UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

SISTEMA DE DESPLIEGUE DE SIGNOS VITALES BASADO EN EL MICROCONTROLADOR 68HC11

T E S I S

QUE PARA OBTENER EL TITULO DE :

INGENIERO MECANICO ELECTRICISTA

(AREA ELECTRICA - ELECTRONICA)

P R E S E N T A :

SANTOS MARTIN LOPEZ ESTRADA

Director de Tesis: M. I. Esaú Vicente Vivas

MEXICO, D. F.

1994

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

Agradecimientos

En estas líneas quiero agradecer a todas aquellas personas que conciente e inconcientemente colaboraron en el diseño e implementación del presente trabajo, agradeciendo de manera especial a las siguientes personas:

A mis padres Daniel y Tomasa por la oportunidad y el apoyo brindado para lograr mis metas y objetivos.

A mis hermanos Daniel, Alejandro y Noemi por su apoyo brindado.

A Isabel por su apoyo y comprensión brindados todo este tiempo para cumplir con mis metas.

A familiares, amigos y maestros que colaboraron de una u otra manera en mi formación académica.

A Esaú Vicente por su apoyo, asesoria y paciencia brindados para la realización de este trabajo.

A Donaldo Rojas por la asesoria técnica brindada.

A Ricardo, Mauricio y Leonardo por el magnifico grupo de trabajo formado.

Al Instituto de Ingenieria, a la coordinación de automatización y a sus becarios por permitirme usar sus instalaciones para la elaboración del presente trabajo.

A todos mil gracias.

Santos Martín López Estrada

INTRODUCCIÓN

1.1 .- Antecedentes

Con los avances tecnológicos recientes en las áreas de electrónica y computación, ha tomado gran auge el diseño de equipos digitales basados en microprocesadores y microcontroladores, siendo estos últimos y los dispositivos programables, los que constituyen una magnifica herramienta para diseñar equipos atractivos en cuanto a costo, volumen, peso, modularidad y versatilidad de operación. Bajo este entorno destacan las oportunidades para países como el nuestro para contribuir a solucionar problemas de automatización, entre los que se encuentran los relacionados con equipo biomédico; en esta área, un problema en particular lo constutuye el alto costo de equipos importados que realizan el monitoreo y despliegue de signos vitales tales como el electrocardiograma (ECG), frecuencia respiratoria (FR), frecuencia cardiaca (FC), presión sanguínea (PS) y temperatura (T).

En nuestro país, en los centros de investigación de universidades y centros de salud pública, se han realizado diversos diseños de equipo biomédico para el

monitoreo de signos vitales; equipos que se fabrican en pequeña escala y que son utilizados en el ambiente clínico con buenos resultados, como es el caso del electrocardiotacómetro, monitor de frecuencia cardiaca, monitor de electrocardiografía entre otros [1]. Estos equipos se han diseñado y probado en el centro de desarrollo y aplicaciones tecnológicas (CEDAT) de la secretaría de salubridad y asistencia (SSA); como es el caso también de la terminal de elctrocardiografía desarrollada por el Instituto Nacional de Cardiología Ignacio Chavez y los diseños realizados en la Universidad Autónoma Metropolitana (UAM) para la detección de ECG. Sin embargo todos estos equipos están dedicados a la detección y tratamiento de unas u otras señales vitales y no engloban las señales arriba mencionadas, otro aspecto importante de ellos es que algunos requieren de una computadora personal (PC) para el despliegue y procesamiento de las señales y otros solamente presentan información numérica de la FC, lo cual los hace de bajo costo, pero de una no muy buena presentación para el personal médico.

El presente trabajo aunado a otros desarrollos previos representan una opción para solucionar algunos de los problemas de los equipos nacionales, como son el monitoreo y el despliegue de signos vitales en un solo equipo, incluyendo graficación de la señal de ECG, además de proponer un sistema que puede ser utilizado en red. Se presenta el diseño de un equipo electrónico digital, modular y de bajo costo para el monitoreo y despliegue de signos vitales, englobando ECG, FC, FR, PS y T, incluyendo un sistema de alarmas visuales y auditivas en caso de que se presenten signos vitales anómalos en el paciente. El despliegue de signos vitales se realiza de manera global, es decir, conectando varios equipos a un puesto central de enfermeras (PCE) constituido por una PC, formando así una red, en la cual se pueden monitorear uno o varios pacientes a la vez, ver figura 1.1. Por sus características de modularidad y versatilidad de operación el equipo podrá ser usado de manera individual en pequeños consultorios durante auscultaciones o bien de manera global en hospitales para el monitoreo de pacientes, en salas de cuidados intensivos, salas de recuperación, pruebas de esfuerzo, etcétera.

En los siguientes párrafos se describe de manera general la estructura y funcionamiento del equipo. Es importante mencionar que las señales vitales son detectadas, amplificadas y filtradas por un módulo analógico, el cual no forma parte de la presente tesis, pero que, por la participación en su diseño y construcción se menciona para tener una visión global del proyecto, dicho módulo fue diseñadoen una

tesis adicional de licenciatura y cumple con las normas de protección y aislamiento tanto para el paciente como para el equipo, además de otras que se mencionarán posteriormente. Así mismo el módulo FSK representado en al figura 1.1 constituye un trabajo complementario para la presente tesis.

El equipo, de manera general, fue diseñado para monitorear las señales vitales antes mencionadas, previamente amplificadas y filtradas por el módulo analógico, dichas señales son muestreadas, procesadas y almacenadas temporalmente en RAM, para ser enviadas posteriormente al PCE para su despliegue. El equipo está formado básicamente por cuatro módulos o etapas, las cuales se describen a continuación.

1.2 .- Módulo de amplificación y filtrado

Este modulo analógico está compuesto principalmente por transductores, filtros y amplificadores, y es el encargado de realizar la detección amplificación y filtrado de las señales vitales del paciente. Este módulo representa una parte fundamental en la detección de signos vitales, por lo que su diseño estuvo asesorado por personal especializado en el ramo de la Ingeniería Biomédica, básicamente cuenta con etapas de: selección de derivación[2],[3], preamplificación, aislamiento, amplificación y filtrado; fue diseñado para cumplir con las normas de construcción de equipo para monitoreo cardiaco AAMI (Association for the Advancement of Medical Instrumentation), en el siguiente capítulo se describe de manera general su funcionamiento.

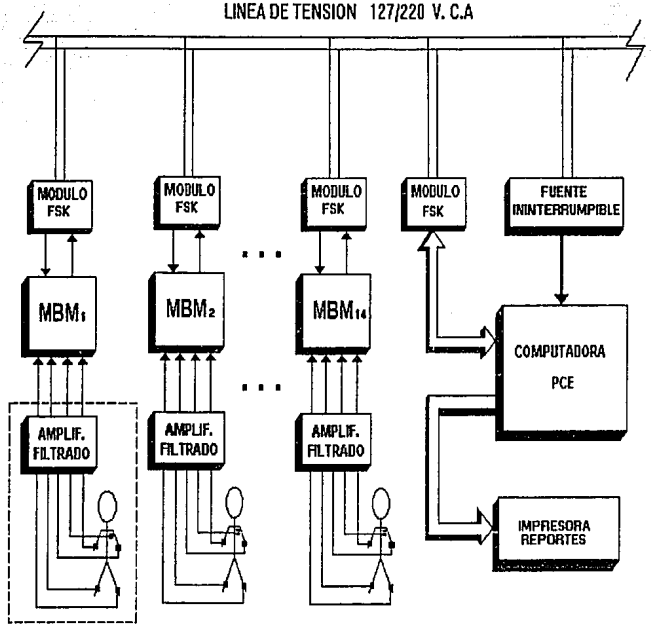


Figura 1.1 Módulos monitores en red.

1.3 .- Módulo básico de monitoreo

El módulo básico de monitoreo (MBM), esta constituido principalmente por el microcontrolador MC68HC11E9, del cual se hace uso extensivo de sus periféricos internos para reducir el número de componentes requeridos para el equipo. En este módulo se lleva acabo el control del sistema local, es decir, se controla el convertidor analógico-digital (CA/D) para muestrear los signos vitales; se direcciona memoria RAM para el almacenado temporal de los mismos, de tal forma que funcione como una memoria de video y se pueda integrar en el futuro un sistema de despliegue adecuado, como un monitor convencional de TV o un display de cristal liquido; se monitorean los puertos de entrada, para la detección de electrodos sueltos en el paciente; se controla el puerto serie para la recepción y transmisión de información entre el MBM y el PCE en caso de encontrarse conectados en red. Ademas se realiza el "procesamiento" de las señales, es decir, se realiza el cálculo de FR y FC de la señal ECG . En el capitulo 2 se presenta en detalle la arquitectura diseñada para el MBM, así como su funcionamiento, mientras que en el capítulo 4 se presentan los algoritmos de programación para el control e interacción del sistema con el PCE.

1.4 .- Enlace de comunicaciones

Cuando el MBM se encuentra conectado en red es necesario tener un medio de comunicación con el PCE, y tomando en cuenta que éste no siempre se encuentra en el mismo piso o se encuentra alejado de los pacientes, existen deversos medios de comunicación para resolver el problema, por ejemplo un sistema de radio, línea telefónica, o bien, un cableado adicional para la conexión entre equipos; sin embargo uno de los objetivos de la tesis es el bajo costo y la modularidad del equipo, por lo que ninguna de las soluciones anteriores resuelve el problema, algunas por resultar demasiado costosas, otras por requerir permisos para la transmisión y otras hacen al sistema poco transportable. Debido a estos problemas se optó por realizar un enlace de comunicaciones vía red eléctrica de baja tensión, ya que ésta se encuentra instalada por todo el inmueble y sólo requiere de un modem especial de bajo costo y diseño versátil para la transmisión serial de datos al PCE. Para el diseño de este

modem se utilizó la técnica de modulación digital de onda continua FSK (frecuency shift keying), así como un sistema de aislamiento de la línea; en el capítulo 3 se presenta el funcionamiento básico del modem, así como sus características principales.

1.5 .- Puesto central de enfermeras

Para el monitoreo global (varios MBM) o individual de pacientes conectados a la red se requiere de una computadora central ubicada en el PCE, la cual desplegará los signos vitales de hasta 14 MBM's, para esta cantidad de equipos podremos graficar solamente 150 muestras de ECG para cada uno, el número de muestras solamente está determinado por la resolución del monitor (VGA) de la computadora central. El número demuestras graficadas es de resolución baja, sin embargo, el objetivo del monitoreo global es la detección de ausencia de ECG y alarmas sobre signos vitales anómalos en el paciente, los cuales en caso de presentarse se podrán monitorear de manera individual cada conectado a la red MBM, es decir se podrá desplegar el equivalente de 250 muestras por segundo. Además de los programas para manejar las gráficas descritas anteriormente, el PCE cuenta con programas de interacción con el personal médico, los cuales le proporcionarán información de manera amigable y de fácil acceso, dichos programas a base de menús, permiten introducir datos de nuevos pacientes para su registro, fijar condiciones de alarma para todas las variables, monitorear pacientes de manera global (para detección de alarmas principalmente) o de manera individual (graficación de signos vitales), permite también la graficación en papel de los signos vitales. En el capítulo 5 se presentan en detalle los algoritmos para la programación del PCE.

DISEÑO Y CONSTRUCCIÓN DE UN MÓDULO BÁSICO PARA ADQUISICIÓN LOCAL DE SIGNOS VITALES

2.1 .- Introducción

Como se ha mencionado, una de las características fundamentales del sistema diseñado es la modularidad, es decir el equipo se compone de varios módulos, los cuales pueden ser utilizados de manera individual para formar parte de otros sistemas. El diseño del MBM debe ser lo más compacto posible, pero debe ser capaz de realizar el procesamiento necesario para funcionar de manera individual y conectado en red, en caso de ser necesario. Debido a estas características de funcionamiento es necesaria la utilización de un microcontrolador, con un diseño compacto y funcional.

En el presente capítulo se describe detalladamente el diseño elaborado para el MBM, incluyendo también una breve descripción del módulo de detección amplificación y filtrado, el cual como ya se ha mencionado no forma parte de la presente tesis, pero se incluye para visualizar la modularidad del sistema (ver figura 1.1).

2.2.- Módulo de amplificación y filtrado

Este módulo constituye un prototipo que permite la detección de signos vitales por medio de sensores o electrodos conectados directamente al cuerpo. En la figura 2.1 se muestra el diagrama de bloques del módulo de amplificación y filtrado, este módulo está diseñado para obtener el registro de 12 señales o derivaciones de ECG: 3 derivaciones clásicas o de Einthoven, 3 unipolares o aumentadas y 6 precordiales [2],[3], las cuales se obtienen al colocar tres electrodos en las extremidades para las derivaciones clásicas (brazo derecho o RA, brazo izquierdo o LA, pie izquierdo o LF), uno más en la pierna derecha como medio para minimizar ruido e interferencias en las derivaciones unipolares (pie derecho o RF) y el último es un electrodo explorador colocado en 6 diferentes puntos alrededor del corazón para obtener las derivaciones precordiales (torax o C).

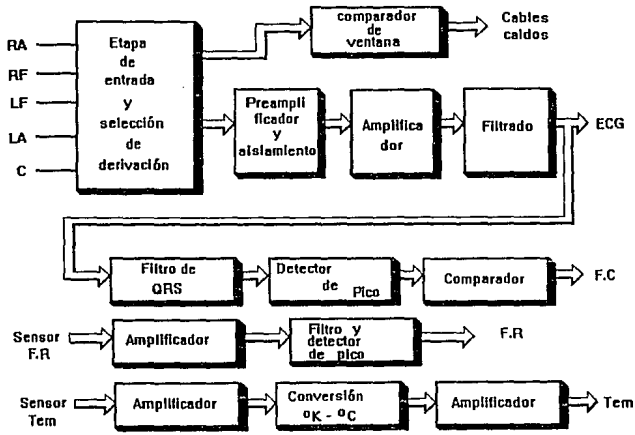


Figura 2.1.- Diagrama de bloques del módulo de amplificación y filtrado

Como se observa en la figura este módulo se subdivide en varios circuitos para detección de ECG, frecuencia cardiaca, temperatura y frecuencia respiratoria, los cuales se describen de manera general a continuación.

Detección de ECG

Esta etapa es fundamental en el diseño del módulo, ya que se encuentra conectada directamente al cuerpo mediante electrodos permitiendo la detección de ECG, por lo que su diseño se sujetó a las normas establecidas por la AAMI para el diseño de monitores de ECG, consta de las siguientes secciones, como se observa en la figura 2.1.

- Circuito de entrada y selección de derivación.- realiza la protección contra descargas provenientes de desfibrilador proporcionando protección de hasta 5000V en 5 milisegundos. Se realiza también la selección de la derivación que se desea monitorear.
- Circuito preamplificador y de aislamiento.- en este punto se realiza la preamplificación de la señal detectada, así como el aislamiento del equipo hacia el paciente, siendo éste uno de los circuitos de protección más importantes.
- Circuito amplificador.- encargado de realizar la amplificación de la señal de ECG, proporcionándole una ganancia fija con la que se obtiene un nivel de señal adecuado para el muestreo y despliegue.
- Filtrado.- en este punto se realiza el filtrado final para eliminar el ruido que pudiera tener la señal, es importante mencionar que en las etapas anteriores se tienen diferentes filtros para eliminar las interferencias existentes (ruido producto del medio ambiente, línea electrica y provocado por el paciente).

