a éste Módulo 3 es el que se muestra en las páginas 34 y 35.

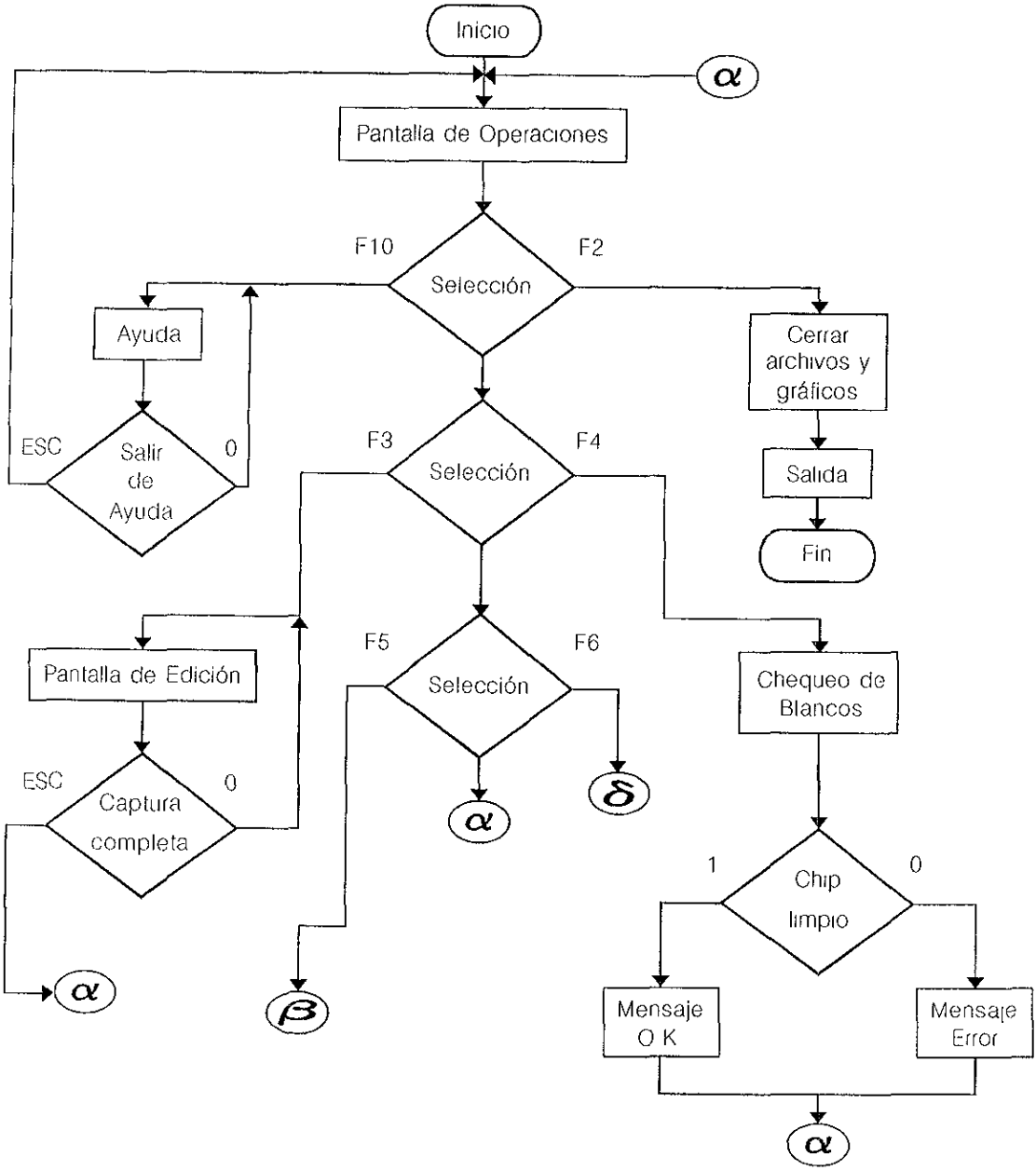


DIAGRAMA DE FLUJO DEL MODULO 3

[OPERACIONES]

Continuación..

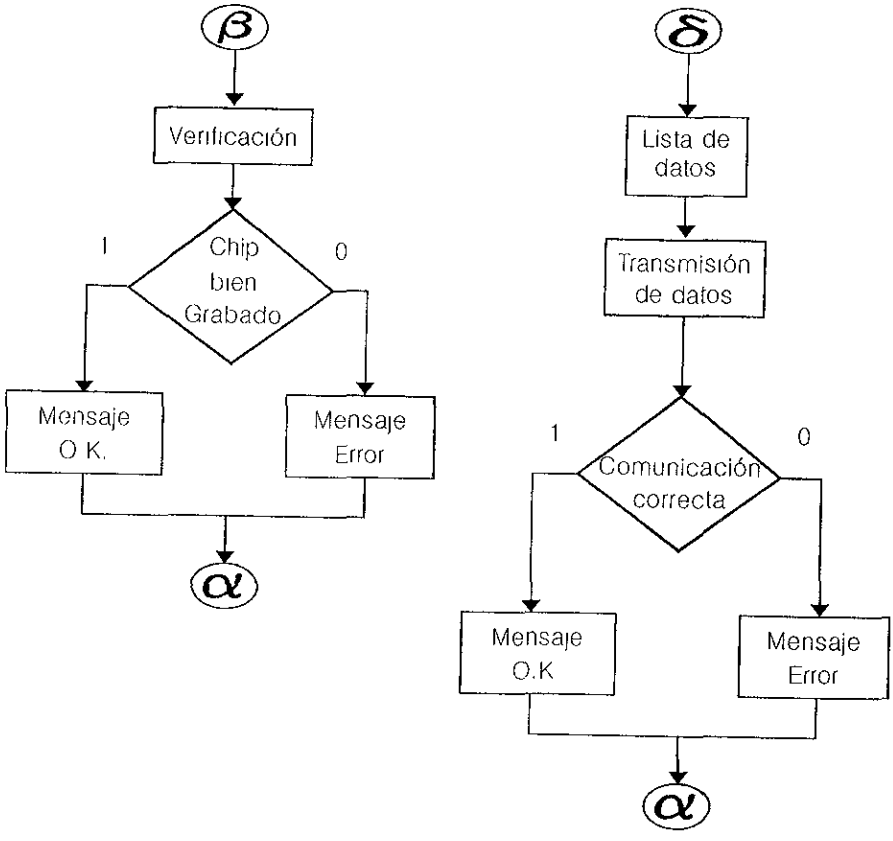


DIAGRAMA DE FLUJO DEL MODULO 3

[OPERACIONES]

PSEUDOCÓDIGO

'Carga de datos'

Para que el usuario capture los dàtos referentes a su programa se definió una estructura capaz de discriminar aquellos elementos que no corresponden a los formatos permitidos por el Sistema, también se le proporciona la ventaja de desplazarse a través de las líneas de edición por medio de las flechas de cursor.

Código de una de la hojas de edición y movimientos de cursor:

```
for (i = 0; i < 16; i++){
  printf("0", i#h);
    se presenta el encabezado de la matriz de edición
  otro_renglón:
  j=0;
  printf("00", j#h);
}

for (i = 0; i < 32; i++){
  printf("FF");
  if( i == 32 ){
    j = j+1;
    goto otro_renglón;
  }
}
```

en este bloque de código se establece la matriz de FF's que componen el área de captura, como puede observarse, se lleva un contador externo, éste se encarga de determinar el número de renglones que integran la pantalla, el ciclo interno permite generar el cuerpo de edición, una vez completa la pantalla se procede a ingresar los datos desde teclado.

En cuanto a los movimientos del cursor para dicha hoja de edición es:

```
tecla_cursor:
switch(tecla_cursora){
   case 'f_derecha':   if(x_max)
                         cont = cont + 1;
                         hoja = hoja + 1;
                        clrscr();
                         encabezado1();
                           para mostrar una pantalla de captura nueva
                         gotoxy(x_ini, y_ini);
                         goto tecla_cursor;
                           caso extremo
                       else
                         cont = cont + 1;
                         x = x + 1;
                           caso sin restricciones
                         gotoxy(x, y);
                         goto tecla_cursor;
   case 'f_izquierda': if(x_ini)
                         cont = cont - 480;
                         hoja = hoja - 1;
                         clrscr();
                         encabezado1();
                         for(x=x_ini; x<x_max; x++)
                           if(y_max)
                             goto tecla_cursor;
                           else
                             printf("%C%C",
                                    lista[cont].captura, lista[cont+1].captura);
                       else
                         cont = cont - 1;
                         x = x - 1;
                         gotoxy(x, y);
                         goto tecla_cursor;
                           caso sin restricción
```

```
case 'f_abajo'    :  cont = cont + 32;
                     hoja = hoja + 1;
                     clrscr();
                     encabezado1();
                       pantalla nueva, caso extremo
                     for(x=x_ini; x<x_max; x++)
                       gotoxy(x, y);
                       printf("%C%C",
                             lista[cont].captura,lista[cont+1].captura);
                       cont = cont + 32;
                       y = y + 1;
                       gotoxy(x, y);
                       goto tecla_cursor;
case 'f_arriba'   :  cont = cont - 32;
                     hoja = hoja - 1;
                     clrscr();
                     encabezado1();
                       pantalla nueva, caso extremo
                     for(x=x_ini; x<x_max; x++)
                       gotoxy(x, y);
                       printf("%C%C",
                             lista[cont].captura, lista[cont+1].captura);
                       cont = cont - 32;
                       y = y - 1;
                       gotoxy(x, y);
                       goto tecla_cursor;
```

'Verificación y Chequeo de Blancos'

Estos dos procedimientos actúan de manera similar es por ello que sólo se muestra uno:

limpia_puerto();

el procedimiento de limpiar el puerto elimina código inútil que se encuentra presente en el puerto serie

```
while(!dir_max){
  recibe();
  if(regs.h.al != 0){
    printf("Error en dispositivo");
    return;
  }
```

la pregunta que se realiza permite cortar el procedimiento de lectura y así continuar con otra operación.

'Programación'

Las funciones involucradas en este programa son más complejas puesto que se recurre a la interfaz RS-232 de tal forma que se necesita la habilitación y control de un manejador de comunicación:

limpia_puerto();
nuevamente el programa busca eliminar código inútil
pin1 = 10;
pin2 = 11;
pin3 = 12;
transmite(pin1);
los valores de pin1, pin2 y pin3 se transmiten para que sean definidos los pin's que involucran casos especiales (V_{CC} , V_{PP} , GND, etc.)
transmite(pin2);
transmite(pin3),
disposicion_pin's();
en este caso se indica y coloca la disposición de pin's para que sea transmitida
tranmite(pinn);

el código útil para la transmisión es el siguiente:

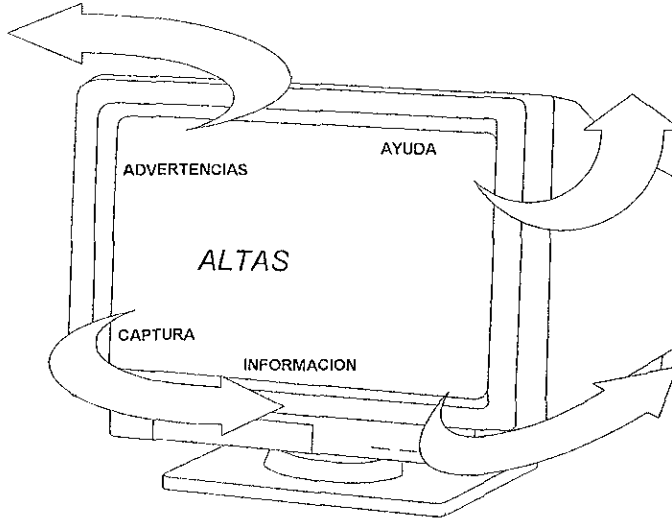
transmite(var_x){
var_x es un auxiliar o parámetro que se pasa de un procedimiento a este.

```
  regs.h.ah = 0X01;
```
donde 0X01 selecciona la operación que se desea realizar (1=transmitir, 2 = recibir)
```
  regs.h.al = var_x;
```
aquí se envía por puerto serie el dato
```
  regs.x.dx = PTO;
```
PTO contiene un cero que a su vez representa el puerto de comunicación serial COM1
```
  int86(0X14, &regs, &regs);
```
la interrupción 0X14 permite la comunicación vía serie
```
  if((regs.h.ah & 128) == 128){
```
esta pregunta permite indagar si la comunicación se completó y si ha sido correcta, en caso de tener una respuesta afirmativa se corta el ciclo de la comunicación
```
    printf("Error");
    bandera = 1;
    closegraph();
    exit(1);
  }
return;
}
```

Como puede observarse el flujo de los programas permite canalizar la información desde su captura hasta que es recibida en el grabador, el punto de unión entre la preparación de datos y la recepción de los mismos lo constituye el manejador del puerto, de tal forma que una posible falla de envío se ve reducida durante su procedimiento.

2.3.4. Altas

Este Módulo constituye la rama esencial del Sistema PR-2000, en él radica la ventaja de proporcionar las herramientas necesarias para poder actualizar los archivos de datos; para llevar a cabo dicha acción se recurrió a funciones que permitieran la entrada de información, la cual fue obtenida de las hojas de especificaciones del fabricante referidas al inicio de este apartado (voltajes, tiempos, capacidades y señales de los dispositivos lógicos programables), así como la presencia de ventanas que restringieran a la ejecución del programa, de tal forma que se protegiera tanto al usuario como al equipo (esquema 2.10). Su diagrama de flujo se encuentra en las páginas 42,43 y 44.



Esquema 2.10 Elementos que conforman la etapa de Altas.

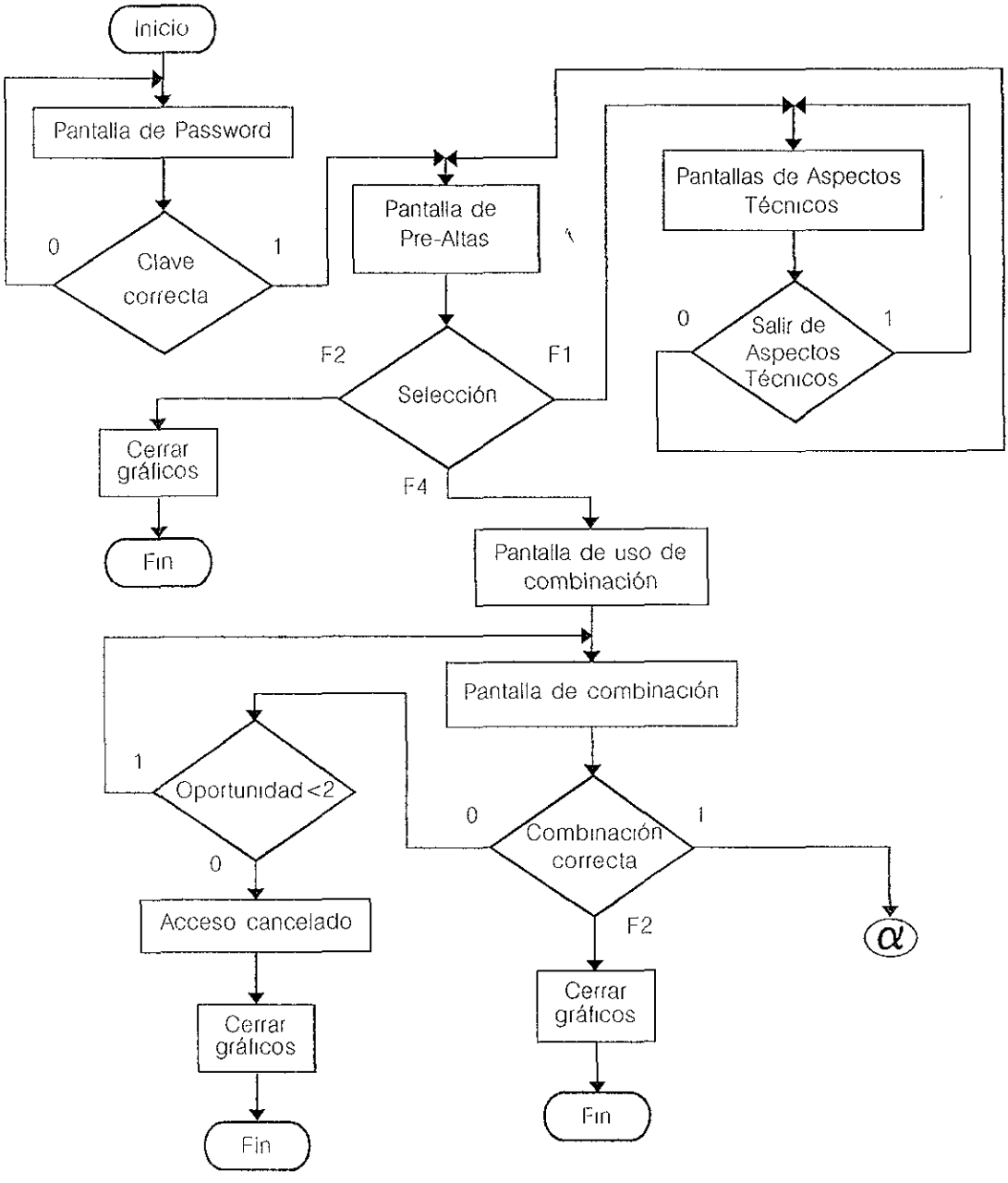


DIAGRAMA DE FLUJO MÓDULO 4

ALTAS

Continuación.