Algunas características de esta etapa son:

Ganancia

Preamplificador	21
Amplificador	68
Total	1428

Ancho de banda 0.5 a 35 Hz

Impedancia de entrada

Modo común	mayor a $5x10^{10}\ \Omega$
Modo diferencial	mayor a $10^{8}\ \Omega$

Detección de frecuencia cardiaca

Para su detección se emplea la señal de ECG obtenida en la etapa anterior, siendo ésta filtrada para detectar solamente el complejo QRS [2],[3], utilizando un filtro pasa bajas con frecuencia central en 17 Hz y ancho de banda de 4 Hz. La señal obtenida pasa a un detector de pico y un comparador, teniendo a la salida un pulso cuadrado de 25ms de duración, en cada latido del corazón.

Detección de temperatura

Esta se basa en el empleo de un circuito integrado como sensor que aunado a un circuito amplificador de instrumentación permite obtener la temperatura del cuerpo, el sensor es colocado en un determinado lugar del cuerpo (voca ,recto o axila) para tener una lectura más exacta, el circuito es de fácil calibración, tiene una salida lineal de 10 mV / °C, proporciona compensación interna de frecuencia, tiene un error de 0.1°C y a su salida se obtiene un voltaje proporcional a la temperatura en °C.

Detección de frecuencia respiratoria

Para su detección se aprovecha el cambio de temperatura producto del flujo de aire presente en las fosas nasales. Para detectar este flujo se emplea un elemento sensor, el cual opera como un diodo zener con un voltaje de rompimiento directamente proporcional a la temperatura. Esta etapa consta de detección, amplificación, filtrado y obtención del pulso que caracteriza cada ciclo respiratorio (inhalación y exhalación).

En general el módulo de amplificación y filtrado proporciona las siguientes señales para ser procesadas por el MBM:

- Señal del corazón (ECG) centrada en 2.5 V con amplitud dentro del rango de 0-5 V.
- Señal de temperatura, conste de un voltaje proporcional a ésta en el rango de 0 a 4.5 V.
- Señal de frecuencia cardiaca, proporciona pulsos cuadrados de 25 mseg. de duración y frecuencia correspondiente a la señal de ECG, su amplitud es compatible con los niveles TTL.

- Señal de frecuencia respiratoria, proporciona pulsos cuadrados de 25 mseg. de duración con frecuencia correspondiente a la inhalación y exhalación del paciente, su salida es compatible con TTL.

- Señal de cable caído, proporciona un nivel lógico "1" en caso de detectar un electrodo suelto.

- Señal de tierra, proporciona la referencia de las señales anteriores.

2.3 .- Descripción del microcontrolador MC68HC11

Para la elaboración de la arquitectura del MBM en forma compacta se optó por la utilización de un microcontrolador de la familia de motorola, del cual se hace uso exhaustivo de sus periféricos internos, para realizar las funciones de control del sistema y las tareas de muestreo, almacenamiento y transmisión de datos por puerto serie. A continuación se presenta una descripción general de este microcontrolador.

La unidad microcontroladora MCU (microcontroller unit) MC68HC11E9, es un sistema de alta integración y sofisticados periféricos internos, de alta velocidad de procesamiento y bajo consumo de energía. Sus características internas lo hacen ideal para el proyecto, como se puede observar en el diagrama de bloqes de la figura 2.2; cuenta además con las siguientes características:

- Sistema de contador libre de 16 bits con 4 niveles de preescalamiento.
- Modos de espera para el ahorro de energía.
- Interface de comunicación serie.
- 5 puertos de 8 bits.
- 2-registros acumuladores de 8 bits y uno doble de 16 bits.
- 2 registros indexados de 16 bits.
- 8 kbytes de memoria ROM.
- 512 bytes de memoria RAM.
- 512 bytes de memoria EEPROM.
- Convertidor analógico-digital de 8 canales de 8 bits.
- 18 fuentes de interrupción

Modos de operación

El MCU utiliza dos pines (MOD A y MOD B) [4], para seleccionar uno de cuatro modos de operación, dos de ellos llamados básicos (*simple chip y expandido*) y dos especiales (*bootstrap y prueba*).

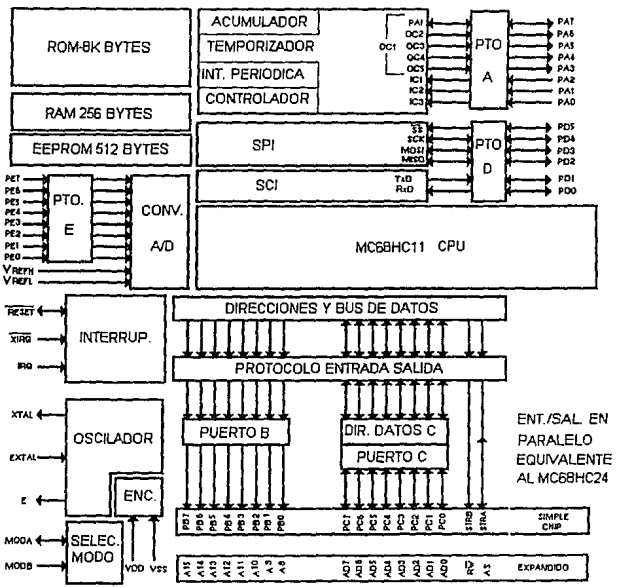


Figura 2.2 Estructura interna del MCU 68HC11.

Simple chip.- en este modo de operación, el MCU hace uso de sus recursos internos y no tiene *buses* externos, por lo que se aprovechan al máximo sus puertos de entrada-salida, pero se tiene la restricción de la memoria interna (512 bytes).

Multiplexado-expandido.- en este modo el MCU puede direccionar hasta 64 Kbytes para memoria externa o puertos adicionales. La parte alta del *bus* de direcciones son las salidas del puerto B, la parte baja del *bus* se encuentra multiplexada con el *bus* de datos y son tomadas del puerto C, la línea "AS", se utiliza como línea de control para el demultiplexaje del *bus* de datos y las direcciones; en este modo se reduce el numero de puertos de entrada-salida, se cuenta además con líneas de control para el acceso a memoria o periféricos externos (AS,R/W,E).

Modo bootstrap.- este es un modo de operación muy versátil, ya que combina los modos anteriores, sólo que cambia la posición de los vectores de interrupción; puede ser usado para programar la EEPROM interna, para probar el programa residente en ROM y puede ser cambiado a otros modos bajo el control del programa.

Modo prueba.- éste solo se utiliza para calibrar los sistemas que componen al MCU y solamente es usado por el fabricante.

Puertos de entrada - salida

Puerto A .- está relacionado con el temporizador, puede ser configurado en todos los modos de operación con tres entradas para captura de datos, cuatro salidas para funciones de comparación y una entrada (PA1) para el acumulador de pulsos.

Puerto B .- en modo *simple chip*, todas las terminales son usados como salidas de propósito general, en modo multiplexado expandido se utiliza para el direccionamiento externo.

Puerto C .- en modo *simple chip* se utiliza como un puerto de propósito general de entrada - salida, en modo expandido está multiplexado con el *bus* de datos y la parte baja de las direcciones, sus entradas pueden tener un registro *latch* controlado por la línea de control "STRA".

Puerto D .- es un puerto bidireccional de propósito general y puede ser configurado como dos subsistemas de comunicación serie, sincrono (PD2 - PD5) y asincrono (PD0 - PD1); pueden seleccionarse diferentes baudajes y sus salidas son TTL.

Puerto E .- puede ser usado como puerto de entrada de propósito general o como entradas para el convertidor analógico - digital, hasta 8 canales en los modelos PLCC (Plastic Leades Chip Carrier).

A lo largo del escrito se describirán con más detalle algunos de los periféricos empleados de este MCU.

2.4 .- Arquitectura diseñada

Una de las tareas del equipo es el almacenamiento de muestras de la señal de ECG , para su posterior reconstrucción, por tal motivo se requiere del uso de una memoria RAM externa, ya que la RAM interna del MCU no es lo suficientemente grande para almacenar las muestras. Para el uso de memoria externa es necesario utilizar al MCU en modo *multiplexado expandido*, con el cual se tiene la posibilidad de direccionar mas memoria o puertos de entrada/salida.

Como podemos observar en el diagrama de bloques de la figura 2.3, el sistema cuenta con memoria RAM , EPROM (para el almacenamiento del programa de aplicación), un decodificador, un *latch*, el modem FSK y el MCU MC68HC11E9, por lo que tenemos un sistema digital compacto, a excepción del modem FSK, pero éste se puede manejar como un bloque independiente, ya que solamente se utilizará para la transmisión serial através de la red eléctrica hacia el PCE.

En modo *multiplexado expandido* como ya se ha mencionado el *bus* de datos se encuentra multiplexado con la parte baja de las direcciones, para realizar el demultiplexaje se utiliza un circuito tipo *latch* cuya habilitación proviene de la línea de control del MCU "AS" [4], como se observa en la figura 2.3. La decodificación utiliza medio ciclo de la señal de reloj "E" para el *bus* de datos y el otro medio ciclo junto con la señal de "AS" para el *bus* de direcciones. Teniendo demultiplexados ambos *buses* podemos plantear el mapa de memoria para la arquitectura. El mapeo de memoria se

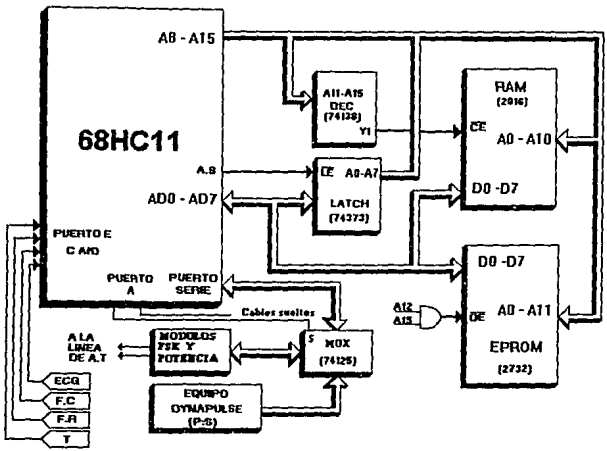


Figura 2.3 Diagrama de bloques de la arquitectura diseñada.

realizó dividiendo el espacio en bloques de 4 Kbytes. En la figura 2.4 se presenta este mapa, la parte sombreada corresponde a las localidades ocupadas por la memoria, registros internos y vectores de interrupción, el resto del espacio puede ser utilizado para direccionar más memoria o puertos de entrada - salida. En la parte alta se encuentran direccionados los vectores de interrupción, el vector de reset, que es el que nos interesa para inicializar el sistema, se encuentra en la dirección $FFFE, y es en esta localidad donde se debe guardar la dirección de inicio del programa de aplicación; este vector de interrupción, al igual que el programa de aplicación serán grabados en una EPROM externa de 4 Kbytes, esta memoria debe ser direccionada en la parte alta del mapa de memoria, es decir debe de estar a partir de la dirección $F000 a la $FFFF; para su direccionamiento utilizamos la parte alta de las direcciones, con las cuales habilitaremos la memoria, es decir las líneas de dirección A_{12} - A_{15} se conectan a una compuerta NAND, cuya salida se conecta al "OE" de la EPROM; con este arreglo nos

aseguramos de que la memoria sólo se activará cuando se tenga una dirección comprendida entre $F000 y $FFFF. La RAM se localiza en el primer bloque de memoria, para su direccionamiento se utiliza un circuito decodificador (74HC138), con el fin de poder direccionar más memoria en caso de ser necesario, para ello utilizamos las líneas de dirección A_{11} - A_{15} y la señal de control "E", con ello direccionamos la RAM a partir de la dirección $0800 a la $0FFF (2 Kbytes); como se observa en la figura 2.4.

En el diagrama de la figura 2.5 se muestran los siguientes aspectos: la circuiteria necesaria para el funcionamiento del MCU en modo *expandido*; el circuito de reset, que pone un nivel bajo en el pin de "RESET" para inicializar al MCU, eliminando los rebotes; el circuito del cristal de 8 Mhz para proporcionarle al MCU el reloj, y que a su vez éste genere la señal de reloj "E" de 2 Mhz; el circuito de decodificación de memorias, descrito anteriormente; el circuito de acondicionamiento entre los niveles TTL del MCU y los niveles RS232 de la computadora, formado por el circuito MAX232, con el que se realiza la comunicación serial entre el MBM y el PCE; tambien se muestran los niveles de referencia para el convertidor A/D (VRH y VRL), los cuales fueron fijados para tener una ventana de conversión de 0 a 5 Volts, que es el rango de variación de las señales vitales proporcionadas por el módulo analógico descrito anteriormente. Las señales previamente amplificadas y filtradas de ECG, FC, FR y T se conectan a los primeros cuatro canales del convertidor A/D del puerto E; por el puerto A se introduce la señal de alarma de cables caídos en el paciente; el puerto serie del MCU (PD0, PD1) se utiliza para la comunicación con el PCE y al mismo tiempo para recibir la señal de PS proveniente de un equipo comercial DYNAPULSE (del cual se darán posteriormente sus características), por ello es necesario multiplexarlo por medio del circuito 74HC125, para el cual la señal de control de tercer estado es provista por el puerto A.

Con este diseño se logra que el MBM sea muy compacto y funcional, siendo una parte muy importante para su funcionamiento la correcta programación del MCU, ya que él será el encargado de realizar el control y procesamiento de las señales. En los siguientes capítulos se detallarán los programas elaborados para el MBM.

En la siguiente lista se enumeran algunas características del MBM.

1.- Rango de conversión de señales de 0 a 5 V.
2.- Tiempo de conversión 16 microsegundos.
3.- Memoria RAM de 2 Kbytes.
4.- Memoria EPROM de 4 Kbytes.
5.- Velocidad de transmisión de 9600 bauds.

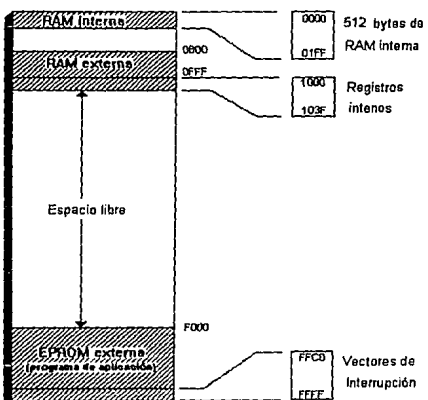


Figura 2.4 Mapa de memoria del MBM.

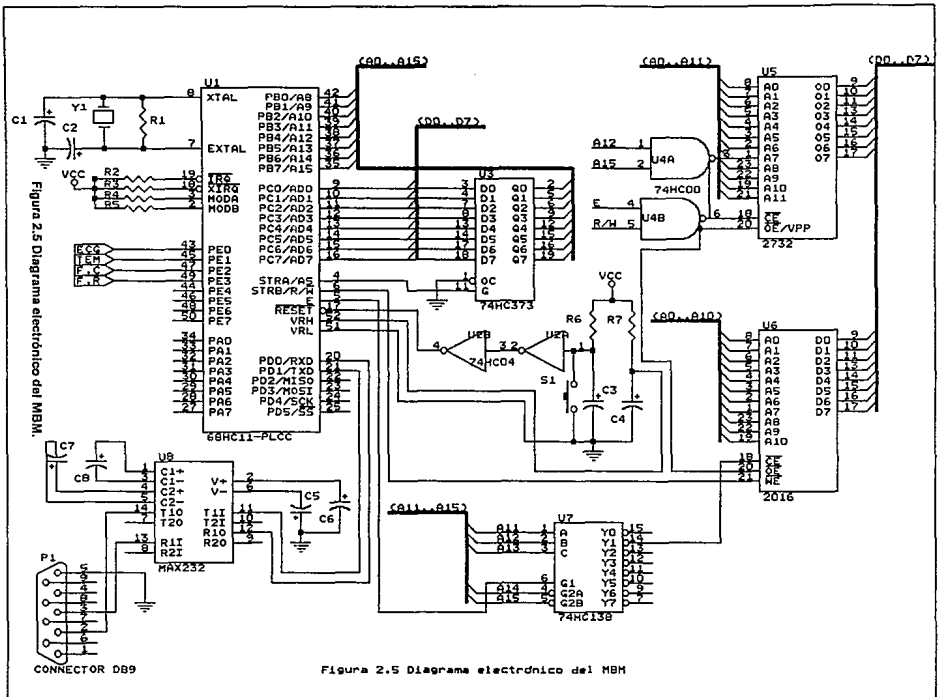


Figura 2.5 Diagrama electrónico del MBM.

Figura 2.5 Diagrama electrónico del MBM

MUESTREO DESPLIEGUE Y TRANSMISION DE DATOS

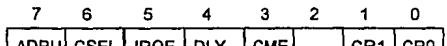
3.1 .- Introducción

La función primordial del MBM es el muestreo y el tratamiento de los datos de la señal de ECG y signos vitales, el muestreo debe ser seleccionado de tal manera que podamos reconstruir la señal de ECG lo más exacto posible a la señal original, para ello, una de las herramientas de apoyo para la selección de la frecuencia de adquisición lo constituye el teorema de muestreo [5],[6], con el cual podemos calcular la frecuencia mínima para reconstruir la señal, sin embargo, el periodo de muestreo para una señal de ECG se encuentra plenamente definido en la bibliografía[3],[6], es decir un periodo de muestreo de 256 Hz es suficiente para la reconstrucción de la señal con un error relativo de 1.6% a una frecuencia máxima de 250 pulsos por minuto, como se verá en el presente capítulo.

En este capítulo tambien se describen algunos análisis realizados para el despliegue local, utilizando un monitor convencional de TV, así como la posible utilización de un display de cristal liquido; por el momento el despliegue se realiza en una computadora central. Se describe también el diseño preliminar de un sistema de transmisión de datos através de la red eléctrica de baja tensión, el cual no forma parte de la presente tesis, sin embargo debido a la participación en su diseño se incluye una breve descripción.

3.2 .- Muestreo de signos vitales

Como se mencionó en el apartado anterior, através del convertidor A/D integrado al MCU se muestrean 4 canales para adquirir muestras de ECG, FC, FR y T. Este convertidor de 8 canales de 8 bits y retención de muestra, utiliza la técnica de conversión de aproximaciones sucesivas con redistribución de cargas [4], este tipo de convertidor es el más popular en aplicaciones biomédicas ya que combina velocidad y simplicidad de conversión [6], tiene un error de cuantización de ±1/2 LSB, con una velocidad de conversión de 16 microseg. Para su utilización es necesario configurar los siguientes registros: OPTIONS Y ADCTL (registro de control del convertidor A/D).



ADPU .- A/D powerup, proporciona la energía necesaria para activar el convertidor.

0 = desactivado

1 = activado

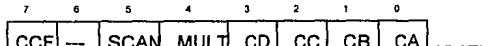
CSEL .- clockselect, selecciona el oscilador de referencia para la conversión.

0 = 2 MHz

1 = 750 kHz

Los bits restantes no son usados en el control del convertidor A/D.

Como queremos trabajar a su máxima capacidad seleccionamos el oscilador de 2 MHz y activamos la energía para su funcionamiento, por lo que debemos configurar este registro con un $80.



CCF.- bandera de conversión completa, indica la terminación de una conversión.

0= realizando conversión.

1 = conversión completa.

SCAN.- control de conversión contínua, selecciona el tipo de conversión a realizar.

0 = conversión contínua.

1 = una sola conversión.

MULT .- control de canal multiple, selecciona la conversión de uno o cuatro canales.

0 = 1 canal.

1 = 4 canales.

CD, CC, CB, CA.- seleccionan el canal a convertir.

Cada vez que necesitemos los datos de una conversión se debe escribir este registro, despues de 16 microsegundos los resultados de la conversión de los cuatro canales se tendrán disponibles en los registros ADR1-ADR4 respectivamente, debemos realizar una conversión de 4 canales a la vez cada 3 milisegundos, seleccionando los primeros cuatro canales, por lo que debemos configurar este registro con un $10.

En la figura 3.1 se muestra un periodo típico de una señal de ECG, la cual puede variar dependiendo de las características de salud y edad de la persona, pero que normalmente presenta las características mostradas. De esta señal se pretenden visualizar 4 periodos a la vez; si tomamos una muestra cada 2.4 milisegundos (256 muestras/periodo) tendremos un periodo de muestreo de 427 Hz valor que se encuentra por encima de la frecuencia de muestreo utilizada entre otros por L. Edenbrandt et al. [7] y Wendel C. Ocasio et al. [8] (256 Hz) y que además cumple con las normas AAMI (Association for the Advancement of Medical Instrumentation) para monitoreo cardiaco [9]. La frecuencia de muestreo usada se puede r variar para tener diferentes resoluciones, generalmente se usa una frecuencia de 256 Hz con la que se tiene un error teórico de 1.6 %, como muestra la ecuación 3.1; esta frecuencia es un valor adecuado para un sistema digital, ya que generalmente la señal se despliega a 256 Hz, sin embargo, para facilitar el procesamiento de las señales se utilizó una frecuencia de 333 Hz, es decir, se toma una muestra cada 3 milisegundos, a esta frecuencia se adquieren las señales vitales del paciente.

$$error = \frac{Frec.max}{Frec.muestreo} \times 100 = \frac{250ppm}{256m/seg} = 1.6\% \qquad (3.1)$$

Para la señal ECG se toma una muesta cada 3 milisegundos y se almacena en RAM hasta alcanzar un total de 1024 muestras (el equivalente a 3 segundos aproximadamente de la señal), de donde serán leidos los datos por el sistema de despliegue para su posterior reconstrucción, en nuestro caso la muestra será enviada al PCE para su graficación si éste así lo requiere. En el caso de la temperatura el valor obtenido con el convertidor A/D se multiplica por un factor, para transformar los datos a grados centígrados y su valor se almacena en RAM para ser enviado al PCE cuando éste lo solicite. El factor de conversión está determinado por la ventana de conversión y la resolución del convertidor (8 bits), por lo que tendremos 256 posibles valores provenientes del convertidor y cada valor corresponderá a un voltaje de 5 V/256 = 0.0195312 V, siendo éste el factor de conversión para la temperatura.

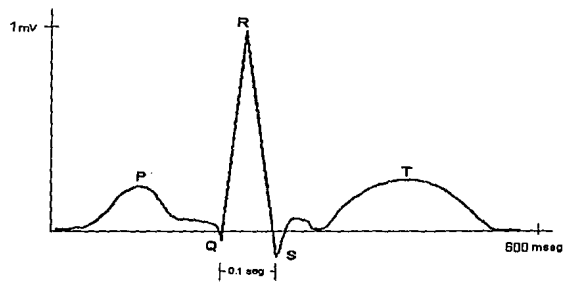


Figura 3.1 Señal típica de ECG

Las señales de FC y FR, están formadas por pulsos a los cuales se les tiene que calcular la frecuencia en pulsos por minuto, para calcularla se necesita conocer la duración de la parte baja del pulso, ya que la parte alta es de duración constante, bajo estas condiciones el MCU calcula la frecuencia en pulsos por minuto de acuerdo al diagrama de flujo de la figura 3.2, utilizando la técnica de "beat a beat", es decir se calcula la frecuencia en cada periodo de la señal; posteriormente se incluirá la técnica

de promediación en la que se calcula la frecuencia tomando el promedio de varios periodos de la señal.

El MCU toma la muestra y evalúa si se trata de la parte alta o baja del pulso, mientras sea la parte baja incrementa un contador, si la muestra corresponde a la parte alta se realiza el cálculo de la frecuencia, multiplicando el valor del contador por el periodo de muestreo y se le suma el tiempo del pulso alto, teniendo así el periodo total de la señal, a este valor se le calcula la frecuencia en pulsos por minuto, y su valor es almacenado en RAM. En las siguientes muestras de la parte alta se inicializa el contador para calcular la frecuencia del siguiente pulso. De acuerdo a los estandares de las normas AAMI, todo equipo para monitoreo cardiaco debe ser capaz de detectar frecuencias comprendidas de 20 a 250 pulsos por minuto para la frecuencia cardiaca y de 8 a 150 pulsos por minuto para frecuencia respiratoria, ambas con un error máximo de ±5 pulsos por minuto. En nuestro caso el cálculo de las frecuencias se realiza dividiendo una constante por el numero del contador, como se observa en la ecuación 3.2. Realizar esta división de 16 bits en el MCU involucra un error inherente en el cálculo de las frecuencias de ±3 pulsos por minuto, error que es aceptable de acuerdo a las normas AAMI.

$$FC = \frac{60}{\#\ contador * 3mseg} = \frac{20000}{\#\ contador} \qquad (3.2)$$

El valor de las frecuencias utilizando una frecuencia de muestreo de 256 Hz, tiene un error de 1.6 % cuando se detecta una frecuencia de 250 ppm como se explicó anteriormente, lo que corresponde a un error de ± 4ppm, error que es aceptable de acuerdo a normas AAMI, sin embargo, como estamos utilizando una frecuencia de muestreo de 333Hz este error se reduce a ± 3ppm en la detección de frecuencias de 250ppm, reduciendose el error en la detección de frecuencias más bajas, llegando a ser de ± 0.25ppm en frecuencias de 20ppm.

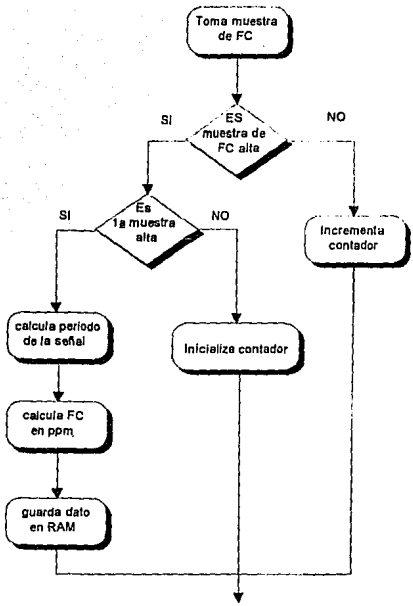


fIGURA 3.2 Diagrama de flujo para el cálculo de frecuencia cardiaca y respiratoria.

Para medir la presión sanguínea se utilizará un módulo comercial "DYNAPULSE" [10], de tipo no invasivo, el cual no requiere calibración desde el punto de vista del usuario. Este módulo utiliza el método oscilométrico para medir la presión sanguínea y entrega los datos através del puerto serie. Por el momento la compra de este equipo se encuentra en proceso, por lo que no se realiza adquisición de presión sanguínea.

Para que el sistema funcione en tiempo real el MCU debe realizar las tareas de muestreo, procesamiento y transmisión serial, además de la detección de electrodos sueltos en el paciente (leyendo el estado del puerto A) en un periodo menor a 3 milisegundos, periodo que es suficiente si el MCU opera a 2 Mhz. Para lograr que el MCU realice las rutinas en un periodo de 3 milisegundos se tomaron en cuenta los ciclos de maquina de cada instrucción, de tal manera que a cada rutina se le añaden retardos para ajustar el tiempo de ejecusión de la rutina a 3 milisegundos.

Debe subrayarse que tanto el MBM como el PCE son de tipo pasivo, por lo cual sólo monitorean y grafican los signos vitales, por lo que no se aplican algoritmos digitales para el análisis o reconocimiento de señales, eventualmente este tipo de registro dinámico [11] se podría integrar en el futuro.

3.3 .- Despliegue de signos vitales

El desplegado local de signos vitales podría realizarse mediante equipos especiales como osciloscopios con memoria o polígrafos, pero resultaría un equipo sumamente costoso; por esta razón presentamos el análisis de algunas opciones para el desplegado local, las cuales se encuentran en proceso de estudio para seleccionar la más adecuada, por el momento el despliegue se realiza mediante la computadora en el PCE.

Una de las opciones es la utilización de un monitor convencional de TV modificado [12], en donde podemos dibujar la señal de ECG en el tubo de rayos catódicos (TRC) controlando las placas deflectoras X-Y, en cuyo caso la señal de ECG sería la señal de contraste para el barrido vertical, para ello está señal debe ser compatible con la norma NTSC (National Television System Comittes) [13], en la cual se tienen 15750 Hz de barrido horizontal y 60 Hz de barrido vertical. Para visualizar los 4 periodos de ECG se tendrían que agrupar (4)(256 muestras/periodo) = 1024 muestras que deben ser procesadas en 1/60 de segundo es decir 61440 muestras por segundo, frecuencia muy elevada en comparación con la frecuencia de muestreo del MCU, bajo estas circunstancias es necesario diseñar un autómata de despliegue que lea los datos de la RAM a la frecuencia televisiva de 61440 muestras por segundo y

enviarlas a un convertidor digital - analógico para formar la señal de contraste. Para la visualización de los valores de frecuencias, temperatura, presión sanguínea y demás parámetros es necesaria la utilización de un generador de caracteres; con todas estas características el diseño del sistema se vuelve más robusto y más caro, sin embargo se estudia además la posibilidad de incorporar un display de cristal liquido controlado por puerto serie, esto haría al sistema muy compacto y ligero, pero el costo también es elevado, por ello se encuentra en estudio la opción para el despliegue local.

3.4 .- Transmisión de datos

Para la transmisión de los datos através de la línea de voltaje, es necesario utilizar algún tipo de modulación, ya que los datos en formato RS-232 no pueden ser transmitidos directamente, debido a las bajas frecuencias que utiliza; es por esto que el diseño del módulo se basa en la técnica de modulación digital de onda continua denominada llaveo o manipulación por corrimiento de frecuencia FSK (frecuency - shift - keying) [14]. En esta técnica la frecuencia de la señal moduladora o frecuencia de transmisión se desplaza entre dos frecuencias a medida que se transmite un "1" o un "0" lógicos.

En la figura 3.2 se muestra el diagrama de bloques para el módulo FSK el cual está formado por los siguientes bloques:

1.- El transmisor formado principalmente por un circuito integrado generador de ondas senoidales, del cual obtenemos las dos frecuencias moduladoras correspondientes a un "1" y "0" lógicos, dichas frecuencias se obtienen utilizando al generador como un modulador de FM, para ello se utiliza un circuito que acondiciona los voltajes RS-232 y los hace compatibles con el generador, teniendo un voltaje para el "1" lógico con su correspondiente frecuencia a la salida del generador y otro para un "0" lógico con su correspondiente frecuencia, ambas frecuencias moduladoras pasan através de un amplificador de potencia para ser transmitidas por la línea de tensión.