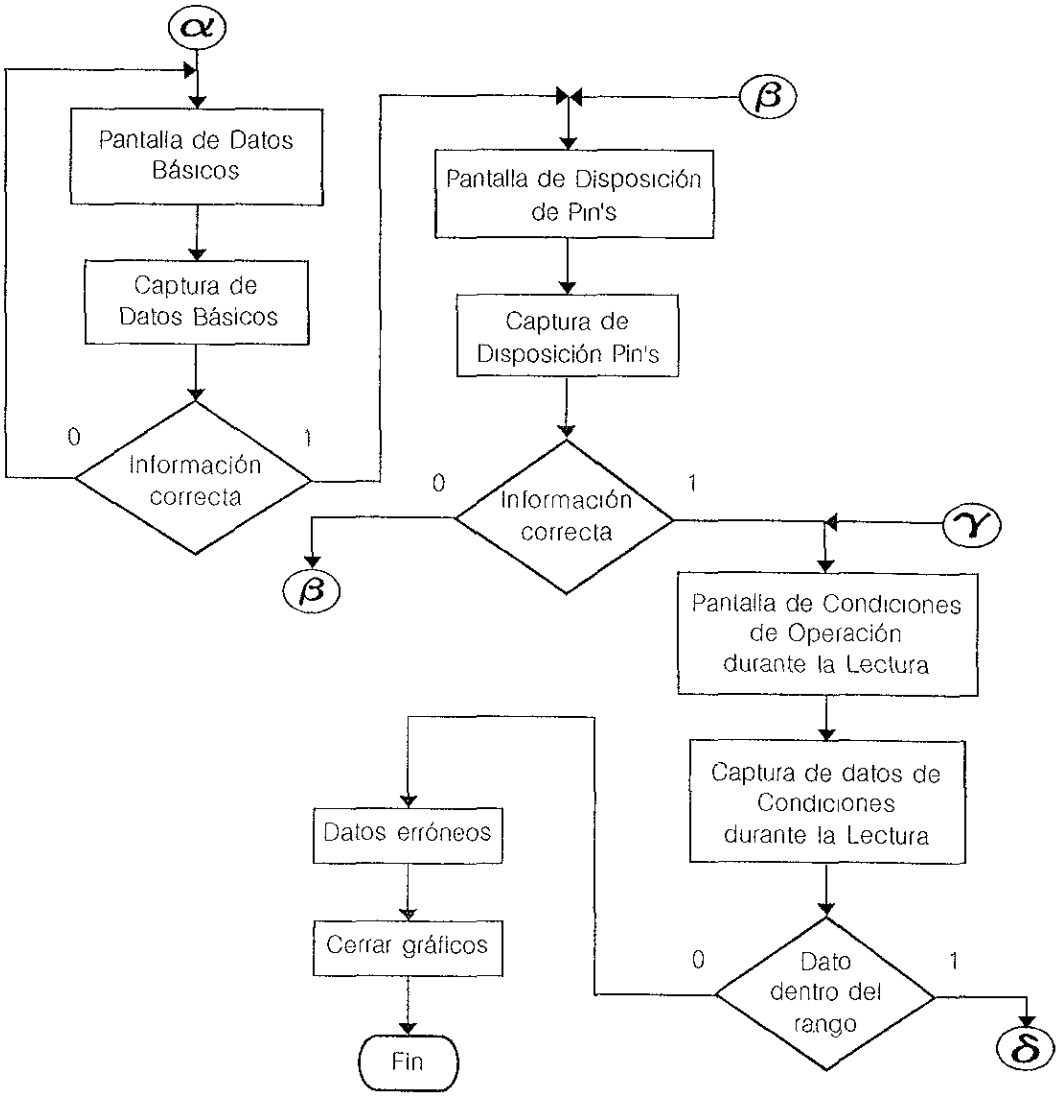


DIAGRAMA DE FLUJO DEL MÓDULO 4

ALTAS

Continuación..

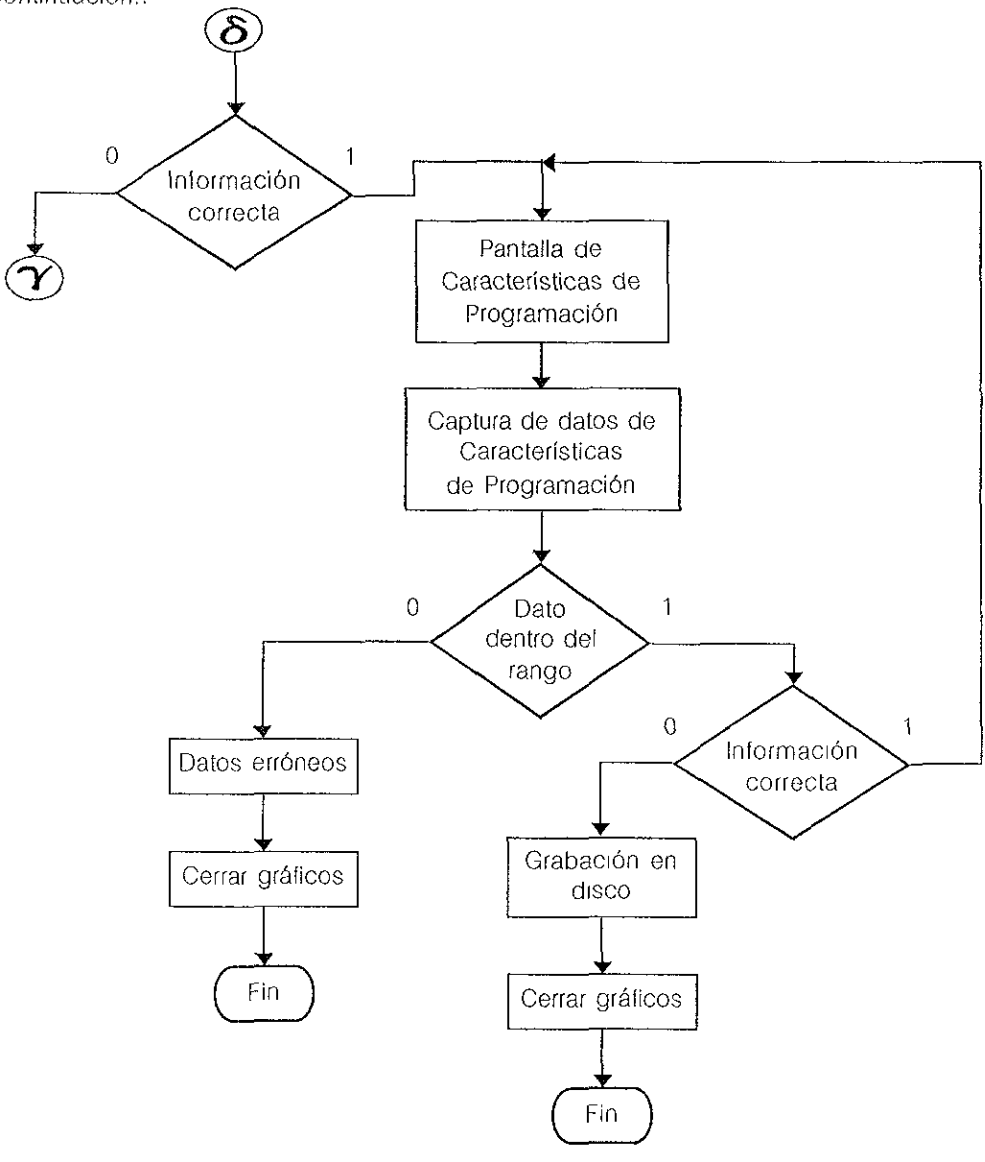


DIAGRAMA DE FLUJO DEL MÓDULO 4

ALTAS

PSEUDOCÓDIGO

Esta sección se ha resguardado a fin de evitar anomalías en la captura de información, tal y como se observa en el diagrama de flujo; para iniciar y limitar el uso de esta parte se muestra una pantalla que captura un password cuyo código es:

rectangle(x, y, x2, y2);
  rectángulo del marco
rectangle(x, y, x2, y2);
  rectángulo que cubre la clave
password(captura);
  estructura que compara si es el password es correcto
if(password==correcto)
  continua_altas;
else printf("¡ ERROR !");

(Nota : el lector deberá percatarse del uso continuo de coordenadas con identificadores [x, y] y la falta de incremento de ellos; ésto se debe a su carácter explicativo.)

Una vez que se ha introducido la clave correcta aparece una pantalla previa cuyo objetivo es el de hacer una pausa antes de entrar de lleno a la sección de captura con el fin de conocer más sobre los aspectos técnicos del Módulo 4, cuya duración queda restringida al tiempo que el usuario desee consultarla.

setfillstyle(1, 2);
bar(x, y, x2, y2);
setfillstyle(1, 3);
bar(x, y, x2, y2);
setfillstyle(1, 7);
bar(x, y, x2, y2);
  rectángulos que enmarcan la opción a elegir
settextstyle(3, 0, 2);
outtextxy(x, y, "F1 ASPECTOS TÉCNICOS");

```
outtextxy(x, y, "F2 SALIDA");
outtextxy(x, y, "F4 CONTINUA/ALTA");
repite:
getch(opcion);
switch(opcion){
   case 'F1  ':   pantallas_aspectos_tecnicos();
                    pantallas de información técnica
   case 'F2'  :   exit(1);
                    salida al Sistema Operativo
   case 'F4'  :   combina();
                    ingresa a la pantalla de indicaciones para habilitar la
                    siguiente llave

   default    :   goto repite;
}
```

Se presenta la opción de continuar siempre y cuando se reunan todos los requisitos. Una vez que se oprime la tecla F4 se ingresa a la segunda llave del programa, ésta se encuentra integrada por un texto explicativo que indica la forma en que puede ser superada dicha pantalla tal y como se muestra:

```
rectangle(x, y, x2, y2);
rectangle(x, y, x2, y2);
  límites del marco
outtextxy(x, y, " INSTRUCCIONES  DEL USO DE LA PANTALLA DE
                 COMBINACION");
   en tales párrafos se exponen las indicaciones pertinentes para el
   empleo de la pantalla de combinación
while(!ESC){
 continua presentando;
}else
  pantalla_de_combinación;
    se determina la duración de la pantalla.
```

El sistema de combinación involucra movimientos derechos y/o izquierdos por medio de las teclas de cursor; cada vez que el usuario selecciona un número deberá pulsar RETURN y continuar eligiendo. La clave consta de dos dígitos y dos oportunidades antes de que se finalice el programa.

En esta pantalla se aprecian mezclas de figuras geométricas con color, recuadros con información y un vector de números.

```
setfillstyle(1, 13);
bar(x, y, x2, y2);
  fondo de la pantalla
setfillstyle(1, 3);
bar(x, y, x2, y2);
  primer cuadro
setfillstyle(1, 9);
bar(x, y, x2, y2);
  segundo cuadro
setfillstyle(1, 6);
bar(x, y, x2, y2);
  tercer cuadro
setfillstyle(1, 2);
bar(x, y, x2, y2);
  cuarto cuadro
setfillstyle(1, 1);
bar(x, y, x2, y2);
outtextxy(x, y, "DESPLIEGA NUMEROS DEL 0-9")
  barra de dígitos
bar(x, y, x2, y2);
outtextxy("F2 SALIR");
  barra de salida al sistema operativo
otra_vez:
getch(opcion);
switch(opcion){
   case '→'        :        move derecha;
                              movimiento del cursor hacia la derecha
   case '←'        :        move izquierda;
```

movimiento del cursor hacia la izquierda

```
    case 'RETURN' :     selecciona número;
    case 'F2'     :     exit(1);
                          salida al sistema operativo
    default       :     goto otra_vez;
}
```

En caso de haber introducido correctamente los dígitos que conforman la clave aparece el número 2000 impreso en los cuadros y un rectángulo que dice ¡ ACCESO ! acompañado de un pitido.

La pantalla que sigue es la apertura a una serie de captura de datos intrínsecos de los dispositivos lógicos programables. En ella se consideran los parámetros de identificación tales como: nombre de la marca, tipo de dispositivo, número de fabricante, número de pin's, número de entradas y de salidas. También incluye teclas para corregir o continuar la secuencia del programa.

```
line(x, y, x1, y2);
line(x, y, x1, y2);
line(x, y, x1, y2);
line(x, y, x1, y2);
line(x, y, x1, y2);
line(x, y, x1, y2);
rectangle(x, y, x1, y2);
setfillstyle(1, 12);
bar(x, y, x1, y2);
outtextxy(x, y, "DATOS BASICOS");
outtextxy(x, y, "MARCA:");
  entrada del dato desde teclado
outtextxy(x, y, "DISPOSITIVO");
  captura del tipo de dispositivo
outtextxy(x, y, "NUMERO");
  ingreso del número de fabricante del dispositivo
outtextxy(x, y, "NUMERO DE PIN'S");
```

recepción del número de pins que componen al circuito
outtextxy(x, y, "NUMERO DE ENTRADAS");
ingreso del número de entradas
outtextxy(x, y, "NUMERO DE SALIDAS");
recepción del número de salidas que componen al circuito.

Una nota importante que hay que resaltar es la forma en que se capturan los datos, pues lo hace de una manera secuencial presentando la oportunidad de modificar la información.

Antes de continuar con la siguiente página de captura se le brinda al usuario la facilidad de verificar sus datos o de corregirlos en caso de que sea necesario para ello se requiere del siguiente switch.

```
otra_vez:
  getch(opcion);
  switch(opcion){
     case '→ ':        move derecha;
                         continua con la captura
     case '↑-' :       move arriba;
                         correcciones
     default   :       goto otra_vez;
  }
```

En la pantalla posterior se encuentra un dibujo de un circuito integrado, una rejilla en la cual se depositan las asignaciones de pin's correspondientes y un apartado destinado a la simbología que debe cumplirse. Dentro de esta pantalla se localiza en la parte central superior un cuadro de captura dedicado a recibir los datos del usuario, en él se pueden realizar correcciones y en el momento que se pulsa RETURN la información se deposita en la localidad de la reja señalada por un cursor circular de color verde. Al concluir dicha sección aparece un rectángulo orientado a enmendar alguna falla o proseguir.

```
de_nuevo:
  getch(opcion);
  switch(opcion){
      case '→'  :     move derecha;
                        pase a la siguiente página
      case '↑'  :     move arriba;
                        corregir
      default   :     de_nuevo;
  }
```

La continuación del programa se trata de la recopilación de información correspondiente a las condiciones de operación durante la lectura. La forma de captura es secuencial y limita la posibilidad de error al presentar tanto a un costado de la introducción de datos como en un recuadro las unidades convenientes.

```
settextstyle(2, 0, 0);
outtextxy(x, y, "CONDICIONES DE OPERACIÓN DURANTE LA
                 LECTURA");
setfillstyle(1, 10);
bar(x, y, x2, y2);
settextstyle(0, 0, 0);
texto();
  llamado a función para desglosar unidades
gotoxy(x, y);
printf("Vcc: ");
outtextxy(x, y, "Volts");
gets(variable);
if (4.49>variable)
  pantalla_aviso_parametros();
else if (5.51<variable)
      pantalla_aviso_parametros();
      dato fuera de rango
   else
gotoxy(x, y);
printf("Vpp: ");
```

```
outtextxy(x, y, "Volts");
gets(variable);
if (4.3>variable)
  pantalla_aviso_parametros();
else if (5.7<variable)
      pantalla_aviso_parametros();
      dato fuera de rango
   else
gotoxy(x, y);
printf("VIL: ");
outtextxy(x, y, "Volts");
gets(variable);
```

Nota importante estos parámetros manejan valores Mínimos, Típicos y Máximos.

```
gotoxy(x, y);
printf("VIH: ");
outtextxy(x, y , "Volts");
gets(variable);
gotoxy(x, y);
printf("VOL: ");
outtextxy(x, y, "Volts");
gets(variable);
gotoxy(x, y);
printf("VOH: ");
outtextxy(x, y, "Volts");
gets(variable);
regresa:
  getch(opcion);
  switch(opcion){
     case '→' :        move derecha;
                         pasa a la siguiente hoja
     case '↑' :        move arriba;
                         vuelve hoja
     default   :       regresa;
  }
```

Esta nueva hoja de captura conserva la estructura de la anterior, sólo que en esta ocasión se hace referencia a las características de programación:

```
settextstyle(2, 0, 0);
outtextxy(x, y, "CARACTERISTICAS DE  PROGRAMACION");
setfillstyle(2, 10);
bar(x, y, x2, y2);
texto1();
  llamado a función para desglosar unidades
gotoxy(x, y);
printf("Vcc: ");
outtextxy(x, y, "Volts");
gets(variable);
if (4.5>variable)
  pantalla_aviso_parametros();
else if (7.2<variable)
      pantalla_de_aviso_de_parametros();
      error en el rango
    else
gotoxy(x, y);
printf("Vpp: ");
outtextxy(x, y, "Volts");
gets(variable);
if (9.5>variable)
  pantalla_aviso_parametros();
else if (26<variable)
      pantalla_aviso_parametros();
      error en el rango
    else
gotoxy(x, y);
printf("VIL: ");
outtextxy(x, y, "Volts");
gets(variable);
```

Nota importante estos parámetros manejan valores Mínimos, Típicos y Máximos.

```
gotoxy(x, y);
printf("VIH: ");
outtextxy(x, y, "Volts");
gets(variable);
gotoxy(x, y);
printf("PW: ");
outtextxy(x, y, "seg");
regresa:
  getch(opcion);
  switch(opcion){
    case '→' :        move derecha;
                        salva todo
    case '↑' :        move arriba;
                        vuelve hoja
    default  :        regresa;
  }
```

2.4. NOTAS CONCLUYENTES

Las características del paquete Borland C++ aunado a su versatilidad gráfica, permite la introducción de estructuras de tipo interactivo, mismas que fueron introducidas en el programa, por ende se reducen los engorrosos ambientes de trabajo con opciones áridas y monótonas.

Otro aspecto relevante que se consideró es el de la programación modular, de esta forma si se detecta un mal funcionamiento éste puede ser rápidamente localizado y corregido; por otro lado pueden incluirse procesos que incrementen el potencial del Sistema, por ejemplo: anexar rutinas que permitan leer la información o programas desde disco flexible y grabarlos en el circuito integrado otra es el generar el código para el manejo del ratón, todo ello en el módulo que corresponda.

3
FASE II DEL DISEÑO

En este capítulo se expondrá el hardware, visto desde su diseño e implementación hasta las etapas operativa, directiva y de enlace.

3.1. ASPECTOS BASICOS

Conforme se desarrollaba el software del Sistema, paralelamente se iba estudiando la manera de como cubrir la parte suplementaria (el hardware). Para comenzar, se necesitaban estimar ciertas condiciones de carácter técnico y económico; citando a las primeras se hace énfasis en la compatibilidad entre ambas unidades remotas, dado que hay un programa que manipula la información que se interpreta y transforma en una secuencia de órdenes y cuyo objetivo a nivel circuito es accionar una salida física; la cual va a tener un valor establecido por los fabricantes y para ello se solicitará la presencia de dispositivos electrónicos versátiles que sean capaces de envolver tales rangos.

Por otro lado, si bien es indispensable satisfacer el concepto técnico también es imprescindible el aspecto económico, motivo por el cual se recurrió al uso de herramientas y circuiterías adquiridas en el ámbito estudiantil; de la selección de materiales destaca el microcontrolador MC68HC11F1, el cual es un miembro de la familia Motorola disponible en un encapsulado PLCC de 68 pin's, además cuenta con una arquitectura de varios periféricos y opciones que pueden ser utilizadas para un propósito especial dependiendo del modo de operación que se elija; por lo que puede cubrir y satisfacer las necesidades de comunicación, interacción, distribución, control, verificación y retroalimentación de los datos y particularidades inherentes al proceso de lectura y escritura de los circuitos lógicos programables; y para lograr cada tarea específica este dispositivo requerirá del auxilio de circuitos de apoyo. Como puede

apreciarse esta parte del Sistema PR-2000 estará configurada como un sistema digital.

Para dar un mejor enfoque del circuito primero daremos un vistazo a los elementos que intervienen en la ejecución de cualquier operación con un dispositivo lógico programable: en el caso de una lectura se requiere polarizar, habilitar a ciertos pin's y direccionar para obtener el dato.

Durante una programación además de polarizar se tiene que controlar el voltaje de programación, la duración del pulso, la dirección y el dato. Como puede observarse a partir de esta breve explicación salen a la vista los constituyentes del circuito: fuentes de alimentación tanto constantes como variables, un controlador de esta última, seleccionadores de terminales, direccionadores y receptores de datos.

Para poder conjuntarlos no debemos olvidar la valiosa colaboración del microcontrolador y de la vía de comunicación; es por ello que se comenzará explicando el funcionamiento de la unidad a partir de la fase de comunicación ya que ésta representa el primer contacto entre la computadora y el hardware; enseguida se abordará cada una de las secciones aportadoras que forman parte del grabador.

3.2. MODO DE TRANSFERENCIA

El canal de transferencia de datos debía contemplar lo siguiente: el tratamiento de códigos y las tareas físicas. En la primera se trabaja a partir de programas residentes cuyo principio básico es la definición de comandos semejantes que procesan datos similares. La segunda implica controlar algunas especificaciones planteadas por el fabricante; por lo que el diseño de la parte complementaria dependía grandemente de las condiciones vinculadas al sistema de cómputo, lo cual condujo a que la intercomunicación entre las terminales fuese de un canal de dos sentidos; por otro lado, debido a que la transmisión debe incluir una interfaz que reconozca correctamente los bits, se recurrió a la norma de información de tipo serial denominada RS-232-C, evitando así el incluir una tarjeta

especializada de comunicación de datos que le restaría compatibilidad además de un incremento en el costo.

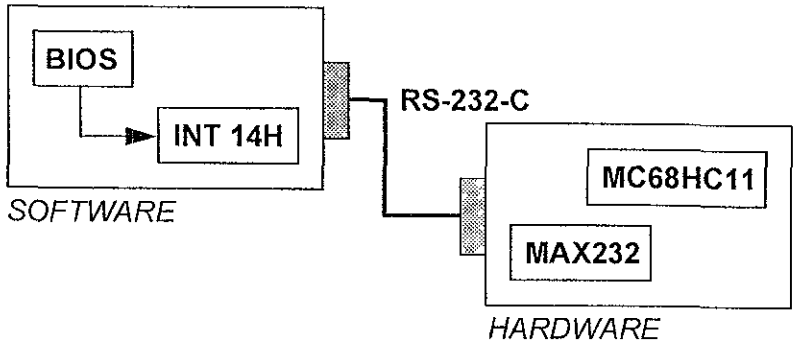
El siguiente paso era definir la manera de como accesar al puerto serie y por conveniencia se seleccionó a la BIOS, dado a que contiene *rutinas sencillas de manejo de puertos, que se activan mediante* interrupciones. De ellas se utilizó la interrupción 14H debido a que establece un método de comunicación entre la interfaz RS-232-C y el programa de aplicación por medio de cuatro servicios que son soportados por la PC:

- Inicialización del puerto serie
- Escribir un carácter
- Leer un carácter
- Obtener el estado del puerto serie.

Al abrir el puerto se ajustaron los parámetros de operación a valores concordantes al Sistema PR-2000. Una vez establecido su uso, se implemento un programa de comunicación. El Borland C++ utiliza la función *int86()* para ejecutar interrupciones, misma que simplificó el acceso a la transmisión de datos.

Hasta este punto la primer parte de la comunicación había sido cubierta; posteriormente se comenzó con la búsqueda de un dispositivo emisor-receptor capaz de identificar e interpretar los niveles de voltaje con los que trabajan las unidades remotas, además de presentar un acoplamiento sencillo. En el mercado hay componentes electrónicos que pueden realizar dichas tareas y dentro de ellos existe un chip el manejador/receptor (MAX232) cuyas características facilitaron el enlace y el ahorro de crear otras fuentes de alimentación.

Para concluir la tarea de transmisión se utilizó en el MC68HC11 el modo de operación especial bootstrap el cual permite que un programa de longitud variable sea cargado en RAM a través del puerto SCI (ver esquema 3.0).



Esquema 3.0 Componentes de la comunicación

3.2.1. Ciclo de transmisión

En el capítulo anterior se explicó a nivel programa como se realiza una transmisión, ahora en este apartado se visualizará como un conjunto acciones y elementos:

En la unidad almacenadora a los datos que van a ser transmitidos se les da un tratamiento en hexadecimal, luego son concentrados para después ser enviados en forma serial y ya estando en el exterior son recibidos en el MAX-232; este intermediario los convierte en señales que reconoce el MC68HC11.

Como desde un principio se especificó que el uso de este microcontrolador implicaría el empleo de dispositivos electrónicos como herramientas de apoyo, al recibir la señal enviada le da cierta interpretación para que a través de sus puertos las emita a diferentes áreas de trabajo. Para cerrar el ciclo de transmisión se monitorean aquellos puntos que indican el fin de una acción, en el caso de que se genere un aviso, éste es captado por el MC68HC11 y arrojado hacia el

MAX-232 para que le de la distinción correspondiente y pueda ser recibido en la computadora.

Para ilustrar la manera de como trabajan los dispositivos anteriormente mencionados, se hizo uso de "programas espejo", que no es más que, el generar un código en lenguaje ensamblador para el microcontrolador similar al código del lenguaje C++. El manejar así a las estructuras proporcionan ciertas ventajas como el obtener una rápida identificación y compilación de los datos.

El siguiente bloque de pseudocódigo ejemplifica el funcionamiento de dichos programas:

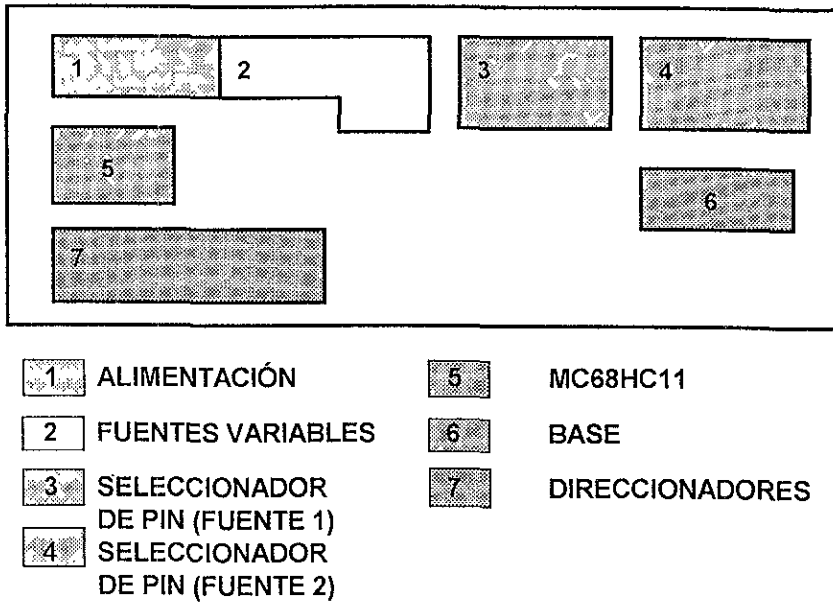
PC (Borland C++)		HC11 (Ensamblador)
		JSR RECIBE
if(volt<5){		LDAA DATOX
clave=1;		CMPA #$01
transmite(clave)		BNE VMAYOR
}		STAA PTOA
else		BRA PROCESO
clave=2;	VMAYOR	STAA PTOA
transmite(clave);		BRA PROCESO2

En el planteamiento del problema existe una disyuntiva con una solución, como puede observarse el programa en C++ es capaz de decidir entre dos posibles alternativas y una vez resuelta la operación transmite el parámetro (clave); por el otro lado, el programa planteado para disponer del microcontrolador realiza la comparación del identificador recibido para después ser emitido por el puerto A.

Para el cuerpo del Sistema PR-2000 se emplean este tipo de formatos; es decir, el manipular los bits en los puertos del MC68HC11 permite establecer el sentido y forma con que operan los demás dispositivos complementarios.

3.3. SECCIONES Y CÓDIGOS DE CONTROL

De acuerdo a lo visto con los modos de operación la primer zona por diseñar era la etapa de alimentación de todo el circuito, por lo que se buscó que los integrados respondieran con niveles de voltaje no mayor a 5 volts la siguiente parte la conformarían las fuentes variables que en combinación con los seleccionadores de pin's entablarían una opción de uno de los bosquejos planeados para cubrir ciertas estipulaciones de carácter técnico y económico planteadas por estos sectores. En cuanto al entorno del direccionamiento siguió un tratamiento similar al anterior. Un aspecto que se consideró durante la elección de los componentes de cada faceta era que el acoplamiento con el microcontrolador no resultase muy tedioso. A continuación se representará la distribución del circuito (ver esquema 3.1).



Esquema 3.1 Bloques del circuito.

3.3.1. Fuente de alimentación

Para hacer la fuente se empleó un modelo sencillo el cual utiliza los siguientes componentes y de base: un transformador conectado a la alimentación de corriente alterna (ca) que reduce el voltaje al nivel deseado, rectificando después con un circuito rectificador de onda completa, enseguida filtrando el voltaje con dos capacitores y por último regulando el voltaje cd con un regulador de voltaje de CI. En el caso particular de la fuente del circuito grabador se tuvo que modificar estratégicamente el prototipo de tal manera que satisfaciera otras condiciones que se presentarían durante el desarrollo de las etapas sucesivas; además de respetar las condiciones de operación de los elementos que serían incluídos.

Del material reutilizable y adecuado para este propósito se escogió un transformador de 127:30 V a 1 A, que de acuerdo con los valores de programación obtenidos los alcanzaba a cubrir, en cuanto a la alimentación del circuito, se realizó en forma escalonada; es decir, se utilizaron tres reguladores de voltaje fijo: uno de 24V, 12V y 5V, esta maniobra se basó en los siguientes puntos:

1) En las hojas de especificaciones recomiendan que el voltaje de entrada para el 7805 no debe pasar de los 25V y ser menor de 7V.
2) Del rango anterior se obtuvo un valor medio, y se optó por usar una referencia cercana (12 volts).
3) El 7824 sirvió como nodo de recepción de los 30 V y como fuente de alimentación de otras zonas.

El diagrama 2 muestra las conexiones de la Fuente de Alimentación.

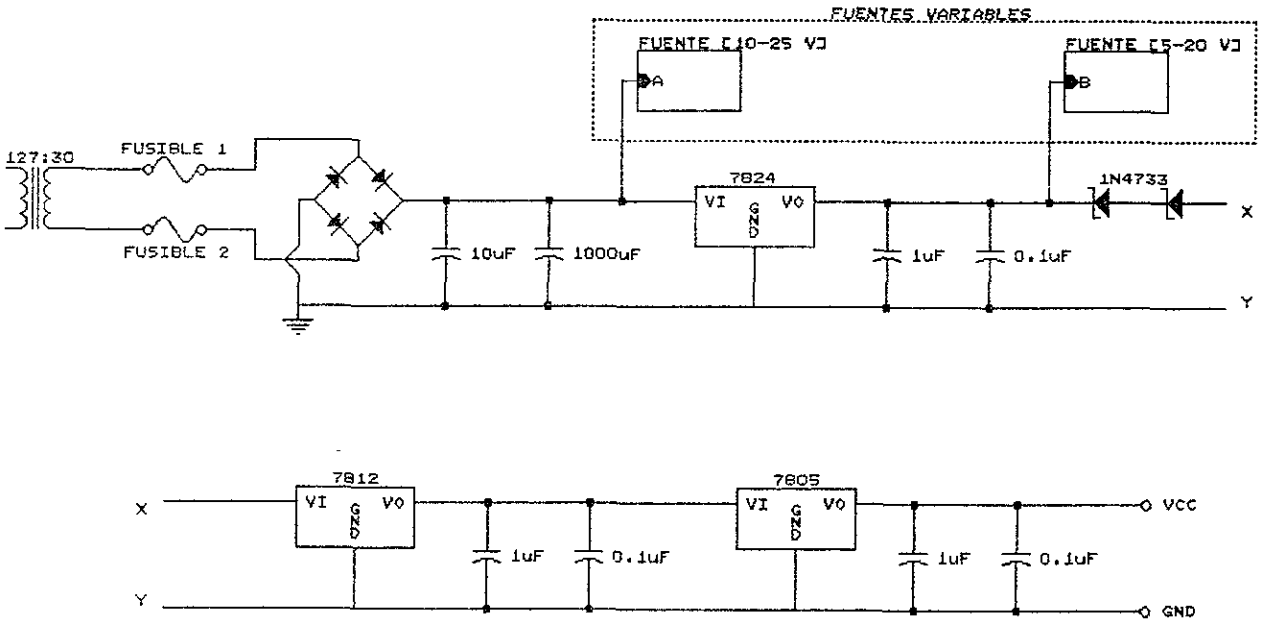


Diagrama 2 Fuente de Alimentación

Los nodos A y B son las entradas correspondientes a las fuentes variables.

3.3.2. Fuentes variables

Este módulo tiene como tarea activar un voltaje específico para poder realizar una programación; sin embargo, al revisar los pseudocódigos de programación de los dispositivos lógicos programables en algunas ocasiones requieren más de 5 volts de polarización (cuyo valor oscila entre 5.75 y 6.25 volts) y en el caso especial de un microcontrolador se vió que requería de 18 volts extras en un pin específico, por lo tanto al desarrollar la zona que se encargaría de entregar el voltaje del pulso de programación se decidió incluir otra, tratando así de cubrir esta clase de componentes. Ambas fuentes tienen como principal integrado al LM317 que es un regulador de voltaje positivo ajustable capaz de suministrar de 1.2 a 37 V, el cual envuelve perfectamente los rangos solicitados [(10-25 V) y de (5-20 V)]. De las aplicaciones típicas que ofrece se empleó la que muestra el diagrama 3.

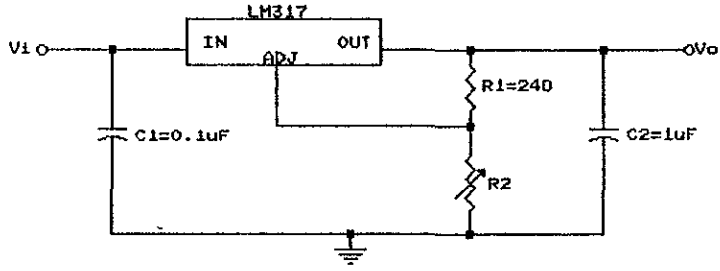


Diagrama 3. Configuración del LM317.

Como puede observarse se necesita de un voltaje de entrada V_I y dos resistencias y dos capacitores; para alimentar a la fuente de (10-25 V) se utilizaron los 30 V (nodo A del diagrama 2) que salen filtrados y la otra (5-20 V) toma el voltaje que se presenta a la salida del regulador 7824 (nodo B del diagrama 2). Una nota importante acerca de la determinación de las conexiones anteriores es el hecho de que al aplicarle la alimentación al regulador variable es aconsejable que supere al menos cuatro volts al valor máximo requerido. Por otro lado, el valor de la resistencia R2 permitió establecer a la salida el voltaje deseado, por medio de la siguiente ecuación:

$$Vo = Vref\left(1+\frac{R2}{R1}\right)$$

donde

$$Vref = 1.25\ V,\ R1 = 240\Omega,\ V01 = 10 - 25\ V \text{ y } V02 = 5 - 20\ V$$

al despejar a R2 se tiene que:

$$R2 = \left[\left(\frac{Vo}{Vref}\right) - 1\right]R$$

Al sustituir los valores para la fuente de (10 - 25 V) se obtuvo lo siguiente:

V01(V)	R2(Ω)
10	1680
11	1872
12	2064
13	2256
14	2448
15	2640
16	2832
17	3024
18	3216

19	3408
20	3600
21	3792
22	3984
23	4176
24	4368
25	4560

y para la fuente de (5 - 20 V)

V02 (V)	R2(Ω)
5	720
6	912
12	2064
14	2448
18	3216
20	3600

Después de haber obtenido estos valores la siguiente acción era determinar la manera de como trabajar con cada uno de ellos. Para comenzar, se necesitaba identificar de alguna manera las fuentes, es por ello que como la fuente de 10 a 25 volts manejaría 16 niveles de voltaje y la fuente de 5 a 20 volts 6; se decidió codificar a la primera con 4 bits y a la segunda con 3 quedando de la siguiente forma:

Niveles	Código	Volts
1	0000	10
2	0001	11
3	0010	12
4	0011	13
5	0100	14
6	0101	15

7	0110	16
8	0111	17
9	1000	18
10	1001	19
11	1010	20
12	1011	21
13	1100	22
14	1101	23
15	1110	24
16	1111	25

y para la otra

Niveles	Código	Volts
1	000	5
2	001	6
3	010	12
4	011	14
5	100	18
6	101	20

Esta estrategia condujo a buscar dispositivos que a partir de un código binario de entrada (de 4 y 3 bits) se activara sólo una salida, de esta manera se seleccionaría el voltaje requerido como su correspondiente resistencia; para lograr tales propósitos la primer parte la integrarían decodificadores 4:16 (74LS154) y 3:8 (74LS138), pero como presentan las salidas de lógica negada se solicitó la presencia de inversores que a su vez complementarían la segunda etapa, la cual estaría diseñada con transistores BC547 funcionando como circuito de switcheo para así poder controlar el estado de conexión-desconexión de la resistencia. Los diagramas 4 y 5 muestran las conexiones de tales arreglos.

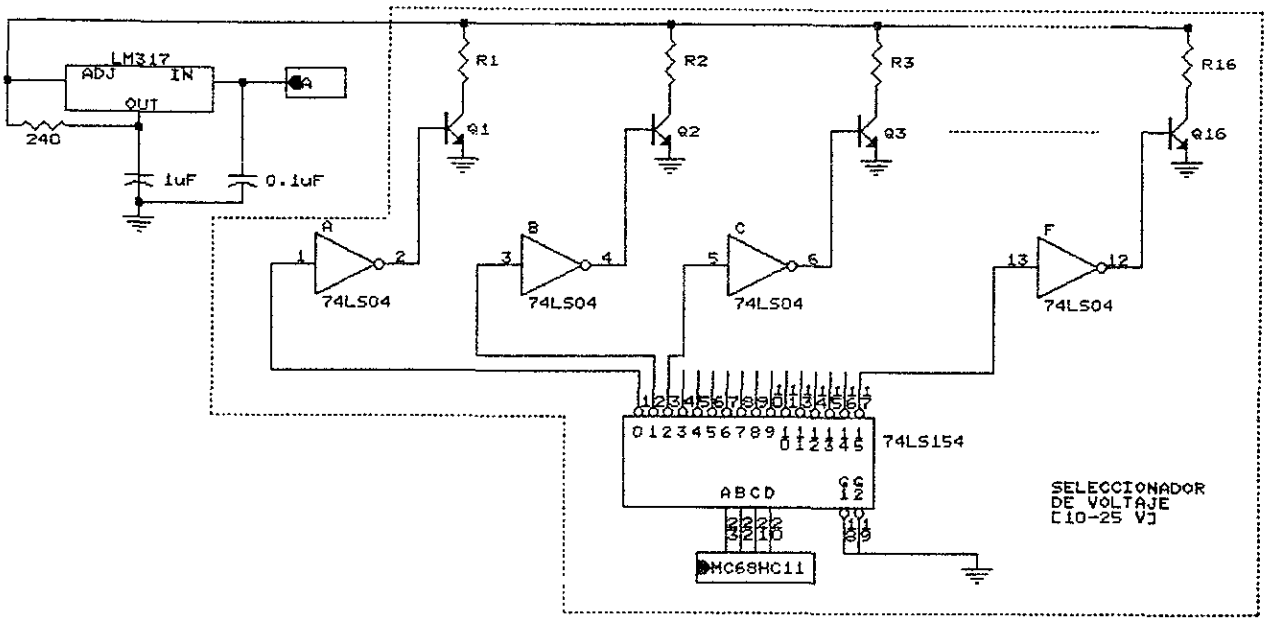


Diagrama 4 Fuente variable de 10-25 volts.