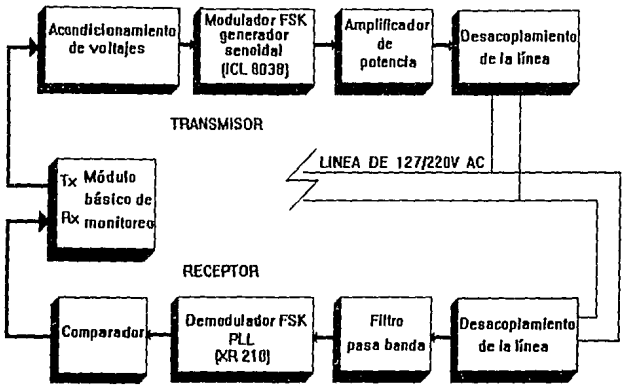


Figura 3.3 Diagrama de bloques del módulo FSK

2.- Etapa de acoplamiento con la línea, formada principalmente por capacitores. Para tener un acoplamiento con pérdidas mínimas los capacitores deben ser seleccionados de tal manera que presenten una alta impedancia a la energía de 60 Hz y un equivalente serie resistivo de bajo valor para el rango de frecuencias moduladoras utilizadas por el transmisor, las cuales deben estar dentro del ancho de banda de la línea de tensión (30 - 500 KHz) [15], además los capacitores deben soportar un voltaje mayor al voltaje pico a pico de operación de la línea. Para la selección de los capacitores se utiliza la ecuación que representa la impedancia de un capacitor, con la

cual lo seleccionamos de acuerdo a la frecuencia de operación del transmisor y a la impedancia de acoplamiento.

$$Zc = \frac{1}{2\Pi fc}$$

f = frecuencia de operación
c = capacitancia
Zc = impedancia del capacitor

3.- El receptor, formado principalmente por un filtro pasabanda, sintonizado a las frecuencias de operación del transmisor, con el cual se elimina una gran parte de las interferencias de baja frecuencia presentes en la línea; las señales filtradas pasan a un circuito demodulador de FM formado por un PLL (phase - locked loop), el cual opera con una excelente linealidad entre las frecuencias de entrada y el voltaje de salida, teniendo así dos niveles de voltaje correspondientes a las dos frecuencias de operación del transmisor, finalmente estos voltajes pasan a un comparador de voltaje para acondicionar la salida del PLL a un formato RS-232.

Por el momento el diseño de este módulo se encuentra en proceso de pruebas y validación.

INTERACCIÓN ENTRE EL MÓDULO BÁSICO DE MONITOREO Y EL PUESTO CENTRAL DE ENFERMERAS

4.1 .- Introducción

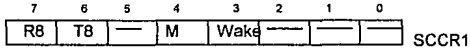
Una de las características funcionales del MBM es su posible uso en red, es decir monitorear de manera local y al mismo tiempo estar conectado a la computadora central del PCE, o como en nuestro caso, estar solamente conectado al PCE. Como éste puede monitorear hasta 14 MBM's es necesario utilizar algún identificador para cada uno, así como un protocolo para realizar la transferencia de datos. El identificador o clave de cada MBM se fija mediante *software*, de esta manera cuando se encuentren conectados más de un MBM al PCE y éste solicite datos de alguno de ellos todos los MBM conectados recibirán la clave pero solo responderá uno de ellos. En cuanto a la transmisión de información entre ambos módulos es necesario establecer un protocolo de comunicación para tener una sincronía entre ellos, por lo que es necesario elaborar algoritmos para ambos módulos y que trabajen en tiempo real, para ello se utilizó un sistema de comunicación serial eficiente como es el caso del SCI (serial comunication interface) integrado en el MCU. El medio de comunicación entre ambos sistemas puede ser la conexión directa o enlazados mediante el módem FSK en caso de encontrarse a una distancia larga, en ambos casos realizando una comunicación half duplex. En el presente capítulo se describe el sistema de transmisión serial de datos del MBM hacia el PCE, así como el protocolo establecido para realizar el intercambio de información.

4.2 .- Transmisión serial del MBM

La interacción entre el MBM y el PCE se realiza mediante la transmisión de datos através de puerto serie, utilizando para ello el SCI integrado al MCU, este puerto serie es un sistema asíncrono full - duplex tipo UART (universal asynchronous receiver / transmiter) [16], el cual permite una excelente comunicación con dispositivos periféricos, en nuestro caso se realizó una comunicación tipo half-duplex con el PCE, debido a las características de la programación. El SCI puede utilizar una gran variedad de velocidades de transmisión, las cuales dependen del cristal utilizado en la arquitectura, para la transmisión utiliza el estandar NRZ (no retorno a cero) con formato de 1 bit de inicio, 8 o 9 bits de datos y 1 bit de paro.

Para la correcta utilización del SCI es necesario configurar 5 registros del MCU: SCCR1 (registro de control 1 del SCI), SCCR2 (registro de control 2 del SCI), BAUD (registro de control de baudaje), SCSR (registro de status del SCI) y SCDR (registro de datos del SCI); los tres primeros se utilizan para inicializar el puerto serie y los dos últimos para la transmisión y recepción.

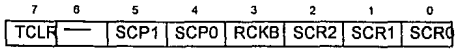
En la parte de inicialización debemos fijar el formato de los datos a transmitir, con el fin de hacerlos compatibles con el formato que espera recibir la computadora del PCE, para ello utilizamos el registro SCCR1 en el cual se tienen los siguientes bits:



R8 - activa la recepción del bit 8 en caso de formato con 9 bits de datos
T8 - activa la transmisión del bit 8
M - selecciona el formato de los datos
0 = 1 bit de inicio, 8 bits de datos, 1 bit de paro
1 = 1 bit de inicio, 9 bits de datos, 1 bit de paro
Wake - selecciona el método de activación de línea
0 = línea desocupada (idle line)
1 = marca de dirección (detecta el bit 8 o 9 para activar la línea)

Para el presente trabajo se seleccionó el formato de 8 bits de datos y el método de línea desocupada, es decir la línea permanecerá en estado alto mientras no haya datos a transmitir, por lo que debemos configurar a este registro con un $00.

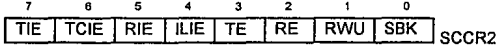
Para fijar el baudaje al cual se transmitirá, se debe configurar el registro BAUD, en el cual se tienen los siguientes bits:



SCP0 - SCP1 - seleccionan el factor para preescalar la frecuencia del cristal
SCR0 - SCR2 - seleccionan el baudaje deseado

Los bits restantes sólo son usados para modo *prueba*. Utilizando un baudaje de 9600 bauds este registro se debe configurar con un $30. Para utilizar baudajes diferentes sólo hay que cambiar la configuración de este registro, teniendo en cuenta que para transmisiones en tiempo real se requiere de una velocidad de transmisión de 7500 bauds (250 muestras / seg / canal , enviando 10 bits / muestra) [6].

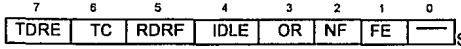
Para que el SCI funcione como un sistema full-duplex se utiliza el registro SCCR2 en el cual se tienen los siguientes bits:



TE - habilita transmisión
0 = deshabilitada
1 = habilitada
RE - habilita recepción
0 = deshabilitada
1 = habilitada

En este registro únicamente se habilitan los bits de transmisión y recepción escribiendo un $0C, los bits restantes se utilizan para interrupciones en transmisión y recepción.

En la transmisión de datos se usan dos registros, el SCDR para escribir el dato a transmitir y el SCSR, en el que se leerá el estado del puerto, en este último registro se tienen los siguientes bits:



TDRE - registro de transmisión de datos lleno
0 = no lleno
1 = lleno
TC - transmisión completa
1 = indica que ha terminado la transmisión
RDRF - registro de recepción de datos lleno
0 = no lleno
1 = lleno

En la recepción primero se lee el registro SCSR para averiguar si el bit 5 se encuentra activado, lo que nos indica que llego un dato, posteriormente se lee el dato del registro SCDR. El resto de los bits se utilizan para saber si ocurrió alguna sobreescritura o algún problema con los datos.

En el apéndice A se muestra el listado en ensamblador de la programación realizada.

4.3 .- Protocolos de comunicación y programación para la interacción entre MBM y PCE

Dado que el sistema de comunicación serie es asíncrono y el proyecto requiere de una cierta sincronía para establecer la comunicación duplex entre el MBM y el PCE, fue necesario programar ambos sistemas con un protocolo determinado para realizar el intercambio de información en tiempo real. Este protocolo se diseñó con una topología especial para el proyecto, sin embargo la modularidad del programa permite modificarlo para adecuarlo a otras necesidades.

El MBM realiza las tareas de conversión, procesamiento y despliegue local, y al mismo tiempo está pendiente de la comunicación con el PCE, la comunicación para

transferencia de datos empieza con el protocolo. En la figura 4.1 se muestra el diagrama de flujo de la programación del MBM, en el cual se incluye el protocolo.

El protocolo se inicia cuando el PCE envía, vía puerto serie, la clave del o de los MBM's que desea supervisar. El programa del MBM detecta la clave que le llega y la compara con la clave que tiene asignada , si no son iguales continúa con el monitoreo y procesamiento de una muestra de signos vitales, generando un retardo 1 para esperar a que pasen 3 milisegundos, pero si corresponde a su clave, envía una contraseña (ACK), la cual le indica al PCE que ése MBM se encuentra activo, posteriormente el PCE le envía el identificador para inicio de transmisión y el MBM envía la contraseña de dato recibido, en caso de que el MBM reciba un dato que no es inicio de transmisión, éste continua con el procesamiento de otra muestra y genera un retardo 2 para ajustar esta rutina a 3 milisegundos; al recibir el inicio de transmisión el MBM se encuentra listo para enviar datos al PCE, como éste tiene que atender a todos los MBM conectados a la red, el tiempo transcurrido entre el envío de cada petición de dato, puede ser mayor a 3 milisegundos, por lo que el programa del MBM tiene que continuar con su labor de muestreo y procesamiento, como se muestra en el diagrama de flujo de la figura 4.1, en donde cada rutina tiene que ajustarse a una duración de 3 milisegundos. En la misma figura se observa que para la transmisión de datos el PCE envía un "CHK" y el MBM un dato de ECG, continuando con este proceso hasta completar una pantalla, es decir la visualización de 4 periodos de ECG en un monitor VGA, lo que corresponde a enviar un dato cada 4 milisegundos (el equivalente a 256 muestras por segundo) por lo que el MBM envia una muestra al PCE por cada 2 muestras de ECG que procesa. Una vez concluido el envío de datos de ECG, el PCE envía un identificador (PARAM) para que le sean enviados los datos de las otras variables, al recibir este identificador el MBM lee de la RAM el dato de cada variable, enviándolo al PCE, continuando con este proceso mientras dure el monitoreo. Cuando termina se envía el identificador de fin de transmisión; al recibirlo el MBM, éste continua monitoreando y procesando los signos vitales, pero se encuentra en espera del protocolo para nuevamente interactuar con el PCE.

La programación del MBM se realizó en lenguaje ensamblador, su diseño es modular para permitir la inserción de nuevas rutinas que permitan realizar el despliegue local, así como la atención al módulo de presión sanguínea. En el apéndice A se incluye un listado completo del programa utilizado para el MBM, el cual se realizo siguiendo el diagrama de flujo de la figura 4.1.

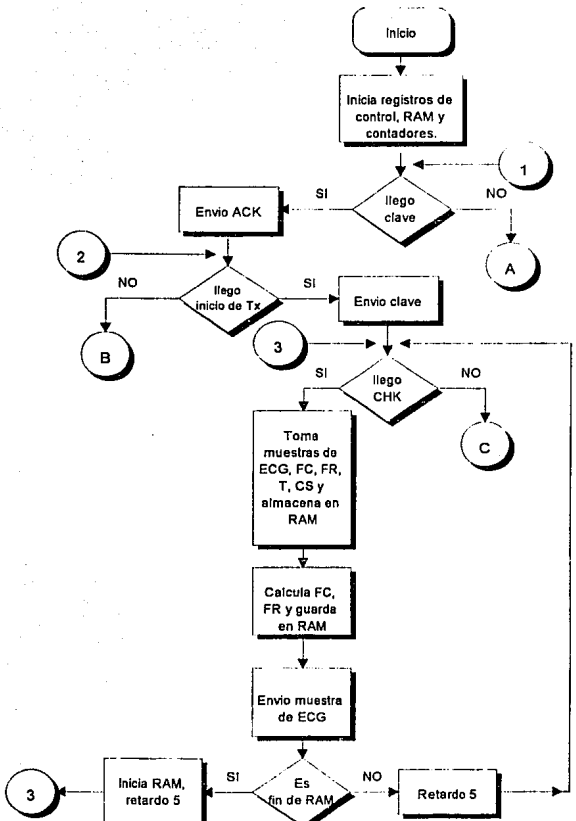


Figura 4.1 Diagrama de bloques de la programación del MBM

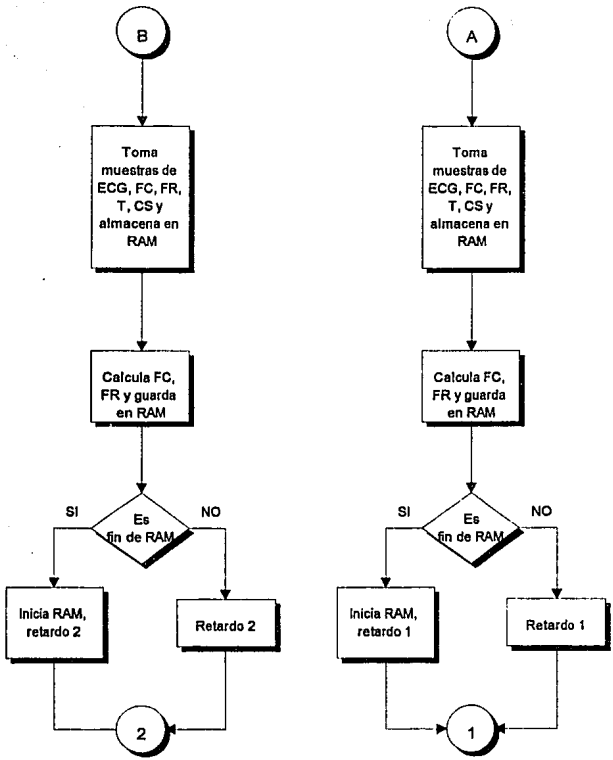


Figura 4.1.-Continuación.

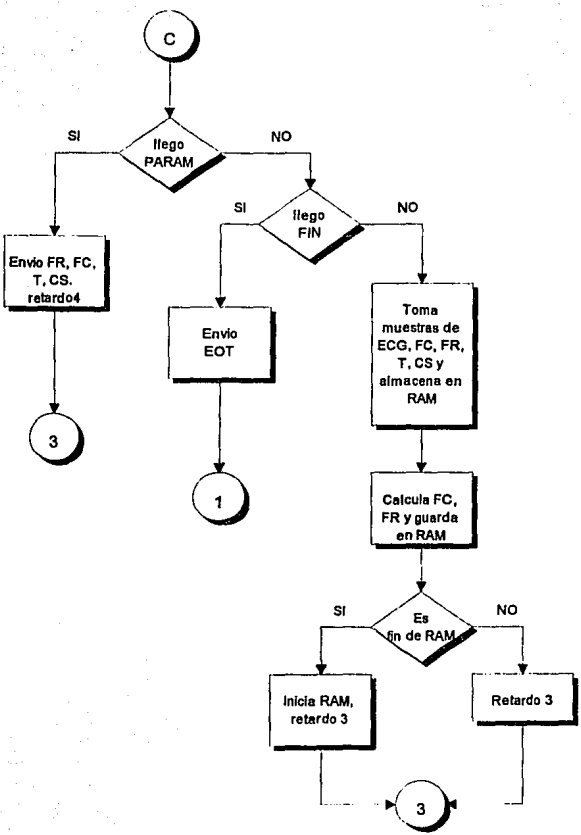


Figura 4.1.- Continuación.

PROGRAMACIÓN INTERACTIVA ELABORADA PARA EL PCE

5.1.- Introducción

El uso de una computadora para el despliegue y adquisición de datos de signos vitales en unidades de cuidados intensivos o en situaciones clínicas similares, es una contribución muy significativa, ya que con ella se cumplen satisfactoriamente los requerimientos de las normas industriales de fabricación de monitores de signos vitales [9], algunos de estos requerimientos (para el despliegue) son los siguientes:

- Los métodos de despliegue deben ser capaces de presentar datos válidos en el tiempo, es decir datos en tiempo real.
- Tener la capacidad de retener la imagen para que pueda ser visualizada.
- Tener una sección de presentación de alarmas.
- Desplegar a la vez caracteres alfanuméricos y gráficas.
- Poder introducir fácilmente datos que interaccionen con el despliegue.