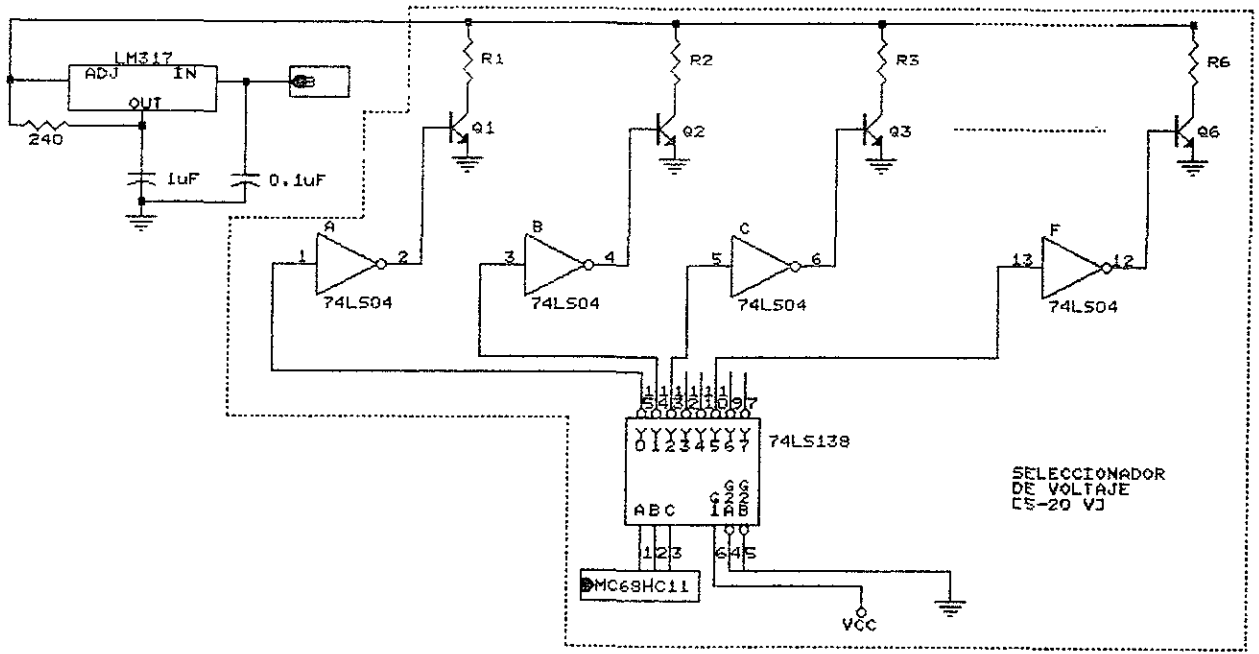


Diagrama 5. Fuente variable de 5-20 volts.

Para comandar estos circuitos el MC68HC11 envía por un puerto el código que relaciona al voltaje de programación y/o el de polarización especial, esta señal es captada por los decodificadores 74ILS154 y 74LS138 y como la respuesta es de lógica negada es invertida por los 74LS04 para así poder controlar a una serie de transistores dispuestos a operar como switches; esta parte se diseñó de esta forma para que trabajara solamente uno de los arreglos compuesto por un transistor y la resistencia (R2) requerida, y una vez obtenido el voltaje se le hace un tratamiento similar a la salida del integrado LM317; es decir, como actúa de llave para la sucesiva zona también es vigilado por interruptor cuyo estado de encendido-apagado es atendido por mando del microcontrolador; para comprender como opera esta fase de las fuentes se tomará la configuración correspondiente al circuito switch con paso de voltaje. Ver diagrama 6.

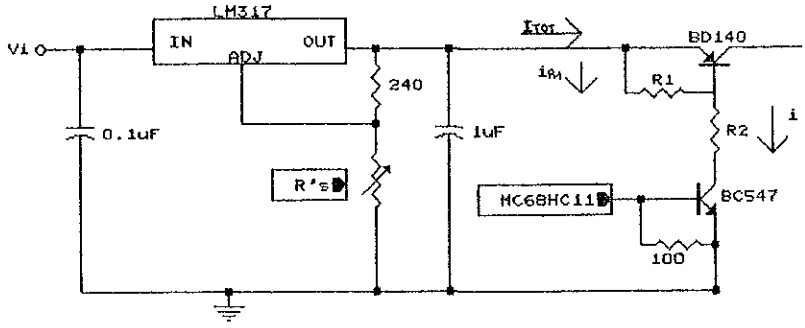


Diagrama 6. Switch del LM317.

Como se requiere trabajar con voltajes que oscilan de 5-25 volts se decidió utilizar un transistor de potencia media (BD140) y un transistor común para el acoplamiento con el microcontrolador. Del diagrama 6 se analizará el circuito donde aparece la resistencia R2; para comenzar se tiene que:

$$= \frac{I_{TOT}}{\beta} + i_{R1}$$

si $_{R1} = \frac{V_{EB\,SAT}}{R1}$, *entonces la ecuación queda como*

$$i = \frac{I_{TOT}}{\beta_{BD140}} + \frac{V_{EB\,SAT}}{R1} \quad .....(1)$$

donde I_{TOT}=1A, β_{BD140}=190, V_{EBSAT}=0.7 V Y R_1=1KΩ, por lo tanto

$$i = \frac{1}{190} + \frac{0.7}{1000} = 5.963 \times 10^{-3} A$$

Una vez obtenida la corriente se procederá a encontrar el valor de la resistencia R2 para el caso de la fuente de 10-25 volts y para la fuente de 5-20 volts.

Fuente de 10-25 V:

$$V_{F25} + V_{EB} + iR2 - V_{CE} = 0$$

sustituyendo valores y despejando

$$25 - 0.7 - (5.936 \times 10^{-3})R2 - 0.2 = 0$$
$$R2 = 4.04 \times 10^{3} \Omega$$

Fuente de 5-20 V

$$V_{F20} + V_{EB} + iR2 - V_{CE} = 0$$

sustituyendo valores y despejando

$$20 - 0.7 - (5.936 \times 10^{-3})R2 - 0.2 = 0$$
$$R2 = 3.2 \times 10^{3}\Omega$$

Con el fin de que el grabador no trabajara con valores críticos, se decidió colocar resistencias con niveles abajo del obtenido cuidando que no se afectaran los límites de operación del transistor. Antes de proseguir es importante recapitular el comportamiento de las fuentes variables debido a que las siguientes áreas son el paso hacia la base de pin's del equipo; para iniciar se hace referencia a que el encendido-apagado de una o de ambas va a depender del tipo de chip a programar o a leer, estas situaciones van a ser supervisados por el microcontrolador a través de los dos switches definidos anteriormente y de los arreglos con los decodificadores. Al dejar pasar el voltaje solicitado y para poder depositarlo se tenía que implementar un circuito que permitiera seleccionar el pin específico.

3.3.3. Seleccionador del pin

Esta sección tiene como objetivo reflejar la instrucción que viene del MC68HC11 de localizar el pin al cual se le va a suministrar el voltaje (es importante hacer una breve pausa para recordar que el Sistema acepta como máximo 40 pin's y 25 volts; por lo tanto, al elegir los componentes de este sector se tenían que tener muy presente dichas características) y para ello se estudiaron los elementos que tienen como labor el cambiar de condición cuando se presenta una referencia; los posibles aspirantes arrojaron lo siguiente: el primero estaba constituído por una serie de contadores de secuencia binaria y como cada encapsulado maneja 4 bits se vió que se necesitaban 10 de ellos por cada fuente dando un total de 20 circuitos integrados, situación que no redituaba en el aspecto económico y de espacio; por otro lado, el segundo contemplaba a los decodificadores que como es bien sabido son de lógica negada y por ende

para este propósito se tenían que auxiliar de un conjunto de inversores resultando un total de 20 chip's (4 decodificadores 4:16, 2 de 3:8 y 14 inversores 7), es por ello que la cuestión económica y técnica se vería un tanto afectada; el tercero fue planeado con registros de corrimiento de entrada en serie y salida en paralelo de 8 bits y de acuerdo a lo establecido por el Sistema, sólo se solicitaban 5 chip's por cada fuente, además de que se podía arreglar en cascada evitando así el utilizar más de dos puertos; como se puede observar éstos reportaron buenas ventajas sobre los anteriores. Para llevar a cabo la selección sólo basta recibir la orden indicando el número de pin al cual se le va aplicar el voltaje, tal información llega al microcontrolador para ser codificada y enviada por un puerto hacia los pin's de entrada serie, el reloj y el clear de los registros de corrimiento cuya conexión encadenada facilitó el control de los mismos. La ubicación del pin queda asignada por un 1 y para hacer el enlace con el sector anterior (un voltaje de 10-25 y/o de 5-20 volts) faltaba planear un arreglo que permitiera hacer ello y además de que dejara trabajar con el resto de los pin's. Esta situación tenía que ser resuelta por medio de transistores como interruptores dado a que algún chip que pudiese ser acoplado no soportaba los máximos de voltaje manejados. El circuito switch con paso de voltaje es similar al anterior sólo que los componentes cambian, a continuación se muestra:

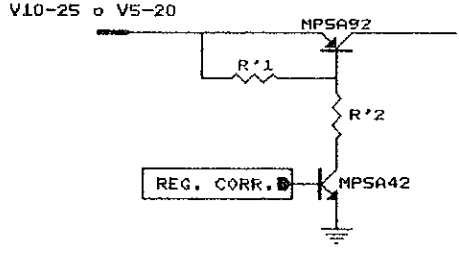


Diagrama 7 Switch del seleccionador del pin.

Dada la naturaleza de los índices de operación referente al voltaje y corriente que soportan los transistores MPSA92 y MPSA42 además de su bajo costo, se concluyó utilizarlos. Para realizar los cálculos correspondientes a este arreglo daremos un vistazo al diagrama 8.

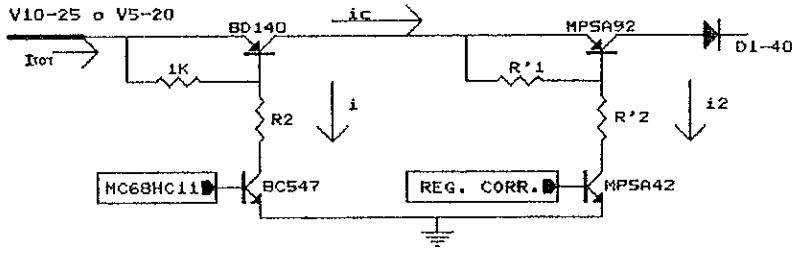


Diagrama 8. Seguimiento de corrientes

De la ecuación (1) se obtuvo que

$$i = 5.963 \times 10^{-3} A$$

por lo que

$$i_c = I_{TOT} - i$$

sustituyendo valores se tiene que

$$i_c = 1 - (5.963 \times 10^{-3}) = 0.994A$$

entonces

$$2 = \frac{i_C}{\beta_{MPSA92}} + \frac{VEB_{SAT}}{R'1}$$

donde β_{MPSA92}=100, V_{EBSAT}=O.7 V y R'1=1KΩ, por lo tanto

$$i2 = \frac{0.994}{100} + \frac{0.7}{1000} = 10.64 \times 10^{-3}$$

finalmente para la fuente de 10-25 V tenemos que

$$V_{F25} + V_{EB} + iR'2 - V_{CE} = 0$$

$$25 - 0.7 - (10.64 \times 10^{-3})R'2 - 0.2 = 0$$
$$R'2 = 2.265 \times 10^{3}\Omega$$

y para la fuente de 5-20 V

$$V_{F20} + V_{EB} + iR'2 - V_{CE} = 0$$

$$20 - 0.7 - (10.64 \times 10^{-3})R'2 - 0.2 = O$$
$$R'2 = 1.795 \times 10^{3}\Omega$$

A nivel programa el orden cronológico que siguen los dos sectores compuestos por una fuente variable y un seleccionador de pin cada uno es:

1. De la computadora baja el número de pin en específico.
2. El cual es interpretado por el microcontrolador para saber cual de las dos fuentes trabajará.
3. De la interpretación anterior se desglosan tres señales básicas, una es dirigida al seleccionador de pin, otra a los decodificadores y por último al switch del LM317, tomando en consideración este orden.

Los diagramas 9 y 10 esquematizan a los seleccionadores de pin.

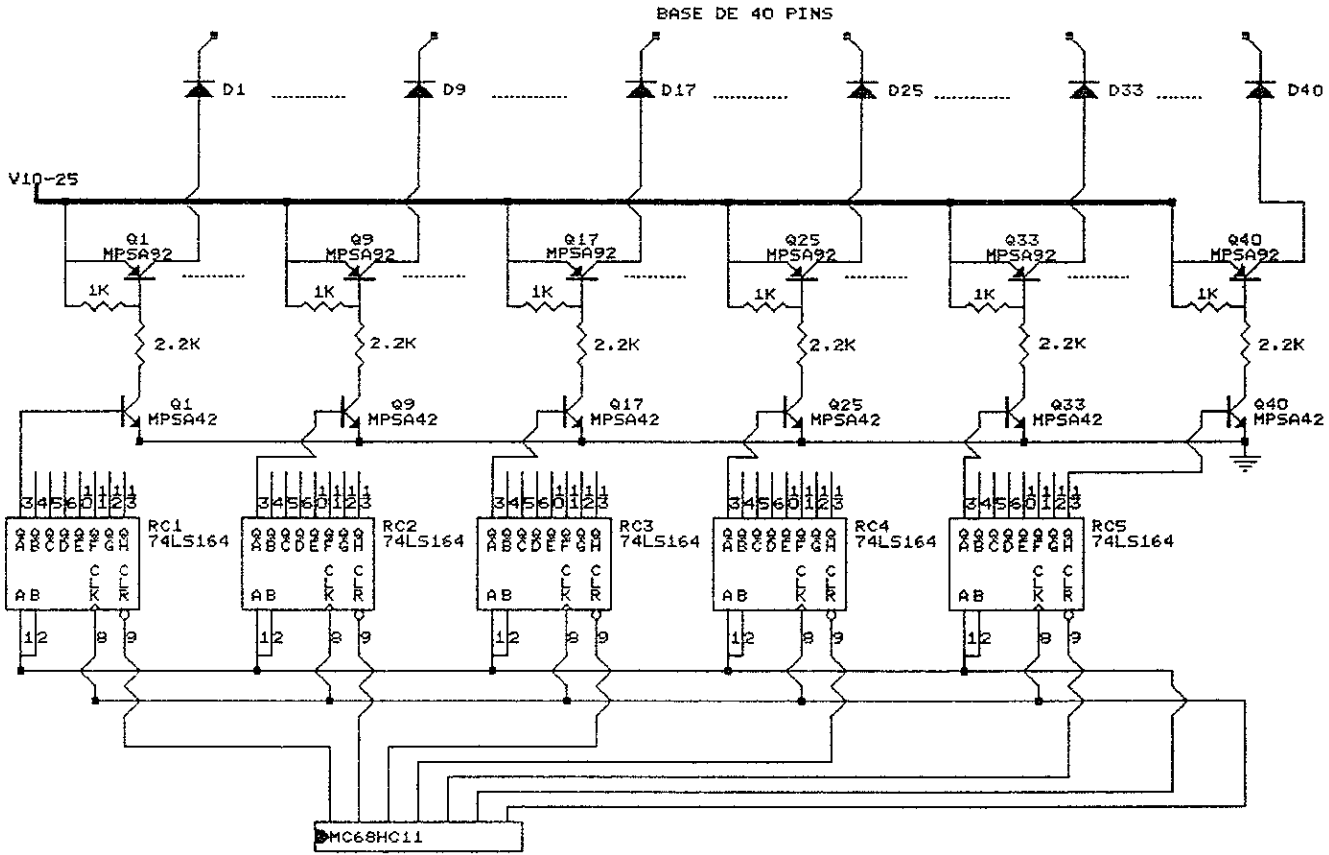


Diagrama 9. Seleccionador del pin [10-25 Volts].

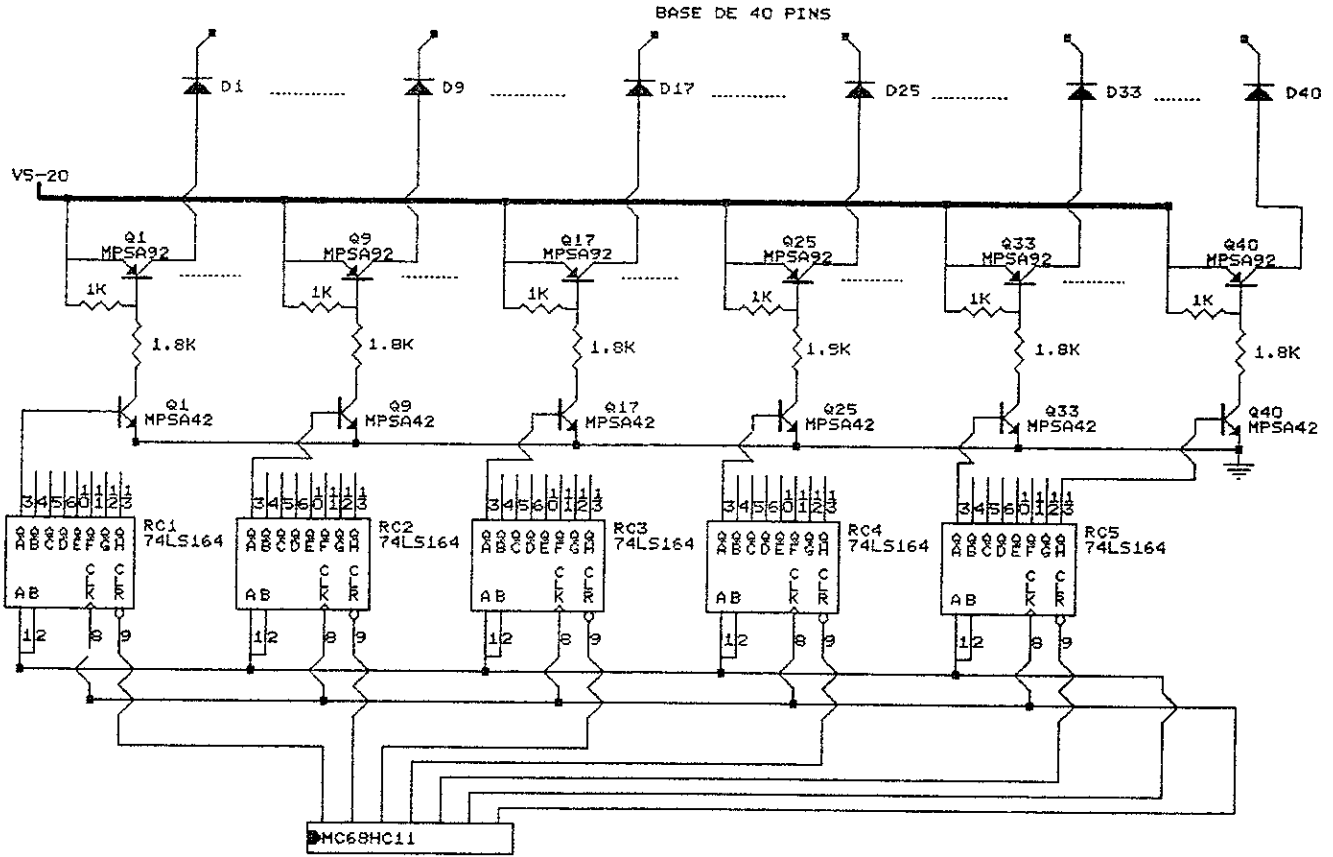


Diagrama 10. Seleccionador del pin [5-20 Volts]

3.3.4. Direccionamiento

En uno de los apartados anteriores se expuso como se cubren los voltajes de polarización requeridos en una operación de lectura/escritura en un dispositivo lógico programable y más adelante se mostró de que manera se pueden seleccionar a los pin's que les llegarán tales voltajes; esta última parte se tomó como referencia para desarrollar a uno de los prototipos que cubrirían la sección del direccionamiento. No obstante, antes de proseguir con una explicación más concreta acerca de los diseños que se pensaron se hará un recuento de las acciones que se deben efectuar en cualquier operación con los chip's programables; ambas formas recurren a la alimentación y al direccionamiento para después obtener o introducir un dato y como puede observarse estos dos procesos son de diferente naturaleza por lo cual se decidió separarlos y tratar como primer instancia a la programación. En ella es importante reconocer la ubicación de los pin's que involucran a las entradas de direcciones y a los datos, por lo que a partir de esta característica se buscó que los elementos que intervendrían en esta zona pudiesen llevar a cabo un recorrido a los pin's del dispositivo que se va a programar y a su vez localizar a las direcciones de entrada (A_0 - A_n) e inicializarlas para comenzar a introducir los datos.

Una vez definida la tarea a realizar se concebió plasmarla por medio de contadores binarios en décadas que ejecutaran una secuencia ascendente de 0 a 39 (cubriendo así el número máximo de pin's) y bancos de flip flops D (mismos que se encargarían de mantener la información hasta que fuese requerida); sólo que esta agrupación de elementos presentaba una serie de inconvenientes de tipo económico y de espacio debido a que cada contador maneja 4 bits y los flip flops D vienen dos por encapsulado, por lo tanto se iban a necesitar 30 dispositivos y si consideramos a los chip's que auxiliaran en la colocación del dato el número de éstos se incrementará. Por otro lado, se hizo un bosquejo con contadores y decodificadores esta combinación resultaba un tanto atractiva, ya que en el mercado existen bancos de dec's; sin embargo, la salida de éstos se vería afectada debido a que son de lógica negada y

antes de llegar las señales a su destino, tendrían que pasar por unos inversores; situación que aumentaba la cantidad de chip's.

Después de analizar cuidadosamente los diseños anteriores, surge por segunda vez la presencia de los registros de corrimiento dada la versatilidad que presentaron.

El lector recordará que para dar una idea más clara de los dos tipos de operación (lectura/escritura), se estableció una separación por lo que *quizás los planteamientos antepuestos no satisfacían del todo el objetivo.* Esta situación marcó una definitiva aprobación para el uso de los registros de corrimiento cuyos aspectos técnicos y económicos redituaban más ventajas. Pues bien, dado a que cada encapsulado ofrece 8 bits de salida sólo se solicitarían 5 registros de corrimiento ofreciendo así un tren de salida total de 40 bits reduciendo notablemente el número de chip's y lo más extraordinario que éstos ofrecían era la manipulación de dos señales distintas (direcciones de entrada y datos) simplificando el control de las mismas. A partir de este momento es conveniente ser más explícitos en la postura anterior y para ello se describirá como se conjuntó el trabajo del soporte lógico y físico:

Una vez que se tiene identificado el tipo de componentre a programar se sigue la rutina de localización, activación y depósito del dato de interés

⇒Paso 1. Se localizan las direcciones de entrada (A's), para ello es necesario saber el número total de pin's del circuito; al determinarlas se realiza un barrido en el vector de pin's del programa (realizado en Borland C++) para establecer quien es A_0, A_1, A_2,..., An; ahora bien, para cada A se asocia un valor clave en C++ y se crea un vector paralelo de claves, en él se asocian valores de 1 y 0 por lo que de esta forma se tiene un tren binario; cabe destacar que las siguientes etapas (ubicación de los pin's de entrada de datos O's y habilitadores) son semejantes.

⇒Paso 2. El tren binario se envía al microcontrolador MC68HC11F1 y se alista para ser puesto en un pin de uno de sus puertos; para que sea

adecuada la etapa de direccionamiento se requiere efectuar un ciclo que incremente los valores de las direcciones de entrada (A's) y coordinarse con las entradas de datos (O's) para que sean depositados correctamente.

⇒Paso 3. El tren de pulsos binarios sirve como entrada al pin A de uno de los registros de corrimiento y como se encuentran en cascada son capaces de recorrer hasta 40 localidades; una vez fija la disposición de pin's se depositan los datos y dependiendo de la duración del pulso de programación se tardará en aplicar el reset para dejar listo el direccionador.

El diagrama 11 de la página 79 muestra las conexiones que se efectuaron en la zona de direccionamiento para la grabación.

Otro aspecto relevante es que tanto este circuito como algunas partes del proceso expuesto con anterioridad servirían para activar a las direcciones de entrada en el modo de operación de lectura, reiterando una vez más la incorporación de los registros de corrimiento en el grabador y para concluir la modalidad de lectura sólo bastaba con definir al receptor de los datos contenidos en el circuito lógico programable, para ello es necesario contar con el arreglo adecuado de dispositivos que en conjunto permitan mantener latentes las señales que emite el dispositivo programado, agruparlas y enviarlas al microcontrolador quien a su vez cerrará el ciclo de comunicación con la computadora.

La etapa encargada de recolectar la información que proviene de las salidas del circuito integrado previamente grabado es de tal naturaleza que recurre a la secuencia de direccionamiento que se emplea para grabar, con la salvedad de que en este caso las salidas denominadas O's, tienen prioridad en su operación y manejo de la información que contienen pues de ellas serán obtenidos los datos que componen al conjunto de instrucciones o programa que en un momento determinado pueden ser consultados por el usuario.

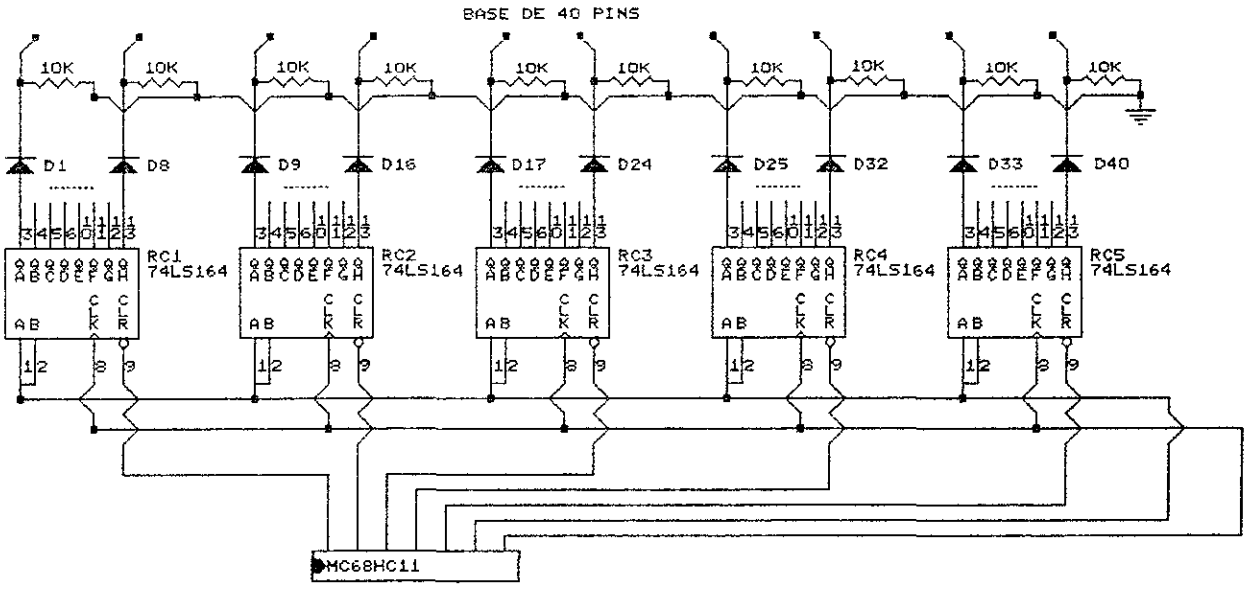


Diagrama 11 Direccionamiento.

Los pasos que se realizan para llevar cabo una lectura son los siguientes:

⇒Paso 1. Los registros de corrimiento se inicializan con un valor cero para evitar que existan datos ajenos al proceso, posteriormente se recurre a la disposición de terminales para definir la ubicación de las entradas A's y salidas O's.

⇒Paso 2. Una vez que se ha definido la posición de las direcciones de entrada, se realiza un ciclo que incremente el valor de éstas para que se lean los valores contenidos en las localidades programadas, a la par de esta iteración se envía un tren de 1's en las terminales correspondientes a las salidas O's, lo anterior es para que se identifiquen los datos.

⇒Paso 3. Con la información obtenida durante el proceso de lectura y mediante un registro de corrimiento se forman grupos de 8 bits que serán transmitidos al microcontrolador.

⇒Paso 4. El MC68HC11 reune los datos y los envía a la computadora, de esta forma se cierra el ciclo de comunicación y el usuario puede consultar la información contenida en su dispositivo lógico programable.

3.4 NOTAS CONCLUYENTES

Este capítulo ha reunido las características del hardware que integra al Sistema: se revisó la importancia de contar con una interfaz de comunicación confiable, de igual forma se trató el diseño de las fuentes de voltaje y como se regula su operación en el circuito; por último, se revisó la forma de direccionar el circuito para que sean aplicados los voltajes y datos deseados, ahora se procede a evaluar el funcionamiento del conjunto para determinar el grado de solución que se dio al problema originalmente planteado.

4
PRUEBAS

En este proyecto la organización modular resultó fundamental, por lo que al fusionar cada sección de hardware con su respectivo software y efectuarles las pruebas correspondientes, la táctica de dividir las zonas arrojaron una serie de datos de carácter técnico y económico que resultaron en una mejor propuesta de diseño. A continuación se describirán los procedimientos y resultados de las pruebas.

ENLACE COMPUTADORA-GRABADOR

Para llevar a cabo la comunicación entre la computadora y el grabador se realizó un alambrado en project board del MC68HC11 en modo de operación especial bootstrap (ver diagrama 12), el cual fue alimentado por una fuente de 5 volts ensamblada durante la estadía en la Facultad.

Cabe aclarar que de acuerdo a cada operación que realiza el grabador se fue predeterminando la función de cada puerto del microcontrolador; es por ello, que el puerto A fue resguardado para la transmisión de datos, una vez definido dicho puerto se le adicionaron unos led's a la salida para verificar por medio de un pequeño programa si existía comunicación entre ambas unidades. La principal función de la computadora era enviar un nivel alto para encender un led y la del microcontrolador era devolver un nivel alto para así cerrar el ciclo de comunicación inicial. En esta fase no se presentaron dificultades mayores debido a lo sencillo del circuito.

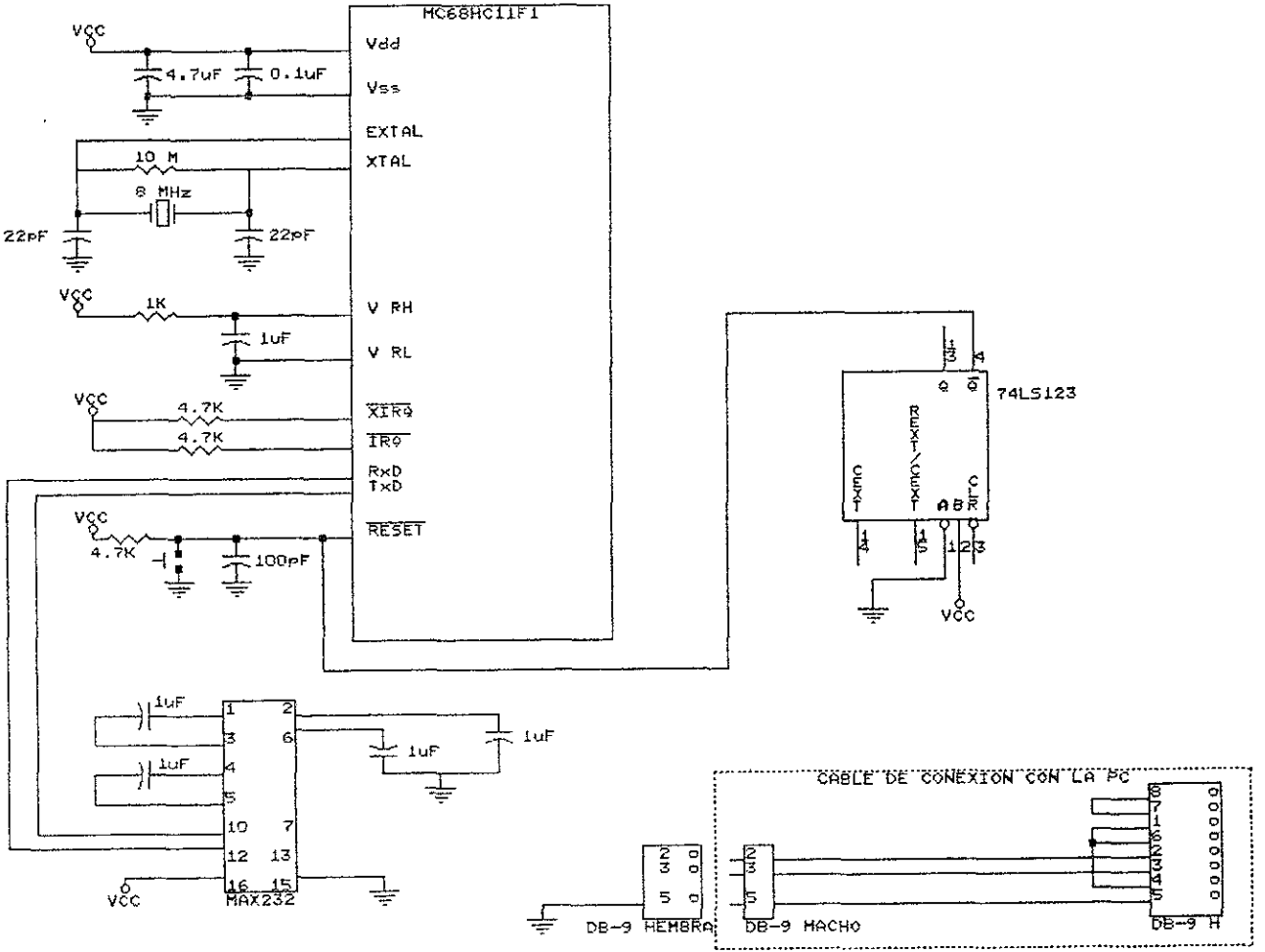


Diagrama 12 Modo Bootstrap del MC68HC11.

Para completar esta prueba se verificó el caso contrario; es decir, aquella situación en la cual el circuito del microcontrolador no tuviese alimentación; previamente se habían diseñado tanto el código como las pantallas de advertencia y de diálogo que intervendrían en esta acción las cuales permitirían solventar esta situación y continuar con la corrida normal del programa, al efectuar esta fase se detectó que efectivamente las pantallas se activaban y se mantenían mientras no se corrigiera el estado del circuito.

Como el número de elementos que intervienen en la configuración del modo bootstrap del MC68HC11 es mínimo se decidió montarlo en una tableta fenólica perforada; posteriormente se llevaron a cabo las mismas pruebas resultando satisfactorias.

FUENTES DE ALIMENTACIÓN Y VARIABLES

La fuente de alimentación se ensambló en una project board y al efectuarle las pruebas se observó que había un calentamiento en los reguladores de voltaje fijo (7812 y 7805), por lo que se colocaron diodos zener a la entrada del 7812 y además se acoplaron unas barras de aluminio como disipadores y un pequeño ventilador; una vez corregido este detalle se procedió a adjuntar en la misma tableta a los reguladores de voltaje positivo ajustable (LM317).

Como puede apreciarse éstos elementos se relacionan con las fuentes variables; sin embargo, antes de proseguir se le recordará al lector que se utilizó una aplicación y una fórmula determinada (ver páginas 62 y 63) para manejar diversos niveles de voltaje, es por ello que al buscar el valor más aproximado de la resistencia variable R2 (por medio de un arreglo de 2 resistencias) se obtuvieron los siguientes datos prácticos:

Para la fuente de 10-25 volts

Valores teóricos

V_{01}(V)	R_2(Ω)
10	1680
11	1872
12	2064
13	2256
14	2448
15	2640
16	2832
17	3024
18	3216
19	3408
20	3600
21	3792
22	3984
23	4176
24	4368
25	4560

Valores prácticos

V_{01}(V)	R_2(Ω)
10.17	1680
11.22	1882
12.13	2070
13.23	2256
14.07	2440
15.35	2670
16.18	2850
16.98	3033
18.03	3220
19.01	3410
20.10	3690
21.00	3820
21.60	4080
22.70	4230
23.70	4450
24.60	4690

y para la fuente de 5-20 volts

Valores teóricos

V_{02}(V)	R_2(Ω)
5	720
6	912
12	2064
14	2448
18	3216
20	3600

Valores prácticos

V_{02}(V)	R_2(Ω)
5.24	740
6.18	920
12.14	2040
14.04	2420
18.17	3220
20.10	3600

En algunos casos se modificaron los valores de las resistencias para aproximarse más al voltaje deseado. Cabe señalar que cada dato fue obtenido de manera particular; es decir, que en la tableta de las fuentes se fue alambrando uno por uno cada arreglo de resistencias (R2).

Al establecer el switch (arreglo con BC547) de conexión-desconexión de cada R2 y su respectivo seleccionador (decodificadores 3:8 y 4:16) se procedió a ensamblar en otra project board la etapa del seleccionador de voltaje (ver diagramas 4 y 5). Para llevar a cabo las primeras pruebas se colocaron led's en lugar de resistencias y la decodificación se realizó en forma manual; es decir, por medio de alambres colocados en las entradas de los decodificadores (3:8-ABC- y 4:16-ABCD-) se les daban un cierto código, 000 a 111 para el 74LS138 y de 0000 a 1111 para el 74LS154 por medio de Vcc y GND; al ver que no presentaban problemas y respondían a las expectativas se decidió colocar las resistencias (R2).

Para conjuntar las tabletas de las fuentes variables con la del seleccionador del voltaje se probó el LM317 junto con el decodificador 4:16 de la siguiente forma: se le retiró la alimentación al LM317, mediante el manejo de alambres se le asignó un código binario al decodificador, le fue devuelto el suministro al regulador de voltaje positivo ajustable y previamente colocada una resistencia de carga a la salida del mismo se le tomaba la lectura al voltaje. Un test similar se aplicó al otro LM317 con su respectivo decodificador 3:8

Para culminar el trabajo de las fuentes variables se implementaron los switches de los integrados LM317 (ver diagrama 6), de esta forma se obtenía el control del paso del voltaje requerido y a manera de comprobar su funcionamiento se realizó una prueba manual al LM317 con su decodificador 4:16 que consistía en: desactivar el switch del LM317, dar el código al 74LS154 y posteriormente activar al switch del regulador para tomar la lectura del voltaje previsto.

Al terminar las pruebas en project board de las fuentes, se determinó ensamblar en una tableta fenólica perforada tanto a los componentes de la fuente de alimentación como a los elementos

(incluyendo a los switches) de las fuentes variables; sin embargo, la parte correspondiente a la selección de voltajes (conjunto de R2 y decodificadores 3:8 {74LS138} y 4:16 {74LS154}) se quedó en project board y para llevar a cabo la conexión entre ambas tabletas se utilizó alambre tipo telefónico.