Tomando en cuenta estas características se presenta un programa prototipo de supervisión y administración de la red para el PCE. El programa debe realizar una serie de tareas en las que se incluyen: la interacción con el personal médico, manejo de la información de los pacientes proporcionada por el usuario, y la interacción con cada MBM conectado a la red. Estas tareas deben ser transparentes para el usuario, por ello la programación se elaboró a base de menús en los que el usuario solamente debe elegir la opción que desee realizar y proporcionar la información que se le pide. El programa se elaboró de manera modular para facilitar su mantenimiento y poder integrar nuevas rutinas que mejoren el sistema, fue escrito totalmente en lenguaje C , ya que éste cuenta con las características necesarias para hacer el programa funcional y transportable [17].

Como se ha mencionado el programa se divide en dos partes fundamentales, interacción con el personal medico e interacción con cada MBM, la primera parte se encarga de administrar una pequeña base de datos para almacenar información general de los pacientes, así como de regular las condiciones de encendido de alarmas, esta parte del programa se presenta a base de menús elaborados con el auxilio de la utileria de C denominada "Vlib" [18]. La programación para la interacción con cada MBM incluye el protocolo de comunicaciones descrito en el capitulo anterior, incluye también la parte de ayudas gráficas, con las que se grafica la señal de ECG y los demás signos vitales, cuenta con la activación de alarmas visuales y auditivas para la detección de signos vitales anómalos en el paciente. En los siguientes párrafos se describen detalladamente ambas secciones del programa, mientras que en el apéndice B se muestra un listado completo de la programación elaborada.

5.2 .- Programación interactiva con el personal médico

La programación modular nos permite agregar rutinas que permitan modificar el funcionamiento básico del sistema, por ello en esta parte de la programación se elaboraron varias rutinas que permiten trabajar con la base de datos e interactuar con el personal médico, estas rutinas son las opciones del menú principal, como podemos observar en el diagrama de flujo de la figura 5.1, las cuales se describen a continuación.

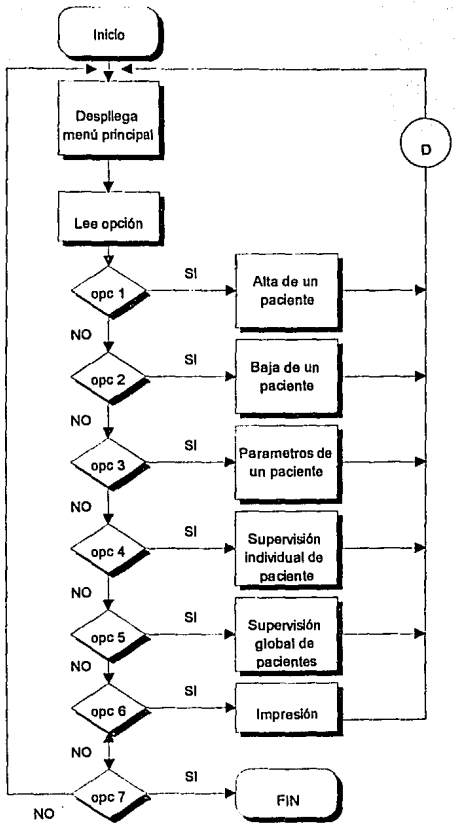


Figura 5.1 Diagrama de flujo para la programación interactiva con el personal medico

Alta de un paciente.- esta rutina actualiza la base de datos de un nuevo paciente, al seleccionarla aparece en pantalla un formato que el usuario debe llenar, en él se pide número de cama, nombre del paciente, dirección, edad y peso, los cuales se pueden cambiar o se pueden añadir nuevos datos modificando esta rutina. Se cuenta también con el uso de dos teclas especiales: <ESC> que se utiliza en todos los casos para cancelar una opción y regresar al menú principal, y <F2> con la que se guardan los datos en disco duro.

Parámetros del paciente.- esta rutina es la encargada de actualizar los niveles de activación de todas las alarmas, al seleccionar esta opción, se despliega en pantalla un formato en el cual el usuario deberá escribir el número de cama, los niveles altos y bajos de activación de alarmas para FC, FR, PS y T, en este punto a diferencia de algunos equipos nacionales y otros comerciales que solamente tienen niveles de alarma determinados, se cuenta con la opción de poder elegir cualesquiera niveles, para activar las alarmas de todas las variables, cumpliendo así con los requerimientos para el sistema de alarmas especificados en la norma 3.2.8 de la *Association for the Advancement of Medical Instrumentation* (AAMI). Esta opción cuenta también con las teclas de <ESC> y <F2> para retornar al menú principal y guardar la información respectivamente.

Baja de un paciente.- cuando se desea retirar un paciente de la base de datos, esta rutina es la encargada de eliminar todos sus datos incluyendo los niveles de activación de alarmas. Al seleccionar la opción se despliega una ventana preguntando por el número de cama a dar de baja, posteriormente se despliegan en pantalla los datos del paciente seleccionado, esto con el fin de verificar que sea el paciente que se desea dar de baja, finalmente se ratifica la opción seleccionada quedando limpio su registro y retornando el programa al menú principal.

Supervisión individual.- esta rutina se encarga de desplegar en forma gráfica los signos vitales del paciente, dibujando la señal ECG y mostrando los valores numéricos de FC, FR, PS y T, indicando también condiciones anómalas en los signos vitales. En la siguiente sección se discutirán los algoritmos empleados para la graficación de los signos vitales.

Supervisión global.- al igual que en la rutina anterior al ejecutar esta opción se grafican los datos de todos los MBM conectados al PCE, en esta opción se grafica el ECG y se detectan anomalías en signos vitales para la activación de alarmas. Estos algoritmos se discuten en la siguiente sección.

Impresión.- esta rutina se encarga de adquirir un bloque de datos, es decir 4 periodos de la señal de ECG, con sus respectivos datos de FC, FR, PS y T, y los imprime junto con los datos del paciente, la rutina permite elegir dos tipos de impresora, matriz de puntos y láser; esta opción no se utiliza para tener registro del monitoreo en tiempo real, ya que la velocidad de impresión es de aproximadamente 3 minutos por gráfica, solamente se incluye para tener un registro de un determinado momento del monitoreo.

5.3.- Programación para monitoreo de pacientes

Esta parte del programa está formada por tres rutinas gráficas, dos de ellas se utilizan para el monitoreo de signos vitales y una para la impresión de la señal en papel. Como se está graficando en tiempo real los procedimientos no pueden ser muy largos, se deben ejecutar en el menor tiempo posible, por ello se utilizan utilerías gráficas de turbo C llamadas "graphics.h". Las rutinas de supervisión individual y global involucran procedimientos gráficos semejantes, por lo que detallando el algoritmo para supervisión global se entenderá el individual, estas rutinas cuentan con varios procedimientos, como se observa en el diagrama de flujo de la figura 5.2, los cuales se describen a continuación.

Inicializa.- en esta parte se inicializan todas las variables involucradas y se inicilaliza el modo gráfico de C para poder ejecutar las rutinas gráficas.

Protocolos.- para iniciar la identificación de los MBM conectados a la red se envía el protocolo de comunicaciones, dejando a cada MBM conectado listo para el intercambio de información.

Estando desplegada la pantalla de supervisión global y mientras no se pida supervisar un MBM en particular, se ejecutarán los siguientes procedimientos:

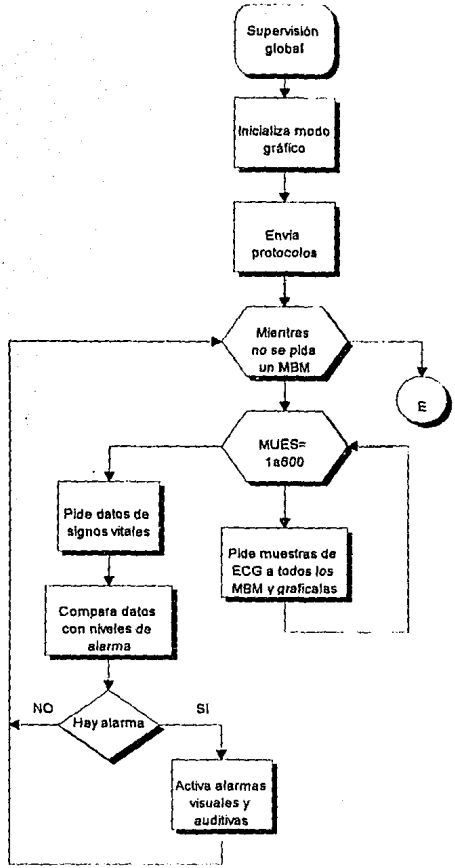


Figura 5.2 Diagrama de flujo para la programación de monitoreo de pacientes

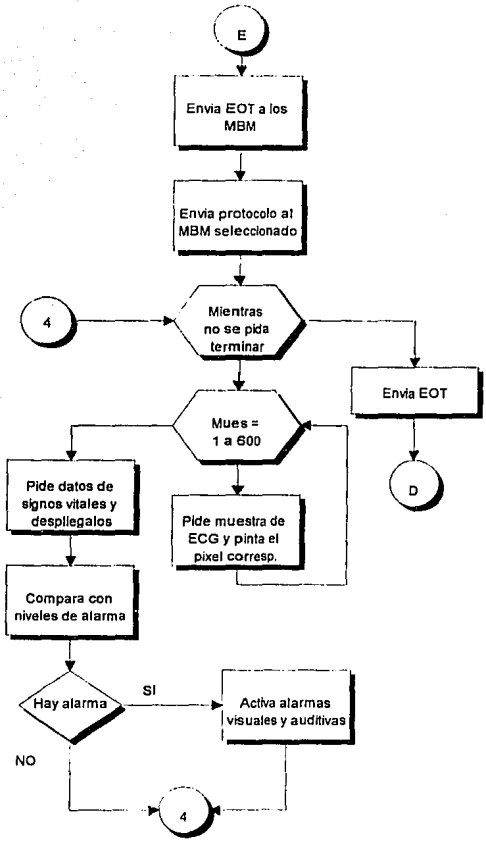


Figura 5.2 Continuación.

Muestras.- una vez identificados los MBM activos se procede a pedir muestras de ECG en orden ascendente, es decir, cada MBM es identificado con un número del 1 al 14, siendo en ese orden la petición de muestras de ECG, se pide una muestra y se pinta el pixel correspondiente de cada MBM, continuando con este proceso hasta completar 150 muestras, que es la resolución que tenemos en el monitor VGA para cada MBM, lo que representa una baja resolución para el monitoreo de ECG, sin embargo el objetivo de este modo de supervisión es el detectar signos vitales anómalos para encender las alarmas y pasar entonces a supervisar de manera individual.

Parámetros.- al completarse la graficación de las 150 muestras de ECG se ejecuta este procedimiento, en el cual se piden los datos de FC, FR, PS y T, los cuales serán comparados con los niveles de alarma proporcionados por el personal médico en la opción de parámetros del paciente.

Alarma.- en el caso de que la comparación reporte signos vitales anómalos, se ejecuta este procedimiento, el cual activa las alarmas visuales y auditivas, las visuales constituyen botones gráficos a los que se añade color y van acompañadas por tonos audibles de corta duración.

Si se pide monitorear algún MBM se ejecutaran los siguientes procedimientos, los cuales corresponden a la supervisión individual.

Termina.- este procedimiento envía el identificador de fin de transmisión a todos los MBM activos, dejándolos listos para iniciar un nuevo protocolo.

Protocolo.- esta rutina envía el protocolo solamente al MBM seleccionado, es decir solamente habrá intercambio de información con él.

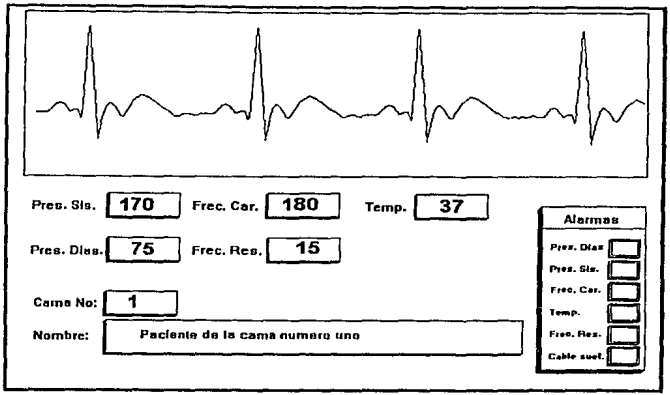
Muestra_indiv.- este procedimiento, solicita una muestra de ECG al MBM seleccionado y grafica su pixel correspondiente, continuando con este proceso hasta completar 600 muestras (deteminado por la resolución del monitor), lo que representa una frecuencia de muestreo de 250 Hz para una visualización de 4 periodos de ECG, tomando como base la señal mostrada en la figura 3.1; es decir se tiene máxima resolución para el monitoreo de ECG.

Parametros_indiv.- al terminar de graficar las 600 muestras de ECG se ejecuta este procedimiento, el cual pide los datos de los otros signos vitales, los despliega en pantalla y realiza la comparación con los niveles de activación de alarmas, los que en caso de ser revasados activan la rutina de encendido de alarmas descrita anteriormente.

Termina.- al ejecutarse este procedimiento se envía el comando de fin de transmisión al MBM y regresa el programa al menú principal.

El programa diseñado puede correr en cualqier sistema con 640Kb de memoria, puerto serie, disco duro y monitor VGA, pudiendo correr con algunas modificaciones en un monitor EGA o CGA.

En la figura 5.3 se muestran las pantallas para el modo de supervisión individual y global.



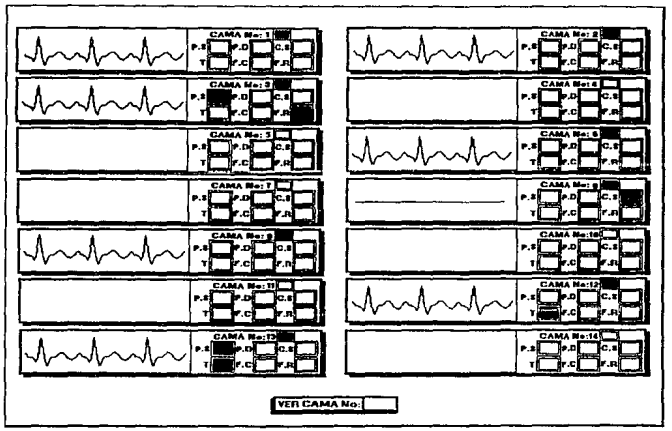


Figura 5.3 Pantallas para monitoreo individual y global.

PRUEBAS REALIZADAS AL MÓDULO BÁSICO DE MONITOREO Y AL PUESTO CENTRAL DE ENFERMERAS

6.1 .- INTRODUCCIÓN

Como se ha mencionado en los capítulos anteriores el equipo está diseñado de manera modular, en donde cada módulo fue probado de manera individual y posteriormente de manera conjunta al integrarse todos los módulos. En el presente capitulo se describirán las pruebas de funcionamiento realizadas al módulo digital (MBM) y al PCE, así como las pruebas realizadas a todos los módulos trabajando en conjunto a excepción del modulo FSK, el cual se encuentra en proceso de validación. Las pruebas realizadas se dividen en tres partes: pruebas modulares, en las cuales se probó cada módulo simulando las señales a procesar; pruebas con simulador, en las cuales se probó el equipo de manera conjunta utilizando como fuente de señal un simulador de ECG, el cual es utilizado para la calibración de equipo médico, y pruebas con paciente, en las cuales se cambia el simulador por un paciente voluntario; este último tipo de pruebas se realizaron previo a la validación del aislamiento del equipo, realizada por personal calificado en el área de ingeniería biomédica.