Por medio de un programa que respondiera a las necesidades de control de circuito ya implementado se realizó una simulación de la operación de los switches, reguladores y decodificadores; dicho programa contiene las siguientes instrucciones:

¿CUAL ES LA FUENTE A UTILIZAR?

La selección de la fuente se lleva a cabo por medio de la captura de una letra:

(**A**) para el rango de 10-25 V
(**B**) para el rango de 5-20 V

¿CUAL VOLTAJE SE UTILIZARA?

Una vez seleccionada la fuente se captura el valor numérico del nivel unitario del voltaje.
En este momento se indica al MC68HC11 que prepare un nivel de voltaje alto para aquella fuente de voltaje variable que trabajará, mientras que se determina un nivel bajo a la fuente que se quedará fuera de operación, posterior a esta acción se transmite el código necesario para que el decodificador realice su labor.

¿ CUAL ES LA FUENTE QUE TRABAJARA?

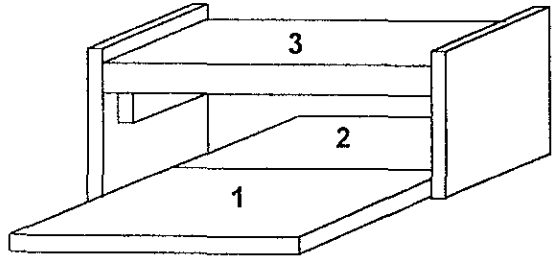
Retomando el programa anterior se recordará que se mantenían latentes las señales que habilitarían y/o deshabilitarían a las fuentes

variables; sin embargo, es en esta etapa cuando el microcontrolador las emite hacia los transistores y efectúan su labor; por ello aunque parezca repetitivo, se realiza un segundo cuestionamiento acerca de la fuente de operación.

Por último se realizaron las pruebas necesarias no presentando ningún problema hasta este momento.

SELECCIONADOR DEL PIN

Al controlar la regulación y paso de voltaje de 10 a 25 volts y de 5 a 20 volts se prosiguió a instalar la zona que daría cause a dicho voltaje; para llevar a cabo ésta tarea se recurrió a un sistema de switcheo constituído por unos transistores manejados por circuitos digitales (ver diagramas 9 y 10). Para poder ejecutar las pruebas primero se ensambló el seleccionador del pin de 10 a 25 volts, para ello se buscó la forma de acomodar a todos los componentes de tal manera que el circuito del grabador no se extendiera mucho, fue entonces que se optó utilizar tres project board: dos empalmadas y la tercera montada en unas bases de acrílico (ver esquema 4.0).



Esquema 4.0 Disposición de tabletas project board.

En la tableta 1 se alambraron 20 switches constituídos por 20 MPSA92, MPSA42, 20 resistencias de 1KΩ y 20 de 2.2KΩ, en la tableta 2 se ensamblaron los 5 registros de corrimiento (74LS164) y en la tableta 3 los restantes 20 switches.

Como el resto del diseño lo integran 40 diodos cuyas salidas van directo a la base del grabador, se dispuso a usar una tableta fenólica perforada donde se colocaron los diodos conectados a un jumper y por último con fines de prueba, en una tableta project board se ensamblaron led's con un jumper.

A manera de dar inicio a las pruebas se utilizaron las tabletas 1 y 2 cuya interconexión se hizo con alambre tipo telefónico y para unir la salida de los switches con los diodos se utilizó alambre de la misma clase, sólo que ahora se entrelazó y se cortó a la medida de cada salida; sin embargo, el empleo de esta clase de conector traería 2 consecuencias: por un lado permitió introducir cada terminal en la project board de manera sencilla y ordenada debido a que se utilizaron varios colores; por otro lado el haber soldado la otra terminal a los diodos y dada la rigidez del material conductor se presentaron casos de terminales sueltas.

Al lograr la unión de las tabletas participantes en la prueba y para visualizar el comportamiento de este sector, se realizó un pequeño programa que consistía en involucrar preguntas relacionadas con las posiciones de los pin's, la secuencia es:

CUAL ES LA MARCA DEL DISPOSITIVO? *CUAL ES EL TIPO DEL DISPOSITIVO?* *CUAL ES EL NUMERO DEL DISPOSITIVO?*	Estas preguntas se relacionan con la posibilidad de identificar la existencia del dispositivo en sus archivos, además no se envía ninguna señal al MC68HC11.
CUAL ES EL NUMERO DE PIN'S?	En este momento se hace la búsqueda del pin especial para poder depositar el voltaje requerido.

En este caso se envía un código dependiendo del pin, ésta clave alista a los registros de corrimiento para su oportuna participación e incluye el número de veces que debe recorrerse un pulso para indicar el pin elegido. Al recibir el microcontrolador esta cadena de datos le da el tratamiento correspondiente para colocar en el puerto A un valor en hexadecimal, tales órdenes llegan directamente a los registros de corrimiento finalizando con el encendido de un led sin que se reportaran mayores problemas.

El siguiente test consistió en reemplazar los led's por resistencias de carga mismas que permitirían medir el voltaje que proporcionaba el circuito, de esta forma se estaba acercando la mayor prueba: el depositar el voltaje deseado en el pin indicado, lo anterior sin olvidar que todo este proceso involucra la unión de los programas que sirvieron para los items anteriores.

Una vez conjuntado el software necesario y los implementos electrónicos adecuados se procedió a realizar la prueba de designación de pin y selección de voltaje de la fuente variable, observándose una ligera caída en los niveles de voltaje (0.08 hasta 0.12 V), misma que no fue considerada como relevante; sin embargo el lector no debe olvidar que hasta el momento se ha trabajado con un arreglo de 20 switches, al agregar los 20 restantes y realizar las conexiones necesarias se presentaron otros inconvenientes.

La conexión de la tableta 3 con la tableta 2 (unión salidas de registros de corrimiento con transistores) se llevo a cabo con alambre tipo wire wrap enrollado a pin's, esta decisión se tomó considerando que es más manejable pero su tratamiento es más delicado, si consideramos este último argumento resultaba poco factible emplearlo en la unión transistor-diodo, pues al momento de ser soldado su fraccionamiento sería frecuente y si a esto le aunamos la posición vertical que tendría el alambre, no convenía hacer uso de él; dada esta situación se empleo alambre tipo telefónico.

Cabe destacar que una vez conjuntado el arreglo de los 40 switches y al efectuarle la prueba de selección de voltaje y designación de pin, los

valores de los voltajes se vieron variados tal y como se aprecia en la tabla siguiente:

Fuente de 10-25 V

V_{01}(V)
10.09
11.15
12.08
13.19
14.18
15.16
16.01
17.24
18.03
19.17
20.00
21.60
22.30
23.10
24.30
25.40

La implementación del circuito asi como las pruebas que permitieron conocer los valores de voltaje antes mencionados, se aplicaron a la otra fuente variable (5-20 V) y a continuación se muestran los resultados obtenidos:

Fuente de 5-20 V

V_{02}(V)
5.12
6.06
12.30
14.08
18.06
20.10

NOTAS CONCLUYENTES

Esta sección resumió las acciones que se realizaron para verificar el correcto funcionamiento de las distintas etapas del circuito grabador, a través de este recuento de pruebas se observó como se estableció la comunicación entre dispositivos, también se pudo apreciar la manera en que funcionan las fuentes de voltaje realizando mediciones tanto a aquella que sirve de alimentación general como a las que permiten seleccionar *una gama de valores escalonados de voltaje por unidad*; *la última prueba* consistió en verificar la puesta en operación del circuito que se encarga de seleccionar al pin que utilizará el voltaje de programación, de todo ello se obtuvieron resultados satisfactorios sin embargo; existieron problemas en la implementación debido a las tabletas de ensamble utilizadas, es por ello que deberá proponerse la creación de un circuito impreso que permita montar los componentes adecuadamente para elevar la respuesta operacional del grabador.

5
CONCLUSIONES

El objetivo principal de este trabajo fue el diseñar un programador inteligente de dispositivos lógicos programables, para ello se buscó un caso práctico que permitiera conocer los principios de operación del sistema y en la Facultad se contaba con un programador denominado UP600; sin embargo se observó que el actuar sobre la estructura rígida de este equipo era poco rentable pues no existían las condiciones técnicas necesarias para ello, ante tal situación se concibió el diseño del Sistema PR-2000, el cual sería capaz de soportar modificaciones en sus archivos de datos a fin de programar nuevos dispositivos.

Al analizar al UP600 y su complemento de software se detectó que éste último no permitía alteraciones en sus características internas además de que no era factible incluir nuevos componentes, este resultado originó que se definieran tipos de datos acordes al funcionamiento del Sistema y se contemplara también el empleo de operaciones de captura más dinámicas, de tal forma que conforme se requiriera programar a un nuevo elemento, éste mantuviera sus características en los registros del programa PR-2000. El haber recurrido al lenguaje de programación C++ fue uno de los aspectos que mayor impulso dio al software, lo anterior debido a que C++ proporciona un manejo de las estructuras de almacenamiento de datos adecuado a los requerimientos del usuario, permite realizar la comunicación con el microcontrolador MC68HC11 en forma precisa y es útil para efectuar programación modular.

Todo lo anterior resume la etapa de software del PR-2000 y es ésta la que más aportes innovadores presenta pues, es capaz de ingresar nuevos elementos programables a su conjunto de datos. Cabe destacar que el programa del Sistema fue desarrollado en forma modular, lo que permitió probar y corregir las etapas con algún problema, más adelante se refuerza este planteamiento de desarrollo por fases pues es uno de los factores que permitió concluir el trabajo con menores fallas.

Por otro lado; se tiene al soporte eléctrico del programador, el cual se trató en forma modular para facilitar sus pruebas y avanzar a la par del desarrollo del programa; es en esta etapa cuando se presentan diferencias entre los datos calculados teóricamente y los que se obtuvieron de manera práctica, pero ahora se comentarán las acciones más relevantes que se realizaron con el Sistema y sus variaciones con respecto a los valores esperados.

Como se mencionó con anterioridad el diseño de esta fase fue en forma modular, lo que permite determinar las fallas de manera rápida y en su caso; ejercer una acción correctiva, el primer conjunto de pruebas se aplicó al enlace entre la computadora y el circuito grabador, para ello fue necesario definir un programa que habilitara los puertos de comunicación de la computadora, alistara los datos para su envío, transmitiera la información al microcontrolador y dejara listo el puerto para recibir el dato que éste último le enviara; a la par se desarrolló el circuito que trabajaría con el programa, para ello se utilizó un circuito integrado capaz de interpretar las señales del puerto de la computadora y convertirlo a voltajes que pudieran ser tratados por el microcontrolador; éste circuito (MAX232) dadas sus cualidades, respondió a la acción encomendada sin mayores complicaciones y la aplicación permitió entablar la comunicación entre ambas unidades sin problemas tal y como lo pudo notar el lector en el capítulo de pruebas.

Al situarnos en este nivel de implementación, se puede concluir que el haber generado las etapas del grabador por separado para posteriormente evaluarlas, fue una de las claves para alcanzar el objetivo inicialmente planteado; sin embargo más adelante se comentarán algunas fases que son susceptibles de mejorarse y que indudablemente elevarán la calidad del diseño aquí mostrado.

La implementación, las pruebas y correcciones que se efectuaron a las fuentes variables y a la fuente de alimentación resultaron ser muy interesantes, el por qué de esta aseveración radica en que los datos técnicos que se utilizaron en el diseño de las mismas sufrieron variaciones al momento de ser implementado el circuito experimental, tal es el caso de los reguladores de voltaje positivo ajustable (LM317) los cuales, al ser

implementados en el circuito de prueba, no contaron con los valores teóricos de las resistencias que integran su configuración; lo anterior tiene como justificación el que las resistencias comerciales no tienen un valor exacto sin embargo; esta aproximación no perjudicó el funcionamiento de la implementación, el control de encendido-apagado de las fuentes variables se estableció por medio de un transistor de potencia media, mismo que dependiendo de la señal que le enviara el microcontrolador, activaría o desactivaría la unidad, esta etapa se ensambló en una tableta fenólica para evitar problemas de operación, la conclusión es que debe mantenerse un margen de aplicación en los valores de los componentes pues no siempre se pueden implementar los circuitos con las características obtenidas mediante el cálculo. Por lo que respecta a la fuente de alimentación cabe señalar que no se detectaron problemas en su funcionamiento.

La fase correspondiente al circuito seleccionador del pin no tuvo tantos problemas como los que se presentaron en las fuentes variables de voltaje en cuanto a valores teóricos contra las características comerciales de los circuitos que se emplearon en su implementación experimental sin embargo; el haber montado los componentes en tabletas project board trajo como consecuencia dificultades en su conexión con las etapas restantes del circuito aquí, a manera de conclusión se puede decir que una aplicación en tabletas experimentales resta confianza en el funcionamiento de la aplicación sin embargo; el desarrollar un circuito impreso para esta fase del circuito resultaría con un costo elevado pues el lector debe considerar que se trata de dos áreas de características semejantes.

En general se puede concluir que el objetivo originalmente planteado se cubrió en cuanto a la aportación y diseño de una solución integral que permita el grabar dispositivos lógicos programables por medio de un ambiente amable de cómputo y una interfaz electrónica versátil; como puede apreciarse, el bosquejo realizado es susceptible de ser mejorado, sobretodo si se considera que en la actualidad se cuenta con herramientas de programación orientadas a la creación de ambientes de trabajo en Windows, mismos que amplian la gama de actividades que el

usuario puede realizar en ellos, además de contar con la comodidad de utilizar un señalizador gráfico (mouse).

Por otro lado, si bien se presentaron problemas en la implementación del circuito, también se obtuvieron resultados aceptables en las pruebas realizadas sin embargo, es recomendable trabajar con circuitos impresos para evitar fallas en las aplicaciones pero, como se mencionó con anterioridad, éste paso involucra un costo económico alto. En lo referente a los dispositivos electrónicos empleados éstos también se encuentran en posibilidades de cambiarse por componentes más actualizados, tal es el caso de los circuitos integrados TTL que podrían ser reemplazados por otros de tecnología CMOS que presentan una respuesta más rápida y una menor disipación de potencia.

El diseño planteado en este trabajo es la base para desarrollar un sistema con un potencial elevado de respuesta y que se encuentra en posibilidades de aumentar su capacidad de trabajo, además de que está documentado en idioma español y que fue implementado con componentes que se adquieren con relativa facilidad en el mercado nacional, por todo lo anterior este programador representa una opción de tecnología funcional a bajo costo.

APÉNDICE A

PROGRAMAS FUENTE EN LENGUAJE C++

```
// PROGRAMA PRINCIPAL.
// Realiza todas las funciones del software del sistema PR-2000.

# include <graphics.h>
# include <stdio.h>
# include <stdlib.h>
# include <dos.h>
# include <conio.h>
# include <alloc.h>
# include "envia.h"
# include "payuda.h"
# include "panpres.h"
# include "bigchip.h"

void pantalla_de_presentacion(void);
void big_chip(void);

main(){
  int manejador_de_graficos = DETECT;
  int modo_de_graficos;

  initgraph(&manejador_de_graficos, &modo_de_graficos, "a:\\bgi");
  cleardevice();
  pantalla_de_presentacion();
  big_chip();
  closegraph();
  return 0;
}
```

```
// Programa que contiene el menú principal del sistema PR-2000
# include <stdio.h>
# include <stdlib.h>
# include <dos.h>
# include <conio.h>
# include <ctype.h>
# include "chip20.h"
# include "chip24.h"
# include "chip28.h"
# include "chip32.h"
# include "chip40.h"
# include "chipsn.h"
# include "payuda.h"
# include "opera1.h"

# define ESCAPE 27
# define RETURN 13

struct marca{
  int indice_marc;
  char nombre[20];
}lista[20];

struct dispositivo{
  int indice_dispo;
  char dispositivo1[20];
}lista_dispo1[20];

struct numero{
  int indice_num;
  char numero1[20];
}lista_num1[20];

struct maestro{              // Estructura General.
  char marca[20];
  char dispositivo[20];
  Char numero[10];
```

```
    char num_pins[3];
    int num_entradas;
    int num_salidas;
    char dps[40][6];
    char vcclec[5];
    char vpplec[5];
    char villecmin[5];
    char villectip[5];
    char villecmax[5];
    char vihlecmin[5];
    char vihlectip[5];
    char vihlecmax[5];
    char vollecmin[5];
    char vollectip[5];
    char vollecmax[5];
    char vohlecmin[5];
    char vohlectip[5];
    char vohlecmax[5];
    char vccprog[5];
    char vppprog[5];
    char vilprogmin[5];
    char vilprogtip[5];
    char vilprogmax[5];
    char vihprogmin[5];
    char vihprogtip[5];
    char vihprogmax[5];
    char pwprogmin[10];
    char pwprogtip[10];
    char pwprogmax[10];
}list_componentes[20];
int marcas;
int tipos;
int nums;
int numero_pins;
int clave_dispositivo;          // La clave dispositivo es para la
char opcion;                    // selección y no repetición de tipos
int poligono1 = { 590, 30,      // Flecha hacia arriba.
```

```
                595, 35,
                585, 35,
                590, 30 };
int poligono2 = { 600, 30,      // Flecha hacia abajo.
                610, 30,
                605, 35,
                600, 30 };

void inicia(){                  // En esta sección se limpia la estructura
  register int t;               // para evitar código inútil.
                                // Útil para la estructura maestro
  for (t=0; t<20; t++)
    lista_componentes[t].indice = 0;
}

void cargarlo(){                // Este procedimiento lee de disco los datos
  FILE *fp;                     // y prepara la estructura para manejar la
  register int i;               // informacion contenida en ella.
                                // Válido para la estructura maestro
  if((fp=fopen("archimas.dat", "rb")) == NULL){
    return;
  }

  inicia();
  for (i=0; i<10; i++){
    fread(&lista_componentes[i], sizeof(struct maestro), 1, fp);
  }
  fclose(fp);
  return;
}

void inicia_lista(){            // En esta sección se limpia la estructura
  register int t;               // para evitar código inútil.

  for (t=0; t<20; t++)
    *lista[t].nombre = '\0';
```

```
}

void cargar(){                  // Este procedimiento lee de disco los datos
  int x1, y1, x2, y2, x, y, i, j, k, cont, cont1, cont2;
  char espejo[20][20];
  int resultado;
  char auxiliar[20];

  inicia_lista();           // Limpieza de la estructura o arreglo que recibe
                            // los datos del archivo archimas.dat
  for(i=0, i<20; i++){
    if((*lista_componentes[i].marca == '\0') ||
(*lista_componentes[i].marca == NULL))
       goto copia_espejos;
  }

  copia_espejos:
  j = i;
  printf(" %d ", j);
  for(cont=0; cont<j; cont++){
    lista[cont].indice_marc = lista_componentes[cont].indice_marca;
    memmove(espejo[cont], lista_componentes[cont].marca, 20);
  }

  for(cont1=1; cont1<j; ++cont1)
    for(cont2=j-1; cont2>=cont1; --cont2){
       resultado=strcmp(espejo[cont2-1], espejo[cont2]);
       if( resultado > 0 ){
         memmove(auxiliar, espejo[cont2-1], 20);
         memmove(espejo[cont2-1], espejo[cont2], 20);
         memmove(espejo[cont2], auxiliar, 20);
         continue;
       }
       else if( resultado < 0 )
            continue;
          else if( resultado == 0){
               continue;
```

```
            }
  }

k = 0;
for(cont=0; cont<j; cont++){
  resultado = strcmp(espejo[cont], espejo[cont+1]);
  if( resultado < 0 ){
     memmove(lista[k].nombre, espejo[cont], 20);
     k = k + 1;
  }
  else
     continue;
}

i = 0;
otra_marca:
if(i < 10){        // Se cierra el archivo y se coloca el cursor
  x1=140;
  y1=170;
  x2=150,
  y2=180;
  x= 160;
  y= 170;

  for (cont=0; cont < 10; cont++){
     resultado = strcmp(lista[cont].nombre, lista[cont+1].nombre);
     if( resultado == 0 )
       continue;
     else
       settextstyle(0, 0, 1);
       outtextxy(x, y, lista[cont].nombre);
       y = y + 10;
  }

  setfillstyle(1, 15);
  bar(x1, y1, x2, y2);
  cont = 0;
```

```
    return;
  }
  else
    i = i + 1;
    goto otra_marca;
}

void inicia_lista_dispositivo(){  // En esta sección se limpia la estructura
  register int t;                 // para evitar código inútil.

  for (t=0; t<20; t++)
    *lista_dispo1[t].dispositivo1 = '\0';
}

void cargar_dispositivo(){    // Este procedimiento lee de disco los datos
  int x1, y1, x2, y2, x, y, i, j, k, cont, cont1, cont2;
  char espejo[20][20];
  int resultado;
  char auxiliar[20];

  inicia_lista_dispositivo();

  for(i=0; i<20; i++){
    if(*lista_componentes[i].dispositivo == '\0')
       goto copia_los_espejos;
  }

  copia_los_espejos:
  j = i;
  for(cont=0; cont<j; cont++){
    memmove(espejo[cont], lista_componentes[cont].dispositivo, 20);
  }

  for(cont1=1; cont1<j; ++cont1)
    for(cont2=j-1; cont2>=cont1; --cont2){
       resultado=strcmp(espejo[cont2-1], espejo[cont2]);
       if( resultado > 0 ){
```

```
        memmove(auxiliar, espejo[cont2-1], 20);
        memmove(espejo[cont2-1], espejo[cont2], 20);
        memmove(espejo[cont2], auxiliar, 20);
        continue;
      }
      else if( resultado < 0 )
            continue;
          else if( resultado == 0)
                continue;
  }

k = 0;
for(cont=0; cont<j; cont++){
  resultado = strcmp(espejo[cont], espejo[cont+1]);
  if( resultado < 0 ){
      memmove(lista_dispo1[k].dispositivo1, espejo[cont], 20);
      resultado = strcmp(lista_dispo1[k].dispositivo1, "EPROM");
      if(resultado == 0){
        lista_dispo1[k].indice_dispo = 1;
        goto siguete_de_frente;
      }
      else

     resultado=strcmp(lista_dispo1[k].dispositivo1,"MICROCONTRO
                LADOR");
      if(resultado == 0){
        lista_dispo1[k].indice_dispo = 2;
        goto siguete_de_frente;
      }
      else
      resultado = strcmp(lista_dispo1[k].dispositivo1, "PAL");
      if(resultado == 0){
        lista_dispo1[k].indice_dispo = 3;
        goto siguete_de_frente;
      }
      else
      resultado = strcmp(lista_dispo1[k].dispositivo1, "GAL");
```

```
        if(resultado == 0){
          lista_dispo1[k].indice_dispo = 4;
          goto siguete_de_frente;
        }
        else
        resultado = strcmp(lista_dispo1[k].dispositivo1, "RAL");
        if(resultado == 0){
          lista_dispo1[k].indice_dispo = 5;
          goto siguete_de_frente;
        }
        else
          lista_dispo1[k].indice_dispo = 6;
          goto siguete_de_frente;

        siguete_de_frente:
        k = k + 1;
      }
      else
        continue;
    }

    i = 0;
    otro_dispositivo:
    if(i < 10){              // Se cierra el archivo y se coloca el cursor
      x1=340;
      y1=170;
      x2=350;
      y2=180;
      x= 360;
      y= 170;

      for (cont=0; cont < 10; cont++){
        resultado = strcmp(lista_dispo1[cont].dispositivo1,
lista_dispo1[cont+1].dispositivo1);
        if( resultado == 0)
          continue,
        else
```

```
            settextstyle(0, 0, 1);
            outtextxy(x, y, lista_dispo1[cont].dispositivo1);
            y = y + 10;
        }

        setfillstyle(1, 15);
        bar(x1, y1, x2, y2);
        cont = 0;
        return;
      }
      else
        i = i + 1;
        goto otro_dispositivo;

}

void inicia_lista_numero(){
  register int t;
  for (t=0; t<20; t++)
    *lista_num1[t].numero1 = '\0';
}

void cargar_numero(){
  register int i, cont;
  int x, y, x1, y1, x2, y2, j;
  char espejo[20][20];
  int resultado;
  char auxiliar[20];

  inicia_lista_numero();

  for(i=0; i<20; i++){
    if(*lista_componentes[i].numero == '\0')
       goto copiatodo;
  }

  copiatodo:
```

```
  j = i,
  cont=0;
  otra_pregunta:
  if(lista_dispo1[tipos].indice_dispo ==
lista_componentes[cont]:indice_dispositivo)
     memmove(lista_num1[cont].numero1,
lista_componentes[cont].numero, 20);
     cont = cont + 1;
     if(cont<j)
        goto otra_pregunta;
     else
        goto reventar;

  reventar:
  i = 0;
  otro_numero:
  if(i < 10){          // Se cierra el archivo y se coloca el cursor
     x1=140,
     y1=170;
     x2=150;
     y2=180;
     x= 160;
     y= 170;

     for (cont=0; cont < 10; cont++){
        settextstyle(0, 0, 1);
        outtextxy(x, y, lista_num1[cont].numero1);
        y = y + 10;
     }

     setfillstyle(1, 15);
     bar(x1, y1, x2, y2);
     cont = 0;
     return;
  }
  else
     i = i + 1,
```

```
    goto otro_numero;
}

main(){
  int manejador_de_graficos = DETECT;
  int modo_de_graficos;
  char seleccion1, seleccion2;
  int x, y, x1, y1, x2, y2;
  int cont;

  initgraph(&manejador_de_graficos, &modo_de_graficos, "a:\\bgi");

  cargarlo();
  retorno1:
  cleardevice();
  rectangle(20, 20, 620, 460);
  rectangle(21, 21, 619, 459);
  rectangle(22, 22, 618, 458);

  setfillstyle(1, 3);       //Barra indicadora de selección
  bar(130, 40, 510, 90);

  setfillstyle(1, 2);       //Barras de la pantalla central
  bar(130, 120, 510, 140);
  bar(130, 144, 318, 160);
  bar(322, 144, 510, 160);
  bar(130, 164, 318, 350);
  bar(322, 164, 510, 350);

  setfillstyle(1, 3);       //Barra de menú de opciones
  bar(130, 380, 510, 430);

  settextstyle(0, 0, 0);    //Letreros de barras de la pantalla
  outtextxy(265, 125, "MENU PRINCIPAL");
  outtextxy(190, 149, "MARCAS");
  outtextxy(368, 149, "DISPOSITIVOS");
  outtextxy(150, 390, "<<RETURN>>  SELECCION");
```

```
outtextxy(150, 410, "    F3     ALTAS");
outtextxy(380, 390, "F2   SALIR");
outtextxy(380, 410, "F10  AYUDA");

inicia_lista();          // Se limpian los registros
cargar();                // lee datos de disco y los coloca en pantalla.
x1=140;
y1=170;
x2=150;
y2=180;
marcas=0;

otra_vez:                // Elección de la pantalla de trabajo
seleccion1 = getch();

switch( seleccion1 ){
  case 61 :  mensaje_leer_ayuda();
             goto retorno1;
  case 68 :  opcion = '!';
             ayuda(opcion);
             goto retorno1;
  case 60 :  goto final1;
  case 80 :  if( y1 > 170 )
              if( y1 == 270)
                goto otra_vez;
              else{
                setfillstyle(1, 2);
                bar(x1, y1, x2, y2);
                y1 = y1 + 10;
                y2 = y2 + 10;
                setfillstyle(1, 15),
                bar(x1, y1, x2, y2);
                marcas=marcas+1;
                seleccion1 = getch();
                if(seleccion1 == RETURN){
                   outtextxy(130, 50, lista[marcas].nombre);
                   goto dispositivos;
```

```
            }
            else
               goto otra_vez;
         }
           else
            setfillstyle(1, 2);
            bar(x1, y1, x2, y2);
            y1 = y1 + 10;
            y2 = y2 + 10;
            setfillstyle(1, 15),
            bar(x1, y1, x2, y2);
            marcas=marcas+1;
            seleccion1 = getch();
            if(seleccion1 == RETURN){
               outtextxy(130, 50, lista[marcas].nombre);
               goto dispositivos;
            }
            else
               goto otra_vez;

case 72 : if( y2 < 280 )
           if( y2 == 180 )
            goto otra_vez;
           else{
            setfillstyle(1, 2);
            bar(x1, y1, x2, y2);
            y1 = y1 - 10;
            y2 = y2 - 10;
            setfillstyle(1, 15);
            bar(x1, y1, x2, y2);
            marcas=marcas-1;
            seleccion1 = getch();
            if(seleccion1 == RETURN){
               outtextxy(130, 50, lista[marcas].nombre);
               goto dispositivos;
            }
            else
```

```
                goto otra_vez;
              }
            else
              setfillstyle(1, 2);
              bar(x1, y1, x2, y2);
              y1 = y1 - 10;
              y2 = y2 - 10;
              setfillstyle(1, 15);
              bar(x1, y1, x2, y2);
              marcas=marcas-1;
              seleccion1 = getch();
              if(seleccion1 == RETURN){
                outtextxy(130, 50, lista[marcas].nombre);
                goto dispositivos;
              }
              else
                goto otra_vez;
  case RETURN : outtextxy(130, 50, lista[marcas].nombre);
                goto dispositivos;
  default : goto otra_vez;
}

dispositivos:
setfillstyle(1, 3);
bar(150, 410, 500, 430);
outtextxy(150, 410, "ESC ANTERIOR");
inicia_lista_dispositivo();
cargar_dispositivo();
x1=340;
y1=170;
x2=350;
y2=180,
tipos=0;

otra_vez_mas:
seleccion1 = getch();
```

```
switch( seleccion1 ){
  case 60 :  goto final1;
  case 80 :  if( y1 > 170 )
               if( y1 == 270)
                 goto otra_vez_mas;
               else{
                 setfillstyle(1, 2);
                 bar(x1, y1, x2, y2);
                 y1 = y1 + 10;
                 y2 = y2 + 10;
                 setfillstyle(1, 15),
                 bar(x1, y1, x2, y2);
                 tipos=tipos+1;
                 seleccion1 = getch();
                 if(seleccion1 == RETURN){
                    outtextxy(270, 50, lista_dispo1[tipos].dispositivo1);
                    clave_dispositivo = lista_dispo1[tipos].indice_dispo;
                    goto numeros;
                 }
                 else
                    goto otra_vez_mas;
             }
               else
                 setfillstyle(1, 2);
                 bar(x1, y1, x2, y2);
                 y1 = y1 + 10;
                 y2 = y2 + 10;
                 setfillstyle(1, 15),
                 bar(x1, y1, x2, y2);
                 tipos=tipos+1;
                 seleccion1 = getch();
                 if(seleccion1 == RETURN){
                    outtextxy(270, 50, lista_dispo1[tipos].dispositivo1);
                    clave_dispositivo = lista_dispo1[tipos].indice_dispo;
                    goto numeros;
                 }
                 else
```

```
                goto otra_vez_mas;

case 72 : if( y2 < 280 )
          if( y2 == 180 )
            goto otra_vez_mas;
          else{
            setfillstyle(1, 2);
            bar(x1, y1, x2, y2);
            y1 = y1 - 10;
            y2 = y2 - 10;
            setfillstyle(1, 15);
            bar(x1, y1, x2, y2);
            tipos=tipos-1;
            seleccion1 = getch();
            if(seleccion1 == RETURN){
               outtextxy(270, 50, lista_dispo1[tipos].dispositivo1);
               clave_dispositivo = lista_dispo1[tipos].indice_dispo;
               goto numeros;
            }
            else
               goto otra_vez_mas;
          }
        else
          setfillstyle(1, 2);
          bar(x1, y1, x2, y2);
          y1 = y1 - 10;
          y2 = y2 - 10;
          setfillstyle(1, 15);
          bar(x1, y1, x2, y2);
          tipos=tipos-1;
          seleccion1 = getch();
          if(seleccion1 == RETURN){
            outtextxy(270, 50, lista_dispo1[tipos].dispositivo1);
            clave_dispositivo = lista_dispo1[tipos].indice_dispo;
            goto numeros;
          }
          else
```

```
                goto otra_vez_mas;
  case RETURN : outtextxy(270, 50, lista_dispo1[tipos].dispositivo1);
                clave_dispositivo = lista_dispo1[tipos].indice_dispo;
                goto numeros;
  case 27 : goto retorno1;
  default : goto otra_vez_mas;
}

numeros:
retorno2:
cleardevice();
rectangle(20, 20, 620, 460);
rectangle(21, 21, 619, 459);
rectangle(22, 22, 618, 458);

setfillstyle(1, 3);       //Barra indicadora de selección
bar(130, 40, 510, 90);

setfillstyle(1, 2);       //Barras de la pantalla central
bar(130, 120, 510, 140);
bar(130, 144, 510, 160);
bar(130, 164, 510, 350);

setfillstyle(1, 3);       //Barra de menú de opciones
bar(130, 380, 510, 430);

settextstyle(0, 0, 0);    //Letreros de barras de la pantalla
outtextxy(265, 125, "MENU PRINCIPAL");
outtextxy(290, 149, "NUMEROS");
outtextxy(150, 390, "<<RETURN>>  SELECCION");
outtextxy(150, 410, "ESC ANTERIOR");
outtextxy(380, 390, "F2   SALIR");
outtextxy(380, 410, "F10  AYUDA");
outtextxy(130, 50, lista[marcas].nombre);
outtextxy(270, 50, lista_dispo1[tipos] dispositivo1);
drawpoly(4, poligono1);
drawpoly(4, poligono2),
```

```
inicia_lista_numero();
cargar_numero();

x1=140;
y1=170;
x2=150;
y2=180;
nums=0;

otra_vez2:
seleccion2 = getch();

switch( seleccion2 ){
  case 59 :  return 0;
  case 68 :  opcion = '$';
             ayuda(opcion);
             goto retorno2;
  case 60 :  goto final1;
  case 80 :  if( y1 > 170 )
               if( y1 == 270)
                 goto otra_vez2;
               else{
                 setfillstyle(1, 2);
                 bar(x1, y1, x2, y2);
                 y1 = y1 + 10;
                 y2 = y2 + 10;
                 setfillstyle(1, 15),
                 bar(x1, y1, x2, y2);
                 nums=nums+1;
                 seleccion1 = getch();
                 if(seleccion1 == RETURN){
                    outtextxy(420, 50, lista_num1[nums].numero1);
                    setfillstyle(1, 0);
                    bar(130, 380, 510, 430);
                    setfillstyle(1, 3);
                    bar(460, 420, 470, 430);
                    delay(1000);
```

```
                    bar(480, 420, 490, 430);
                    delay(1000);
                    bar(500, 420, 510, 430);
                    delay(1000);
                    numero_pins =
atoi(lista_componentes[nums].num_pins);
                    pantalla_de_operaciones(clave_dispositivo,
numero_pins);
                    return 0;
                  }
                  else
                    goto otra_vez2;
                }
              else
                setfillstyle(1, 2);
                bar(x1, y1, x2, y2);
                y1 = y1 + 10;
                y2 = y2 + 10;
                setfillstyle(1, 15),
                bar(x1, y1, x2, y2);
                nums=nums+1;
                seleccion1 = getch();
                if(seleccion1 == RETURN){
                  outtextxy(420, 50, lista_num1[nums].numero1);
                  setfillstyle(1, 0);
                  bar(130, 380, 510, 430);
                  setfillstyle(1, 3);
                  bar(460, 420, 470, 430);
                  delay(1000);
                  bar(480, 420, 490, 430);
                  delay(1000);
                  bar(500, 420, 510, 430);
                  delay(1000);
                  numero_pins =
atoi(lista_componentes[nums].num_pins);
                  pantalla_de_operaciones(clave_dispositivo,
numero_pins);
```

```
                return 0;
              }
              else
                goto otra_vez2;

    case 72 : if( y2 < 280 )
              if( y2 == 180 )
                goto otra_vez2;
              else{
                setfillstyle(1, 2);
                bar(x1, y1, x2, y2);
                y1 = y1 - 10;
                y2 = y2 - 10;
                setfillstyle(1, 15);
                bar(x1, y1, x2, y2);
                nums=nums-1;
                seleccion1 = getch();
                if(seleccion1 == RETURN){