6.2.- pruebas modulares

Este tipo de pruebas se llevaron a cabo de acuerdo al avance del proyecto, probando primeramente el software diseñado para el sistema, es decir la programación del PCE y los programas en ensamblador para el MBM, posteriormente se utilizaron ambos programas para llevar a cabo la prueba de la arquitectura del MBM.

Pruebas a la programación del PCE

El objetivo de estas pruebas es verificar el buen funcionamiento de las rutinas en ensamblador para el MBM y las rutinas de graficación de ECG, así como la obtención y presentación de los valores numéricos de los signos vitales del paciente y la correcta activación de las alarmas. Para las rutinas de graficación se utilizaron archivos de datos para simular la señal de ECG y los valores de los signos vitales, la graficación de estos datos se realizó de manera adecuada en el modo de supervisión individual, para el modo de supervisión global se utilizó el mismo archivo de datos para los 14 MBM, en ambos casos la graficación se obtuvo de una manera nítida y sin problemas.

En cuanto a la programación en ensamblador del MBM, ésta se realizó probando por separado el buen funcionamiento de cada subrutina de procesamiento, utilizando para ello la ayuda de un simulador del microcontolador llamado "AVSIM11" , con el cual podemos representar en la computadora todas las características del MCU y así evaluar los programas elaborados. Con el simulador se verificó el buen funcionamiento de las rutinas de conversión y almacenamiento de datos, el cálculo de FC y FR, el monitoreo de puertos, transmisión y recepción de datos através del puerto serie; todo realizado en tiempo real.

Pruebas realizadas al MBM

Estas pruebas fueron realizadas para probar la arquitectura digital diseñada para el MBM. Como el módulo de amplificación y filtrado se encontraba en proceso de desarrollo y validación se simularon las señales vitales a procesar como se observa en el diagrama de la figura 6.1.

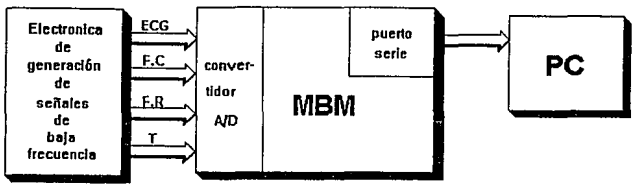


Figura 6.1 Diagrama de bloques para las pruebas realizadas al MBM

Se elaboró un circuito que genera señales de baja frecuencia, cubriendo el rango de la señal de ECG (0.5 - 4 Hz) y que entrega pulsos a la misma frecuencia (30 -240 pulsos por minuto) simulando la señal de FC y FR, además de proporcionar un voltaje correspondiente a la temperatura (15 - 45 °C), con este circuito se simularon la mayor parte de las características del módulo de amplificación y filtrado. Las señales generadas se conectan al convertidor A/D para ser procesadas y enviadas por el puerto serie hacia el PCE para su graficación. Con el correcto funcionamiento del *software* desarrollado se obtuvo un buen funcionamiento de la arquitectura diseñada.

6.3 .- Pruebas realizadas con simulador de ECG

El objetivo de este tipo de pruebas es realizar la integración de todos los módulos, es decir amplificación y filtrado, MBM y PCE, además de realizar la calibración adecuada del MBM para la correcta obtención de los valores numéricos de FC y FR; para lograrlo se utilizó como fuente de señal un simulador de ECG, el cual

cuenta con todas las derivaciones posibles [3] para la generación de la señal ECG y frecuencias de 30, 60, 120 y 240 pulsos por minuto, con lo que se cubre el rango de frecuencias para monitoreo cardiaco; este simulador es utilizado en hospitales para la calibración de equipo de monitoreo cardiaco, por lo que se utilizó como referencia para la calibración del equipo diseñado. Las pruebas se realizaron con la configuración mostrada en el diagrama de la figura 6.2, se utilizaron los programas del MBM y el PCE probados anteriormente. Las pruebas de funcionamiento continuo proporcionaron los siguientes resultados:

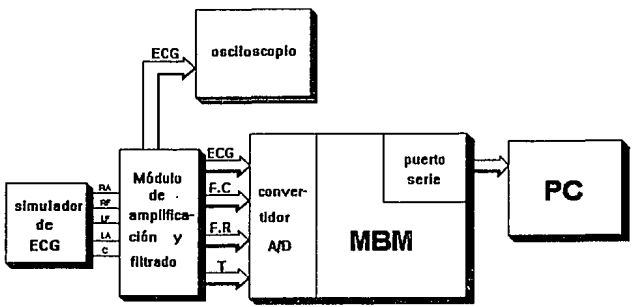


Figura 6.2 Diagrama de bloques para las pruebas realizadas con simulador.

- La visualización de la señal de ECG fue satisfactoria, es decir se presenta una señal con buena nitidez comparándola con la señal original mostrada en el osciloscopio.

-Los valores numéricos de FC, FR y T son aceptables en el rango de 30 a 120 ppm; en el rango de 240 ppm se tiene un error de 2 ppm, el cual de acuerdo a las normas AAMI es aceptable y coincide con el error teórico esperado, en la siguiente tabla se muestran estas características de funcionamiento.

simulador	MBM	error
FC en ppm	FC en ppm	en ppm
30	30	0
60	60	0
120	121	1
240	238	2

-La activación de alarmas se realizo de manera adecuada al sobrepasar los límites establecidos.

En la figura 6.3 se muestra una impresión de la señal ECG y los datos de los signos vitales obtenidos al conectar el simulador, esta gráfica nos muestra el buen funcionamiento del software diseñado para la impresión y graficación de la señal.

6.4.- Pruebas realizadas con paciente

Este tipo de pruebas requiere de un control adecuado del equipo ya que al encontrarse conectado directamente al cuerpo puede resultar peligroso para el paciente si el equipo no está debidamente aislado. Estas pruebas se realizaron bajo la supervisión de personal especializado, los cuales realizaron la validación del aislamiento del equipo. Con la ayuda de un voluntario se realizaron las pruebas de monitoreo obteniendo los siguientes resultados:

-El monitoreo de la señal de ECG es nítido pero presenta desplazamientos de la señal a medida que se mueve el paciente, esto es debido a que el módulo amplificador no cuenta con circuitos de corrección de línea basal [3].

-Los valores de los signos vitales los podemos considerar correctos, ya que el equipo se probó y calibró con el simulador de ECG. El valor de FR se encuentra dentro del rango normal, comparado con un equipo comercial calibrado, con lo que aseguramos que se trata de un valor correcto de FR. La temperatura desplegada se comparó con un termómetro digital, con lo cual podemos asegurar su correcto funcionamiento. Los sensores del equipo diseñado y los de equipos comerciales se colocaron al mismo paciente para realizar una comparación de ambos equipos. En el siguiente cuadro se muestran los resultados obtenidos en esta prueba.

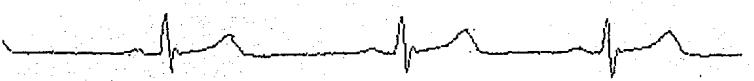
	MBM			Equipo comercial		
No lectura	FC	FR	Temp.	FC	FR	Temp
	(ppm)	(ppm)	(°C)	(ppm)	(ppm)	(°C)
1	58	14	36.4	58	13	36.5
2	60	13	36.6	60	13	36.6
3	60	15	36.1	60	15	36.0
4	65	15	36.3	65	15	36.5
5	71	17	37.0	70	16	36.9
6	78	18	36.6	78	18	36.6
7	70	16	36.4	71	15	36.4
8	68	14	36.7	68	14	36.6
9	60	13	36.3	60	13	36.3
10	60	12	36.4	60	12	36.1

En la figura 6.4 se muestra la impresión de siete derivaciones de ECG y los signos vitales del paciente, mientras que en la figura 6.4a se muestran algunas impresiones de otros equipos diseñados en el país. Es importante mencionar que para una validación clínica, el equipo debe ser probado y validado por personal médico especializado.

En la siguiente tabla se enlistan las especificaciones obtenidas hasta el momento para el equipo diseñado.

ESPECIFICACIONES GENERALES

Características	Especificaciones
Impedancia de entrada	
modo común	mayor a $5x10^{10}\ \Omega$
modo diferencial	mayor a $10^{8}\Omega$
ECG	
respuesta en frecuencia	0.5 a 40 Hz
ganancia	1400
calibración	sin calibración
aislamiento	mayor a 120 dB
Frecuencia cardiaca	
rango	20 a 250 PPM
precisión	±3% o 3 PPM
límite de alarmas	fijado por el usuario en cualquier rango
Frecuencia respiratoria	
rango	8 a 150 PPM
precisión	±3 PPM
límite de alarmas	fijado por el usuario en cualquier rango
Temperatura	
rango	15 a 45°C
precisión	±0.2°C
límite de alarmas	fijado por el usuario en cualquier rango
Comunicación	
Longitud de palabra	8 bits de datos, 1 bit de inicio, 1 bit de stop, sin paridad
modo de operación	fullduplex
velocidad de transmisión	9600 bauds
tipo de comunicación	asíncrona
interface	RS-232
Sistema digital	
frecuencia de muestreo	250 Hz
resolución del convertidor	8 bits
velocidad de conversión	16 microsegundos
rango de conversión	0 a 5 volts
memoria RAM	2 Kbytes
memoria EPROM	4 Kbytes



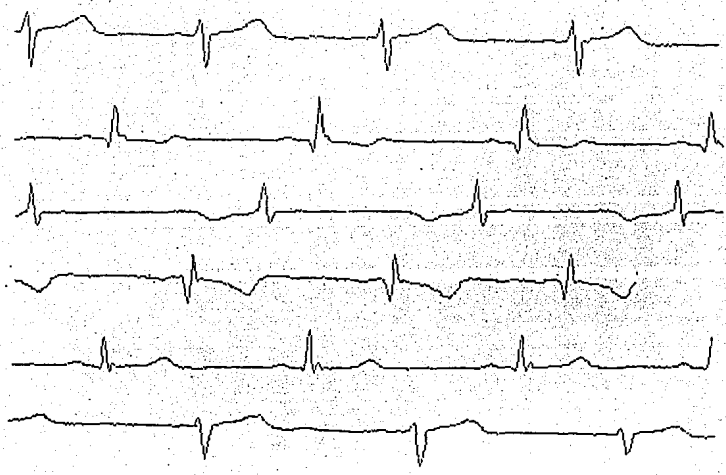
Pres. Sist. : 1 Frec. Car. : 61 Temp. : 36.7

Pres. Diast. : 1 Frec. Resp. : 7

Cama No.: 1

Nombre :

Figura 6.3 Graficación se signos vitales con simulador.



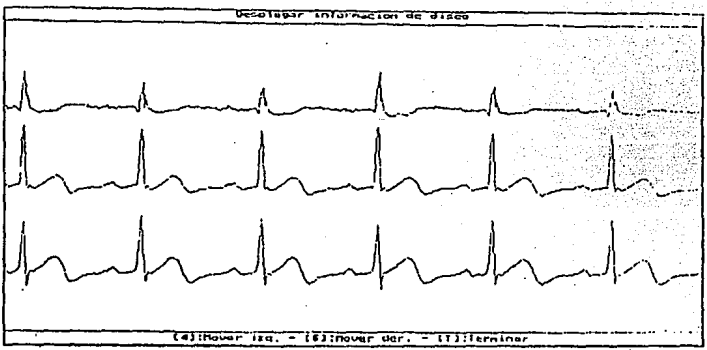
Pres. Sist. : 1 Frec. Car. : 68 Temp. : 36.7

Pres. Diast. : 1 Frec. Resp. : 14

Cama No.: 1

Nombre : Ricardo Silva G.

Figura 6.4 Graficación de signos vitales con paciente.



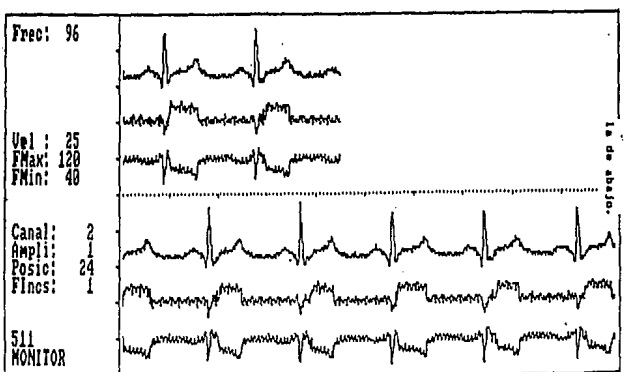


Figura 6.4a Graficación de signos vitales de equipos nacionales.

CONCLUSIONES Y RECOMENDACIONES

A lo largo del presente trabajo se han descrito las características de diseño y construcción de un sistema modular para el monitoreo de signos vitales, se han detallado las características del módulo básico de monitoreo y el puesto central de enfermeras, módulos desarrollados en la presente tesis. A partir del trabajo desarrollado en el diseño, construcción, integración y pruebas realizadas a cada módulo se adquirieron experiencias que permitirán realizar mejoras al sistema de monitoreo.

En el presente capítulo se describen las conclusiones del trabajo elaborado en cada módulo y de las pruebas realizadas al sistema integrado, así mismo, se mencionan algunas recomendaciones que permitirán mejorar y concluir el sistema de monitoreo.

Conclusiones

1) Se construyó un sistema prototípo para el monitoreo de signos vitales, el cual despliega la señal del corazón ECG (electrocardiograma) e indica datos de frecuencia cardiaca, temperatura y frecuencia respiratoria, realiza además la activación de alarmas sobre signos vitales anómalos en el paciente y detección de electrodos sueltos, con posibilidades de integrar en el futuro la detección de presión sanguínea.

2) Se cumplió con el objetivo de realizar un diseño módular, desarrollando los módulos del puesto central de enfermeras y el módulo básico de monitoreo, éste último interactúa con el módulo de amplificación y filtrado para integrar el sistema de monitoreo.

3) Los módulos diseñados en el presente trabajo (MBM y PCE), así como los diseñados en otra tesis complementaria (amplificación y filtrado) cumplen satisfactoriamente con las normas establecidas por la "Association for the Advancement of Medical Instrumentation" (AAMI).

4) Se logró obtener un diseño compacto del MBM, su arquitectura está diseñada para poder integrar en el futuro un sistema de despliegue local usando un circuito de refresco de memoria con un tubo de rayos catódicos, o bien, una pantalla de cristal liquido.

5) En cuanto a la programación del MBM, ésta se realizó de manera módular, logrando con ello procesar datos en tiempo real. Este aspecto permitirá integrar las nuevas rutinas de procesamiento de presión sanguínea, así como actualizar cualquiera de los procedimientos escritos de manera sencilla.

6) Se integró un puesto central de enfermeras que realiza las funciones de despliegue de señales de un MBM y al mismo tiempo realiza tareas de administración de datos de los MBM's que se encuentren conectados en red, por el momento solo un MBM se conecta al PCE.

7) La programación del PCE se realizó de manera modular permitiendo un fácil mantenimiento para la adición de nuevas rutinas de procesamiento; las rutinas de gráficación permiten obtener con excelente nitidez la señal de ECG en comparación con algunos equipos nacionales e importados.

8) Por las características obtenidas el sistema podrá ser empleado por el momento como un sistema de monitoreo, pudiendo utilizarse posteriormente como una red de monitores de signos vitales.

Recomendaciones

1) Para que el sistema sea totalmente independiente del PCE es recomendable continuar el trabajo de despliegue local, siendo más conveniente (peso y potencia) el uso de pantallas de cristal liquido que cuenten con interfaz serie o paralelo.

2) Para aumentar la resolución de las señales adquiridas se puede disminuir la ventana de conversión en el MBM, teniendo así lecturas más exactas de los signos vitales.