                   outtextxy(420, 50, lista_num1[nums].numero1);
                   setfillstyle(1, 0);
                   bar(130, 380, 510, 430);
                   setfillstyle(1, 3);
                   bar(460, 420, 470, 430);
                   delay(1000);
                   bar(480, 420, 490, 430);
                   delay(1000);
                   bar(500, 420, 510, 430);
                   delay(1000);
                   numero_pins =
atoi(lista_componentes[nums].num_pins);
                   pantalla_de_operaciones(clave_dispositivo,
numero_pins);
                   return 0;
                }
                else
                   goto otra_vez2;
              }
```

```
            else
              setfillstyle(1, 2);
              bar(x1, y1, x2, y2);
              y1 = y1 - 10;
              y2 = y2 - 10;
              setfillstyle(1, 15);
              bar(x1, y1, x2, y2);
              nums=nums-1;
              seleccion1 = getch();
              if(seleccion1 == RETURN){
                outtextxy(420, 50, lista_num1[nums].numero1);
                setfillstyle(1, 0);
                bar(130, 380, 510, 430);
                setfillstyle(1, 3);
                bar(460, 420, 470, 430);
                delay(1000);
                bar(480, 420, 490, 430);
                delay(1000);
                bar(500, 420, 510, 430);
                delay(1000);
                numero_pins =
atoi(lista_componentes[nums].num_pins);
                pantalla_de_operaciones(clave_dispositivo,
numero_pins);
                return 0;
              }
              else
                goto otra_vez2;
    case RETURN : outtextxy(420, 50, lista_num1[nums].numero1);
              setfillstyle(1, 0);
              bar(130, 380, 510, 430);
              setfillstyle(1, 3);
              bar(460, 420, 470, 430);
              delay(1000);
              bar(480, 420, 490, 430);
              delay(1000);
              bar(500, 420, 510, 430);
```

```
                delay(1000);
                numero_pins = atoi(lista_componentes[nums].num_pins);
                pantalla_de_operaciones(clave_dispositivo,
numero_pins);
                goto retorno1;
    case 27     : goto retorno1;
    default     : goto otra_vez2;
  }

  final1:
  cleardevice();
  closegraph();
  exit(0);
  return 0;
}

mensaje_leer_ayuda(){
  char pato[2];
  char elige;

  cleardevice();
  setfillpattern(pato, 2);
  bar(0, 0, 640, 480);
  setusercharsize(3, 2, 3, 2);
  settextstyle(2, 0, 0);
  outtextxy(180, 100, "PARA EVITAR QUE SU DISPOSITIVO");
  outtextxy(180, 140, "SUFRA DAÑOS Y EL SISTEMA PRESENTE");
  outtextxy(180, 180, "ANOMALIAS EN SU FUNCIONAMIENTO LE");
  outtextxy(180, 220, "SUGERIMOS CONSULTAR LA AYUDA DEL");
  outtextxy(180, 260, "MENÚ PRINCIPAL, EN CASO CONTRARIO");
  outtextxy(180, 300, "PUEDE USTED CONTINUAR.");
  setfillstyle(1, 2);
  bar(140, 350, 300, 375);
  bar(380, 350, 510, 375);
  outtextxy(145, 355, "<<ESC>> REGRESAR");
  outtextxy(389, 355, "F4 CONTINUAR");
  teclea:
```

```
    elige = getch();
    switch( elige ){
      case 62        : goto comienza_altas;
      case ESCAPE :  return 0;
      default        : goto teclea;
    }
    comienza_altas:
    cleardevice();
    closegraph();
    exit(2);
    return 0;
}
```

```
// Este programa es la combinación para dar de alta un dispositivo.
# include <stdio.h>
# include <stdlib.h>
# include <dos.h>
# include <conio.h>
# include <string.h>
# include "chip20.h"
# include "chip24.h"
# include "chip28.h"
# include "chip32.h"
# include "chip40.h"
# include "capta.h"      // captura_datos.

# define campana 7

void pantalla_de_aviso_de_cancelado(void);

main(){

  char selecc_tecla;
  int x1, y1, x2, y2;
  int oportunidad = 0;
  int t;
  int a, b;

  vuelve.
  cleardevice();

  setfillstyle(1, 1);
  bar(40, 160, 50, 330);
  bar(50, 320, 170, 330);
  setfillstyle(1, 3);
  bar(50, 150, 180, 320);
  delay(100);
  setfillstyle(1, 1);
  bar(170, 140, 180, 310);
  bar(180, 300, 300, 310);
```

```
setfillstyle(1, 9);
bar(180, 130, 310, 300);
delay(100);
setfillstyle(1, 1);
bar(300, 120, 310, 290);
bar(310, 280, 430, 290);
setfillstyle(1, 6);
bar(310, 110, 440, 280);
delay(100);
setfillstyle(1, 1);
bar(430, 100, 440, 270);
bar(440, 260, 560, 270);
setfillstyle(1, 2);
bar(440, 90, 570, 260);
delay(100);

setfillstyle(1, 1);
bar(480, 348, 600, 367);
settextstyle(0, 0, 1);
outtextxy(590, 353, "9");
delay(80);
outtextxy(578, 353, "8");
delay(80);
outtextxy(566, 353, "7");
delay(80);
outtextxy(554, 353, "6");
delay(80);
outtextxy(542, 353, "5");
delay(80);
outtextxy(530, 353, "4");
delay(80);
outtextxy(518, 353, "3");
delay(80);
outtextxy(506, 353, "2");
delay(80);
outtextxy(494, 353, "1");
delay(80);
```

```
outtextxy(482, 353, "0");
delay(80);

setfillstyle(1, 1);
bar(480, 395, 600, 413);
outtextxy(505, 400, "F2  SALIR");

x1=482;
y1=362,
x2=488;
y2=364;

setfillstyle(1, 15);
bar(x1, y1, x2, y2);

oprime:
selecc_tecla= getch();
switch( selecc_tecla ){
  case 77 : if( x1 > 482 )
              if( x1 == 590)
                goto oprime;
              else{
                setfillstyle(1, 1);
                bar(x1, y1, x2, y2);
                x1 = x1 + 12;
                x2 = x2 + 12;
                setfillstyle(1, 15);
                bar(x1, y1, x2, y2);
                goto oprime;
              }
            else
              setfillstyle(1, 1);
              bar(x1, y1, x2, y2);
              x1 = x1 + 12;
              x2 = x2 + 12;
              setfillstyle(1, 15);
              bar(x1, y1, x2, y2);
```

```
            goto oprime;
case 75 : if( x1 < 590 )
            if( x1 == 482 )
              goto oprime;
            else{
              setfillstyle(1, 1);
              bar(x1, y1, x2, y2);
              x1 = x1 - 12;
              x2 = x2 - 12;
              setfillstyle(1, 15);
              bar(x1, y1, x2, y2);
              goto oprime;
            }
          else
            setfillstyle(1, 1);
            bar(x1, y1, x2, y2);
            x1 = x1 - 12;
            x2 = x2 - 12;
            setfillstyle(1, 15);
            bar(x1, y1, x2, y2);
            bar(x1, y1, x2, y2);
            goto oprime;
case 60 : goto final_programa_combinacion;
case 13 : oportunidad = oportunidad + 1;
          switch(x1){
            case 530 : if ((oportunidad==1)||(oportunidad==3)){
                       oprime2:
                       selecc_tecla= getch();
                       switch( selecc_tecla ){
                         case 77 : if( x1 > 482 )
                                   if( x1 == 590)
                                     goto oprime2;
                                   else{
                                     setfillstyle(1, 1);
                                     bar(x1, y1, x2, y2);
                                     x1 = x1 + 12;
                                     x2 = x2 + 12;
```

```
                setfillstyle(1, 15);
                bar(x1, y1, x2, y2);
                goto oprime2;
              }
            else
              setfillstyle(1, 1);
              bar(x1, y1, x2, y2);
              x1 = x1 + 12;
              x2 = x2 + 12;
              setfillstyle(1, 15);
              bar(x1, y1, x2, y2);
              goto oprime2;
case 75 : if( x1 < 590 )
              if( x1 == 482 )
                goto oprime2;
              else{
                setfillstyle(1, 1);
                bar(x1, y1, x2, y2);
                x1 = x1 - 12;
                x2 = x2 - 12;
                setfillstyle(1, 15);
                bar(x1, y1, x2, y2);
                goto oprime2;
              }
            else
              setfillstyle(1, 1);
              bar(x1, y1, x2, y2);
              x1 = x1 - 12;
              x2 = x2 - 12;
              setfillstyle(1, 15);
              bar(x1, y1, x2, y2);
              bar(x1, y1, x2, y2);
              goto oprime2;
case 60 : goto final_programa_combinacion;
case 13 : oportunidad = oportunidad + 1;
          switch(x1){
              case 530 : settextstyle(0, 0, 3),
```

```
outtextxy(93, 223, "2");
delay(80);
outtextxy(228, 203, "0");
delay(80);
outtextxy(363, 183, "0");
delay(80);
outtextxy(498, 163, "0");
delay(80);
for (t=600; t>=480; t=t-10){
  setfillstyle(1, 0);
  bar(600, 348, t, 367);
  delay(80);
}
for (t=0; t<=2; t=t+1){
  setfillstyle(1, 1);
  bar(480, 395, 600, 413);
  delay(500);
  settextstyle(0, 0, 1);
  outtextxy(500, 400,
            "¡ ACCESO !");
  printf("%c", campana);
  delay(400);
}
setfillstyle(1, 0);
bar(0, 0, 640, 480);
setfillstyle(1, 10);
bar(310, 364, 610, 414);
b = 416;
for(a=362; a>=318; a=a-2){
  setfillstyle(1, 2);
  bar(310, 364, 610, a);
  delay(50);
  bar(310, 414, 610, b);
  b = b+2;
}
b = 413;
for(a=365; a<=389; a=a+1){
```

```
                setfillstyle(1, 2);
                bar(310, 364, 610, a);
                delay(25);
                bar(310, 414, 610, b);
                b = b-1;
              }
              setusercharsize(3, 2, 3, 2);
              settextstyle(2, 0, 0);
              outtextxy(333, 340, "¡ LAS
                 CORRECCIONES SE ");
              outtextxy(333, 380,
                 "EFECTUARAN AL
                 FINALIZAR ");
              outtextxy(333, 420, "CADA
                 PANTALLA DE
                 CAPTURA  !");
              sound(120);
              delay(250);
              nosound();
              delay(2000);
              capta();
              break;
default : if (oportunidad==2){
              setfillstyle(1, 1);
              bar(480, 395, 600, 413);
              outtextxy(505, 400,
                "¡ ERROR !");
              delay(800);
              setfillstyle(1, 1);
              bar(480, 395, 600, 413),
              outtextxy(505, 400, "F2
                      SALIR");
              goto oprime;
             }
            if (oportunidad==4)
               pantalla_de_aviso_de_
               cancelado();
```

```
                                    }
                                }
                            }
                        default  : if (oportunidad==2){
                                    setfillstyle(1, 1);
                                    bar(480, 395, 600, 413);
                                    printf("%c", campana);
                                    outtextxy(505, 400, "- ERROR !");
                                    delay(500);
                                    setfillstyle(1, 1);
                                    bar(480, 395, 600, 413);
                                    outtextxy(505, 400, "F2  SALIR");
                                    goto oprime;
                                  }
                                  if (oportunidad==4)
                                    pantalla_de_aviso_de_cancelado();
                    }
        default   : goto oprime;
    }

    final_programa_combinacion:
    cleardevice();
    closegraph();
    exit(0);
    return 0;
}

void pantalla_de_aviso_de_cancelado(){

    cleardevice();
    setfillstyle(1, 2);
    bar(150, 120, 160, 300);
    bar(160, 290, 490, 300);
    setfillstyle(1, 9);
    bar(160, 110, 500, 290),
    settextstyle(0, 0, 2);
    printf("%c", campana);
```

```
    outtextxy(170, 190, "¡ ACCESO CANCELADO !");
    printf("%c", campana);
    delay(1500);
    cleardevice();
    closegraph();
    exit(0);
    return;
}
```

```
// Este programa es la llave para dar de alta un dispositivo.
# include <stdio.h>
# include <stdlib.h>
# include <dos.h>
# include <conio.h>
# include <string.h>
# include "chip20.h"
# include "chip24.h"
# include "chip28.h"
# include "chip32.h"
# include "chip40.h"
# include "capta.h"    // captura_datos.

# define campana 7

void pantalla_de_aviso_de_cancelado(void);

main(){

  char selecc_tecla;
  int x1, y1, x2, y2;
  int oportunidad = 0;
  int t;
  int a, b;

  vuelve:
  cleardevice();

  setfillstyle(1, 1);
  bar(40, 160, 50, 330);
  bar(50, 320, 170, 330),
  setfillstyle(1, 3);
  bar(50, 150, 180, 320);
  delay(100);
  setfillstyle(1, 1);
  bar(170, 140, 180, 310);
  bar(180, 300, 300, 310);
```

```
setfillstyle(1, 9);
bar(180, 130, 310, 300);
delay(100);
setfillstyle(1, 1);
bar(300, 120, 310, 290);
bar(310, 280, 430, 290);
setfillstyle(1, 6);
bar(310, 110, 440, 280);
delay(100);
setfillstyle(1, 1);
bar(430, 100, 440, 270);
bar(440, 260, 560, 270);
setfillstyle(1, 2);
bar(440, 90, 570, 260);
delay(100);

setfillstyle(1, 1);
bar(480, 348, 600, 367);
settextstyle(0, 0, 1);
outtextxy(590, 353, "9");
delay(80);
outtextxy(578, 353, "8");
delay(80);
outtextxy(566, 353, "7");
delay(80);
outtextxy(554, 353, "6");
delay(80);
outtextxy(542, 353, "5");
delay(80);
outtextxy(530, 353, "4");
delay(80);
outtextxy(518, 353, "3");
delay(80);
outtextxy(506, 353, "2");
delay(80);
outtextxy(494, 353, "1");
delay(80);
```

```
outtextxy(482, 353, "0");
delay(80);

setfillstyle(1, 1);
bar(480, 395, 600, 413);
outtextxy(505, 400, "F2  SALIR");

x1=482;
y1=362;
x2=488;
y2=364;

setfillstyle(1, 15);
bar(x1, y1, x2, y2);

oprime:
selecc_tecla= getch();
switch( selecc_tecla ){
  case 77 : if( x1 > 482 )
              if( x1 == 590)
                goto oprime;
              else{
                setfillstyle(1, 1);
                bar(x1, y1, x2, y2);
                x1 = x1 + 12;
                x2 = x2 + 12;
                setfillstyle(1, 15);
                bar(x1, y1, x2, y2);
                goto oprime;
              }
            else
              setfillstyle(1, 1);
              bar(x1, y1, x2, y2);
              x1 = x1 + 12,
              x2 = x2 + 12;
              setfillstyle(1, 15);
              bar(x1, y1, x2, y2),
```

```
            goto oprime;
case 75 : if( x1 < 590 )
            if( x1 == 482 )
              goto oprime;
            else{
              setfillstyle(1, 1);
              bar(x1, y1, x2, y2);
              x1 = x1 - 12;
              x2 = x2 - 12;
              setfillstyle(1, 15);
              bar(x1, y1, x2, y2);
              goto oprime;
            }
          else
            setfillstyle(1, 1);
            bar(x1, y1, x2, y2);
            x1 = x1 - 12;
            x2 = x2 - 12;
            setfillstyle(1, 15);
            bar(x1, y1, x2, y2);
            bar(x1, y1, x2, y2);
            goto oprime;
case 60 : goto final_programa_combinacion;
case 13 : oportunidad = oportunidad + 1;
          switch(x1){
            case 530 : if ((oportunidad==1)||(oportunidad==3)){
                       oprime2:
                       selecc_tecla= getch();
                       switch( selecc_tecla ){
                         case 77 : if( x1 > 482 )
                                   if( x1 == 590)
                                     goto oprime2,
                                   else{
                                     setfillstyle(1, 1);
                                     bar(x1, y1, x2, y2);
                                     x1 = x1 + 12;
                                     x2 = x2 + 12;
```

```
                setfillstyle(1, 15);
                bar(x1, y1, x2, y2);
                goto oprime2;
              }
            else
              setfillstyle(1, 1);
              bar(x1, y1, x2, y2);
              x1 = x1 + 12;
              x2 = x2 + 12;
              setfillstyle(1, 15);
              bar(x1, y1, x2, y2);
              goto oprime2;
case 75 : if( x1 < 590 )
              if( x1 == 482 )
                goto oprime2;
              else{
                setfillstyle(1, 1);
                bar(x1, y1, x2, y2);
                x1 = x1 - 12;
                x2 = x2 - 12;
                setfillstyle(1, 15);
                bar(x1, y1, x2, y2);
                goto oprime2;
              }
            else
              setfillstyle(1, 1);
              bar(x1, y1, x2, y2);
              x1 = x1 - 12;
              x2 = x2 - 12;
              setfillstyle(1, 15);
              bar(x1, y1, x2, y2),
              bar(x1, y1, x2, y2);
              goto oprime2;
case 60 : goto final_programa_combinacion;
case 13 . oportunidad = oportunidad + 1;
          switch(x1){
              case 530 : settextstyle(0, 0, 3);
```

```
outtextxy(93, 223, "2");
delay(80);
outtextxy(228, 203, "0");
delay(80);
outtextxy(363, 183, "0");
delay(80);
outtextxy(498, 163, "0");
delay(80);
for (t=600; t>=480; t=t-10){
    setfillstyle(1, 0);
  bar(600, 348, t, 367);
  delay(80);
}
for (t=0; t<=2; t=t+1){
  setfillstyle(1, 1);
  bar(480, 395, 600, 413);
  delay(500);
  settextstyle(0, 0, 1);
  outtextxy(500, 400,
          "¡ ACCESO !");
  printf("%c", campana);
  delay(400);
}
setfillstyle(1, 0);
bar(0, 0, 640, 480);
setfillstyle(1, 10);
bar(310, 364, 610, 414);
b = 416;
for(a=362, a>=318, a=a-2){
  setfillstyle(1, 2);
  bar(310, 364, 610, a);
  delay(50);
  bar(310, 414, 610, b);
  b = b+2;
}
b = 413;
for(a=365; a<=389; a=a+1){
```

```
                setfillstyle(1, 2);
                bar(310, 364, 610, a);
                delay(25);
                bar(310, 414, 610, b);
                b = b-1;
              }
              setusercharsize(3, 2, 3, 2);
              settextstyle(2, 0, 0);
              outtextxy(333, 340, "¡ LAS
                  CORRECCIONES SE ");
              outtextxy(333, 380,
                  " EFECTUARAN AL
                  FINALIZAR ");
              outtextxy(333, 420, " CADA
                  PANTALLA DE
                  CAPTURA !");
              sound(120);
              delay(250);
              nosound();
              delay(2000);
              capta();
              break;
default : if (oportunidad==2){
              setfillstyle(1, 1);
              bar(480, 395, 600, 413);
              outtextxy(505, 400,
                        "¡ ERROR !");
              delay(800);
              setfillstyle(1, 1);
              bar(480, 395, 600, 413);
              outtextxy(505, 400, "F2
                        SALIR");
              goto oprime;
            }
            if (oportunidad==4)
                pantalla_de_aviso_de_
                cancelado();
```

```
                                    }
                                }
                            }
                        default  : if (oportunidad==2){
                                    setfillstyle(1, 1);
                                    bar(480, 395, 600, 413);
                                    printf("%c", campana);
                                    outtextxy(505, 400, "¡ ERROR !");
                                    delay(500);
                                    setfillstyle(1, 1);
                                    bar(480, 395, 600, 413);
                                    outtextxy(505, 400, "F2  SALIR");
                                    goto oprime;
                                  }
                                  if (oportunidad==4)
                                    pantalla_de_aviso_de_cancelado();
                    }
        default   : goto oprime;
    }

    final_programa_combinacion:
    cleardevice();
    closegraph();
    exit(0);
    return 0;
}

void pantalla_de_aviso_de_cancelado(){

    cleardevice();
    setfillstyle(1, 2);
    bar(150, 120, 160, 300);
    bar(160, 290, 490, 300);
    setfillstyle(1, 9);
    bar(160, 110, 500, 290);
    settextstyle(0, 0, 2);
    printf("%c", campana);
```

```
    outtextxy(170, 190, "¡ ACCESO CANCELADO !");
    printf("%c", campana);
    delay(1500);
    cleardevice();
    closegraph();
    exit(0);
    return;
}
```

BIBLIOGRAFÍA

- Atkinson, Lee y Atkinson, Mark. Using Borlandc C++3 2nd Edition. QUE Corporation, E.U.A., 1992.

- Boylestad, Robert y Nashelsky Louis. Electrónica Teoría de Circuitos. Prentice Hall Hispanoamericana, S.A., México, 1990.

- Jamsa, Kris. *Programación en DOS*. Osborne McGraw Hill, España, 1992

-Joyanes Aguilar Luis. Borlandc C++, Manual de Bolsillo. Mc Graw Hill, España, 1993.

-Larsen, Glenn y Larsen, Kristopher. Domine Harvard Graphics 3. Adisson-Wesley Iberoamericana, E. U. A., 1993.

- Sedra, Adel A. y Smith, Kenneth. Dispositivos Electrónicos y Amplificación de Señales. McGraw-Hill, México, 1989.

- Schiltd, Herbert. C Manual de Referencia. Osborne Mc Graw Hill, España, 1988.