3) Por el momento la electrónica se encuentra alambrada con la tecnica de "wire-wrap", por lo que realizados los cambios que sean necesarios se debe diseñar y fabricar el sistema en circuito impreso.

4) En el PCE se realiza solamente el despliegue de las señales y la administración de la red, sin embargo es conveniente realizar algún tipo de procesamiento para el análisis y reconocimiento de señales de ECG.

5) Para el uso de una red de MBM's se deben concluir los trabajos del modem FSK que permitirá conectar los 14 MBM's al PCE.

6) Es muy importante que el equipo sea examinado y validado por personal médico especializado en cardiología para probar su funcionamiento en el ambiente clínico.

1. CEDAT, "Catalogo de servicios y productos", Secretaría de salubridad y asistencia (SSA), 1992.

2. Richard Aston Principles of Biomedical Instrumentation and Measurements",Merrill Publishing Company, Columbus, 1990.

3. L.A.Geddes, L.E. Baker, "Principles of Applied Biomedical Instrumentation",Jhon Wiley, New York, 1975.

4. Motorola Semiconductors "Reference Manual MC68HC11", 1991.

5. Robert J. Mayhon, "Discrete Time and Continuos Time Linear System", Addison-Wesley, 1984.

6. Willis J. Tompkins. "Design of Microcomputer-based Medical Instrumentation"Webster, Prentice-Hall, 1981.

7. L.Edenbrandt, et al. "Reconstruction of the Electrocardiogram During Heart Surgery", Computers and Biomedical Research, vol.25,No 6, pp.538-546December 1992.

8. Wendell C. Ocasio, et al. "Bpshape WK4: A Computer Program That Implements a Physiological Model for Analyzing the Shape of Blood Pressure Waveforms", Computer Methods and Programs in biomedicinevol. 39, pp. 169-194, april 1993.

9. Association for the Advancement of Medical Instrumentation (AAMI), "Cardiac Monitor, Heart Rate Meters and Alarms Norms", EC13 - 1983.

10. Dynapulse 2000, "Looking Beyond Blood Presure Measurement, TecnicalSpecifications", Pulse Metric, Inc. 1993.

11. Yañez S. O. et al. "Detección de complejos QRS normales en registrosde electrocardiografía ambulatoria de 24 horas", Rev. Mex. Ing. Biomed.pp. 63 - 74, vol. II No. 1, Nov. 1990.

12. Tovar O. et al. "Monitor simulador de ECG para enseñanza", Rev. Mex. Ing. Biomed., pp. 79 - 85, vol. 9, 1988.
13. Schure Alexander, "Basic Television", Marcombo, 1974.

14. Schuartz Mischa, "Information, Transmission, Modulation and Noise", Mc Graw-Hill, 1990.

15. Gerardo A. Paulin, Edgar C. M. "Comunicación por medio de las lineas de alta tensión", Rev. Ingenio, ENEP Aragón UNAM, pp. 19 - 24, feb. 1986.

16. Kruglinski David, "Guia de las comunicaciones del IBM PC", Mc Graw-Hill, 1985.

17. Herbert Schildt, "Turbo C/C++, the Complete Reference", Mc Graw-Hill, 1992.

18. Pathfinder Associates Inc. "vlib Manual", 1991.

LISTADO DE PROGRAMACIÓN DEL MBM

```
0001                ********************************************
0002                *** ASIGNACION DE CONSTANTES ***
0003                ********************************************
0004  f000                      ORG $F000
0005  1039  OPTIONS   EQU $1039    activo C/AD
0006  102c  SCCR      EQU $102C    controla longitud de datos
0007  102b  BAUD      EQU $102B    controla baudaje
0008  102d  SCCR2     EQU $102D    controla Tx y Rx
0009  102e  SCSR      EQU $102E    registro de status
0010  102f  SCDR      EQU $102F    registro de datos
0011  1030  ADCTL     EQU $1030    control del C A/D
0012  1031  ADR1      EQU $1031    registros de datos
0013  1032  ADR2      EQU $1032    del convertidor
0014  1033  ADR3      EQU $1033    analógico-digital
0015  1034  ADR4      EQU $1034
0016  1000  PORTA     EQU $1000    puerto A
```

```
0017  0c10  CONTFC     EQU $0C10     contador de F.C.
0018  0c12  CONTFR     EQU $0C12     contadores para F.R.
0019  0c14  CONTA      EQU $0C14
0020  0c15  CONTB      EQU $0C15
0021  0c16  CONTECG    EQU $0C16     contador de ECG
0022  0c18  DTEMP      EQU $0C18     dato de temperatura
0023  0c1a  DFC        EQU $0C1A     dato de F.C
0024  0c1c  DFR        EQU $0C1C     dato de F.R
0025  0c1e  DCS        EQU $0C1E     dato de  C.S
0026  0c20  DPSS       EQU $0C20     dato de P.S.sistólica
0027  0c22  DPSD       EQU $0C22     dato de P.S diastólica

0028              ************************************************
0029              * INICIALIZACION
0030              ************************************************

0031  f000  8e 0f fe       LDS #$0FFE      inicializo el stack
0032  f003  c6 90           LDAB #$90       activo potencia para el
0033  f005  f7 10 39        STAB OPTIONS    C A/D
0034  f008  c6 00           LDAB #$00       fijo longitud de palabra
0035  f00a  f7 10 2c        STAB SCCR1      en 8 bits
0036  f00d  c6 30           LDAB #$30       fijo baudaje en 9600 bauds
0037  f00f  f7 10 2b        STAB BAUD
0038  f012  c6 0c           LDAB #$0C       habilito Tx y Rx
0039  f014  f7 10 2d        STAB SCCR2
0040  f017  18 ce 00 00     LDY #$0000      inicio cont. de F.C
0041  f01b  18 ff 0c 10     STY CONTFC      guardo cont. en dir. 0
0042  f01f  ce 00 00        LDX #$0000      inicio cont. de F.R
0043  f022  ff 0c 12        STX CONTFR      guardo cont. en dir. 2
0044  f025  86 00           LDAA #$00       inicio contadores de
0045  f027  b7 0c 14        STAA CONTA      F.C y F.R
0046  f02a  c6 00           LDAB #$00
0047  f02c  f7 0c 15        STAB CONTB
0048  f02f  ce 08 00        LDX #$0800      inicio cont. de
0049  f032  ff 0c 16        STX CONTECG     muestras de ECG
```

```
0050  f035  c6 b4        LDAB #$B4   inicio los datos
0051  f037  f7 0c 20     STAB DPSS   simulados de presion
0052  f03a  c6 3c        LDAB #$3C   sistolica y diastolica
0053  f03c  f7 0c 22     STAB DPSD

0054          ************************************************
0055          * PROGRAMA PRINCIPAL
0056          ************************************************
0057          ************************************************
0058          * PREGUNTA POR CLAVE
0059          ************************************************

0060  f03f  bd f1 a5  CLV: JSR RX      pregunta si llego algo
0061  f042  f6 10 2f       LDAB SCDR   si llego algo comparalo con
0062  f045  c1 01          CMPB #$01   clave
0063  f047  26 47          BNE NOCLV   si no es clave ve a noclv
0064  f049  c6 06          LDAB #$06   Tx un ack
0065  f04b  f7 10 2f       STAB SCDR
0066  f04e  bd f1 af       JSR TXC

0067          ************************************************
0068          * PREGUNTA POR INICIO DE Tx
0069          ************************************************

0070  f051  bd f1 a5  INTX: JSR RX     pregunta si llego algo
0071  f054  f6 10 2f       LDAB SCDR   si llego comparalo con
0072  f057  c1 10          CMPB #$10   inicio de transmision
0073  f059  26 4d          BNE NINTX   si no es inicio ve a nintx
0074  f05b  c6 01          LDAB #$01   si es el inicio Tx la clave
0075  f05d  f7 10 2f       STAB SCDR
0076  f060  bd f1 af       JSR TXC
```

```
0077                ****************************************************
0078                * PREGUNTA POR CHK
0079                ****************************************************

0080  f063  bd f1 a5   CHK:  JSR RX          pregunta si llego algo
0081  f066  f6 10 2f         LDAB SCDR       si llego comparalo con CHK
0082  f069  c1 1f            CMPB #$1F
0083  f06b  26 53            BNE VAR         si no es chk ve a VAR
0084  f06d  bd f1 b9         JSR TMGETC      si es chk toma muestra
0085  f070  fe 0c 16         LDX CONTECG
0086  f073  e6 00            LDAB $00,X
0087  f075  f7 10 2f         STAB SCDR       Tx muestra
0088  f078  bd f1 af         JSR TXC
0089  f07b  bd f1 dc         JSR F.C         calcula F.C
0090  f07e  bd f2 26         JSR F.R         calcula F.R
0091  f081  fe 0c 16         LDX CONTECG     cargo cont. de ECG
0092  f084  08               INX             incrementa dir. de datos
0093  f085  8c 0c 00         CPX #$0C00      si no llego al final
0094  f088  26 03            BNE SAL         ve a retardo
0095  f08a  ce 08 00         LDX #$0800      si llego al final
0096  f08d  7e f1 8d   SAL:  JMP RET         inicia contador de muestras

0097                ****************************************************
0098                * SECCION DE NO CLAVE
0099                ****************************************************

0100  f090  bd f1 b9 NOCLV:  JSR TMGETC      ve a tomar una muestra
0101  f093  bd f1 dc         JSR F.C         calcula F.C
0102  f096  bd f2 26         JSR F.R         calcula F.R
0103  f099  fe 0c 16         LDX CONTECG     cargo cont. de ECG
0104  f09c  08               INX             si dir. de los datos no
0105  f09d  8c 0c 00         CPX #$0C00      llego al final
0106  f0a0  26 03            BNE RENO        ve a retardo
0107  f0a2  ce 08 00         LDX #$0800      si llego al final inicio
0108  f0a5  7e f1 75   RENO:  JMP RNC         cont. de ECG
```

```
0109                ****************************************************
0110                * SECCION DE NO INICIO DE TX
0111                ****************************************************

0112   f0a8  bd f1 b9  NINTX:  JSR TMGETC   ve a tomar una muestra
0113   f0ab  bd f1 dc          JSR F.C      calcula F.C
0114   f0ae  bd f2 26          JSR F.R      calcula F.R
0115   f0b1  fe 0c 16          LDX CONTECG  cargo cont. de ECG
0116   f0b4  08                INX          si dir.de los datos no
0117   f0b5  8c 0c 00          CPX #$0C00   llego al final
0118   f0b8  26 03             BNE RENI     ve a retardo
0119   f0ba  ce 08 00          LDX #$0800   si llego al final
0120   f0bd  7e f1 81  RENI:   JMP RNINT    inicia cont. de muestras


0121                ****************************************************
0122                * SECCION DE ENVIO DE VARIABLES
0123                ****************************************************

0124   f0c0  c1 26     VAR:    CMPB #$26    envio variables?
0125   f0c2  26 79             BNE BEND     no, ve a endtx
0126   f0c4  f6 0c 18          LDAB DTEMP   si, envio todas las
0127   f0c7  f7 10 2f          STAB SCDR    variables
0128   f0ca  bd f1 af          JSR TXC
0129   f0cd  bd f1 a5          JSR RX
0130   f0d0  f6 10 2f  FREC:   LDAB SCDR
0131   f0d3  c1 27             CMPB #$27
0132   f0d5  26 f9             BNE FREC
0133   f0d7  b6 0c 1a          LDAA $0C1A
0134   f0da  81 00             CMPA #$00
0135   f0dc  26 62             BNE N.F
0136   f0de  f6 0c 1b          LDAB $0C1B   frecuencia cardiaca
0137   f0e1  f7 10 2f          STAB SCDR
0138   f0e4  bd f1 af          JSR TXC
0139   f0e7  bd f1 a5          JSR RX
0140   f0ea  f6 10 2f  FRER:   LDAB SCDR
```

```
0141  f0ed  c1 28          CMPB #$28
0142  f0ef  26 f9          BNE FRER
0143  f0f1  b6 0c 1c       LDAA $0C1C
0144  f0f4  81 00          CMPA #$00
0145  f0f6  26 53          BNE N.F.R
0146  f0f8  f6 0c 1d       LDAB $0C1D     frecuencia respiratoria
0147  f0fb  f7 10 2f       STAB SCDR
0148  f0fe  bd f1 af       JSR TXC
0149  f101  bd f1 a5       JSR RX
0150  f104  f6 10 2f  CASU: LDAB SCDR
0151  f107  c1 29          CMPB #$29
0152  f109  26 f9          BNE CASU
0153  f10b  f6 0c 1e       LDAB DCS       cable suelto
0154  f10e  f7 10 2f       STAB SCDR
0155  f111  bd f1 af       JSR TXC
0156  f114  bd f1 a5       JSR RX
0157  f117  f6 10 2f  PRES: LDAB SCDR
0158  f11a  c1 30          CMPB #$30
0159  f11c  26 f9          BNE PRES
0160  f11e  f6 0c 20       LDAB DPSS      presion sanguinea sist.
0161  f121  f7 10 2f       STAB SCDR
0162  f124  bd f1 af       JSR TXC
0163  f127  bd f1 a5       JSR RX
0164  f12a  f6 10 2f  PRED: LDAB SCDR
0165  f12d  c1 31          CMPB #$31
0166  f12f  26 f9          BNE PRED
0167  f131  f6 0c 22       LDAB DPSD      presion sanguinea dist.
0168  f134  f7 10 2f       STAB SCDR
0169  f137  bd f1 af       JSR TXC
0170  f13a  7e f0 63       JMP CHK        ve a CHK
0171  f13d  7e f1 56  BEND: JMP ENDTX
0172  f140  86 00       N.F: LDAA #$00
0173  f142  b7 10 2f       STAA SCDR
0174  f145  bd f1 af       JSR TXC
0175  f148  7e f0 ea       JMP FRER
0176  f14b  86 00      N.F.R: LDAA #$00
```

```
0177   f14d   b7 10 2f          STAA SCDR
0178   f150   bd f1 af          JSR TXC
0179   f153   7e f1 04          JMP CASU


0180              ****************************************************
0181              * SECCION DE FIN DE TRANSMISION
0182              ****************************************************

0183   f156   c1 04    ENDTX:  CMPB #$04       si no es fin de tx
0184   f158   26 03            BNE NCHK        ve a nchk
0185   f15a   7e f0 3f          JMP CLV


0186              ****************************************************
0187              * SECCION DE NO ENVIO DE MUESTRA
0188              ****************************************************

0189   f15d   bd f1 b9  NCHK:  JSR TMGETC      ve a tomar una muestra
0190   f160   bd f1 dc          JSR F.C         calcula F.C
0191   f163   bd f2 26          JSR F.R         calcula F.R
0192   f166   fe 0c 16          LDX CONTECG
0193   f169   08                INX             si dir. de los datos no
0194   f16a   8c 0c 00          CPX #$0C00      llego al final
0195   f16d   26 2a             BNE RNCHK       ve a retardo
0196   f16f   ce 08 00          LDX #$0800      si llego al final
0197   f172   7e f1 99          JMP RNCHK       inicia cont. de muestras


0198              ****************************************************
0199              * RETARDOS
0200              ****************************************************

0201   f175   ff 0c 16     RNC:  STX CONTECG genera un retardo
0202   f178   ce 03 b4           LDX #$03B4    y va a preguntar
0203   f17b   09           CIN:  DEX           por clave
```

```
0204  f17c  26 fd          BNE CIN
0205  f17e  7e f0 3f       JMP CLV
0206        ************************************************
0207  f181  ff 0c 16  RNINT: STX CONTECG  genera un retardo
0208  f184  ce 03 a2       LDX #$03A2    y va a preguntar
0209  f187  09        SEIS: DEX           por inicio de tx
0210  f188  26 fd          BNE SEIS
0211  f18a  7e f0 51       JMP INTX
0212        ************************************************
0213  f18d  ff 0c 16    RET: STX CONTECG  genera un retardo
0214  f190  ce 02 44       LDX #$0244    y va a preguntar
0215  f193  09        OCHO: DEX           por chk
0216  f194  26 fd          BNE OCHO
0217  f196  7e f0 63       JMP CHK
0218        ************************************************
0219  f199  ff 0c 16  NCHK: STX CONTECG  genera un retardo
0220  f19c  ce 03 a1       LDX #$03A1    y va a preguntar
0221  f19f  09        SIETE: DEX          por chk
0222  f1a0  26 fd          BNE SIETE
0223  f1a2  7e f0 63       JMP CHK


0224        ************************************************
0225        * SUBRUTINAS
0226        ************************************************
0227        *RECEPPCION
0228        ************************************************

0229  f1a5  f6 10 2e     RX: LDAB SCSR     pregunta si llego dato
0230  f1a8  c4 20           ANDB #$20
0231  f1aa  c1 20           CMPB #$20     por el puerto serie
0232  f1ac  26 f7           BNE RX
0233  f1ae  39              RTS
```

```
0234                ***************************************************
0235                * TRANSMISION
0236                ***************************************************

0237   f1af   f6 10 2e    TXC:  LDAB SCSR      envia un dato por el
0238   f1b2   c4 80              ANDB #$80
0239   f1b4   c1 80              CMPB #$80      puerto serie
0240   f1b6   26 f7              BNE TXC
0241   f1b8   39                 RTS


0242                ***************************************************
0243                *TOMA MUESTRA Y GUARDA
0244                ***************************************************

0245   f1b9   c6 10   TMGETC:  LDAB #$10        activo 4 canales de
0246   f1bb   f7 10 30           STAB ADCTL     conversion
0247   f1be   f6 10 30      TC:  LDAB ADCTL     pregunto si termino
0248   f1c1   c4 80              ANDB #$80      la conversion
0249   f1c3   c1 80              CMPB #$80
0250   f1c5   26 f7              BNE TC
0251   f1c7   fe 0c 16           LDX CONTECG    guardo dato de ECG en la
0252   f1ca   f6 10 31           LDAB ADR1      direccion de CONTECG
0253   f1cd   e7 00              STAB $00,X
0254   f1cf   f6 10 32           LDAB ADR2      guardo dato de temperatura
0255   f1d2   f7 0c 18           STAB DTEMP
0256   f1d5   f6 10 00           LDAB PORTA     leo el puerto a y guardo
0257   f1d8   f7 0c 1e           STAB DCS       el dato de cable suelto
0258   f1db   39                 RTS
```

```
0259                ***************************************************
0260                * FRECUENCIA CARDIACA
0261                ***************************************************

0262  f1dc  f6 10 33    F.C:  LDAB ADR3      cargo el dato de F.C
0263  f1df  c1 80             CMPB #$80      si el dato es bajo
0264  f1e1  23 1b             BLS NOFCHI     ve a nofchi
0265  f1e3  b6 0c 14           LDAA CONTA     si es alto cargo cont a
0266  f1e6  4c                 INCA           para calcular o inic.
0267  f1e7  81 01              CMPA #$01      el contador de F.C
0268  f1e9  26 2a              BNE NOA        ve a noa
0269  f1eb  b7 0c 14           STAA CONTA     calcula la frecuencia
0270  f1ee  fe 0c 10           LDX CONTFC     cardiaca en pulsos por min
0271  f1f1  c6 07              LDAB #$07      "parte alta del pulso"
0272  f1f3  3a                 ABX
0273  f1f4  cc 4e 20           LDD #$4E20
0274  f1f7  02                 IDIV
0275  f1f8  ff 0c 1a           STX DFC
0276  f1fb  7e f2 25           JMP TER
0277  f1fe  18 fe 0c 10 NOFCHI: LDY CONTFC cargo el CONTFC, lo
0278  f202  18 08             INY            incrementa y la guarda
0279  f204  86 00              LDAA #$00
0280  f206  18 ff 0c 10        STY CONTFC
0281  f20a  b7 0c 14           STAA CONTA
0282  f20d  c6 0b              LDAB #$0B      retardo
0283  f20f  5a           CUA:  DECB
0284  f210  26 fd              BNE CUA
0285  f212  7e f2 25           JMP TER
0286  f215  b7 0c 14     NOA:  STAA CONTA     inicia contador de
0287  f218  18 ce 00 00        LDY #$0000     frecuencia cardiaca
0288  f21c  18 ff 0c 10        STY CONTFC
0289  f220  c6 0a              LDAB #$0A      retardo
0290  f222  5a           TRES: DECB
0291  f223  26 fd              BNE TRES
0292  f225  39            TER:  RTS
```

```
0293                ********************************************************
0294                * FRECUENCIA RESPIRATORIA
0295                ********************************************************

0296  f226  b6 10 34    F.R:  LDAA ADR4      cargo el dato de FR
0297  f229  81 80             CMPA #$80      si el dato es bajo
0298  f22b  23 1b             BLS NOFRHI     ve a NOFRHI
0299  f22d  f6 0c 15           LDAB CONTB     si es alto cargo cont b
0300  f230  5c                INCB           para calcular o inic
0301  f231  c1 01             CMPB #$01      el contador de FR
0302  f233  26 27             BNE NOB        ve a nob
0303  f235  f7 0c 15           STAB CONTB     calcula la frecuencia
0304  f238  fe 0c 12           LDX CONTFR     respiratoria en ppm
0305  f23b  c6 07              LDAB #$07      parte alta del pulso
0306  f23d  3a                 ABX
0307  f23e  cc 4e 20           LDD #$4E20
0308  f241  02                 IDIV
0309  f242  ff 0c 1c           STX DFR
0310  f245  7e f2 6a           JMP FIN
0311  f248  fe 0c 12 NOFRHI:   LDX CONTFR     cargo el cont. de FR, lo
0312  f24b  08                 INX            incrementa y lo guarda
0313  f24c  c6 00              LDAB #$00
0314  f24e  ff 0c 12           STX CONTFR
0315  f251  f7 0c 15           STAB CONTB
0316  f254  c6 0b              LDAB #$0B      retardo
0317  f256  5a           DOS:  DECB
0318  f257  26 fd              BNE DOS
0319  f259  7e f2 6a           JMP FIN
0320  f25c  f7 0c 15     NOB:  STAB CONTB     inicia contador de
0321  f25f  ce 00 00           LDX #$0000     _ frecuencia respiratoria
0322  f262  ff 0c 12           STX CONTFR
0323  f265  c6 0a              LDAB #$0A      retardo
0324  f267  5a           UNO:  DECB
0325  f268  26 fd              BNE UNO
0326  f26a  39            FIN:  RTS
0327                          END            fin del programa
```

Listados de programas elaborados para el PCE

Progrma para la interacción con el personal médico.

```
                                /* Archivos de cabecera */
#include <vlib.h>
#include <vlibmenu.h>
#include <stdio.h>
#include <conio.h>
#include <vlibkeys.h>
#include <vlibform.h>
#include <string.h>
#include <vlibmous.h>
#include <stdlib.h>

                                /* Declaración de constantes */
#define size 14
#define si 1
#define no 0

                                /* Declaración de variables */
int com,men,i;
int esckeys[]={ ESC,K_F2,0 };
char nom[35],dom[50];
int edad,peso,cama,pcama,fca,fcb,fra,frb,psa,psb;
int ta,tb;

                                /* Declaración de funciones */

void alta(void),baja(void),guarda(void),limpia(void),signos(void),global(void);
void param(void),guarpar(void),abreread(void),abrewrte(void),tolread(void);
void tolwrite(void),cara(void),imprime(void);
```

```
                    /*  Estructura del menú principal  */

V_MENU menu[]=
{
  {"1.-DAR DE ALTA UN PACIENTE",NULL,1},
  {"                          ",NULL,-2},
  {"2.-FIJAR INDICES DEL PACIENTE",NULL,3},
  {"                          ",NULL,-4},
  {"3.-DAR DE BAJA UN PACIENTE",NULL,5},
  {"                          ",NULL,-6},
  {"4.-MOSTRAR SIGNOS DE UN PACIENTE",NULL,7},
  {"                          ",NULL,-8},
  {"5.-SUPERVISION GLOBAL DE PACIENTES",NULL,9},
  {"                          ",NULL,-10},
  {"6.-IMPRESION DE SIGNOS DE UN PACIENTE",NULL,11},
  {"                          ",NULL,-12},
  {"7.-SALIR",NULL,13},
  {NULL}
   };

                    /*  Estructura de formato de altas  */

struct v_form_def form[]=
{
 8,25,VF_INT,2,"CAMA No",&cama,
 10,25,VF_STRING,sizeof(nom),"NOMBRE",nom,
 12,25,VF_STRING,sizeof(dom),"DOMICILIO",dom,
 14,25,VF_INT,3,"EDAD",&edad,
 16,25,VF_INT,3,"PESO",&peso,
 0,0,VF_EOL
 };
struct v_form_def form1[]=
{
 8,25,VF_INT,2,"CAMA No",&pcama,
 10,25,VF_INT,3,"FREC.CAR.ALTA",&fca,
 10,50,VF_INT,3,"FREC.CAR.BAJA",&fcb,
 12,25,VF_INT,3,"FREC.RESP.ALTA",&fra,
 12,50,VF_INT,3,"FREC.RESP.BAJA",&frb,
 14,25,VF_INT,3,"PRES.SANG.ALTA",&psa,
 14,50,VF_INT,3,"PRES.SANG.BAJA",&psb,
 16,25,VF_INT,3,"TEMP.ALTA",&ta,
 16,50,VF_INT,3,"TEMP.BAJA",&tb,
 0,0,VF_EOL
 };
```

```
                /* Estructura para guardar los datos en un archivo */

struct pac {
        int gcama;
        char gnom[35];
        char gdom[50];
        int gedad;
        int gpeso;

         }gpac_pac[size],spac[size];
struct tol {
        int gpcama;
        int gfca;
        int gfcb;
        int gfra;
        int gfrb;
        int gpsa;
        int gpsb;
        int gta;
        int gtb;
        }gtol_tol[size];
struct dat{
     int camapac;
     int impre;
     }gdat_dat;

                /* Estructura para seleccionar numero de cama   */

V_MULTISELECT list[]=
{
  1,"CAMA No: 1",
  0,"CAMA No: 2",
  0,"CAMA No: 3",
  0,"CAMA No: 4",
  0,"CAMA No: 5",
  0,"CAMA No: 6",
  0,"CAMA No: 7",
  0,"CAMA No: 8",
  0,"CAMA No: 9",
  0,"CAMA No:10",
  0,"CAMA No:11",
  0,"CAMA No:12",
  0,"CAMA No:13",
  0,"CAMA No:14",
  NULL, };
```

```
V_MULTISELECT list1[]=
{
 1,"EPSON (DOT-MATRIX)",
 0,"HP LASERJET III",
 NULL,
 };

                                /*  programa principal  */

main()
{
/*cara();*/
 vInit();              /* inicializo vlib */


/*   Fijo los colores para el menu, formas y cajas de dialogo  */

 vSetColor(0,
          V_LIGHT_GREEN|V_BLUE_BG,
          V_BLUE|V_GREEN_BG,
          V_BRIGHT_WHITE|V_BLUE_BG);
 vSetColor(V_MENU_GRP,
          V_RED_BG|V_WHITE,
          V_GREEN_BG|V_BLUE,
          V_RED_BG|V_BRIGHT_WHITE);
 vSetColor(V_FORM_GRP,
          V_BLUE_BG|V_WHITE,
          V_GREEN_BG|V_BLUE,
          V_BLUE_BG|V_BRIGHT_WHITE);
 vSetColor(V_POPUP_GRP,
          V_RED_BG|V_WHITE,
          V_GREEN_BG|V_RED,
          V_RED_BG|V_BRIGHT_WHITE);

cara();          /* despliega carátula */
 men=1;
 while(men==1)
  {
     clrscr();
     textbackground(BLUE);
     clrscr();
     vBorder();                          /* dibuja borde */
     vWriteStr(2,30,V_BRIGHT,"OPCIONES DE TRABAJO");  /* escribe letrero */
     vFillAttr(2,28,2,50,V_REVERSE);
     vFillAttr(3,29,3,50,V_BLACK);
```

```
    vFillAttr(2,50,3,50,V_BLACK);
    com=vPopMenuT(7,20,menu,1,"SELECCIONE OPCION");/*despliega menú */
    switch(com)
      {
          case 1:
                clrscr();
                alta();     /* alta de un paciente */
                men=1;
                break;
          case 3:
                clrscr();
                param();  /* fija parámetros de alarmas */
                men=1;
                break;
          case 5:
                clrscr();
                baja();     /* baja de un paciente */
                men=1;
                break;
          case 7:
                clrscr();
                signos();   /* signos de un paciente */
                men=1;
                break;
          case 9:
                clrscr();
                global();   /* supervisión global de pacientes */
                break;
          case 11:
                clrscr();
                imprime();
                men=1;
                break;
          case 13: clrscr();
                break;
          default: return(FALSE);
        }
        clrscr();
    }
clrscr();

}       /* fin del main */
```

```
                    /*  Función para dar de alta un paciente  */

void alta(void)
{
 int dat,mas;
 mas=1;
 while(mas==1)
 {
 vBorder();
 vWriteStr(2,30,V_BRIGHT,"ALTA DE UN PACIENTE");
 vFillAttr(2,28,2,50,V_REVERSE);
 vFillAttr(3,29,3,50,V_BLACK);
 vFillAttr(2,50,3,50,V_BLACK);
 dat=vForm(form,esckeys,"ESC = menu ,F2 = salvar");  /* despliega forma */

 switch(dat)
  {
      case 0: mas=0;
              break;
      case 1: clrscr();
              guarda();
              vClear();
              vFormClear(form);  /* limpia forma */
              mas=1;
              break;
      default: return(FALSE);

      }
        }
  vFormClear(form);
 }

                /*  Función para almacenar los datos en un archivo  */

void guarda (void)
{
 abreread();


      gpac_pac[cama-1].gcama=cama;
      strcpy(gpac_pac[cama-1].gnom,nom);
      strcpy(gpac_pac[cama-1].gdom,dom);
      gpac_pac[cama-1].gedad=edad;
      gpac_pac[cama-1].gpeso=peso;
```

```
    abrewrte();

 }
void abreread(void)
{
 int i;
 FILE *fp;
       if((fp=fopen("internos","r"))==NULL)    /* abre un archivo */
       {
         printf("no puedo abrir el archivo");
         return;
         }
       for(i=0;i<size;i++)
       {
         fread(&gpac_pac[i],sizeof(struct pac),1,fp);  /* lee los datos */
         }
       fclose(fp);       /* cierra el archivo */
       }
void abrewrte(void)
{
int i;
FILE *fp;
       if((fp=fopen("internos","w"))==NULL)
        {
         puts("no puedo abrir el archivo\n");
         return;
         }

       for(i=0;i<size;i++)
       {
        fwrite(&gpac_pac[i],sizeof(struct pac),1,fp);
         }
       fclose(fp);
       }

                    /* Función para introducir niveles de alarma */
void param(void)
{
 int par,mparam;
 mparam=1;
 while(mparam==1)
  {
 vBorder();
 vWriteStr(2,30,V_BRIGHT,"PARAMETROS DE PACIENTE");
 vFillAttr(2,28,2,56,V_REVERSE);
```

```
  vFillAttr(3,29,3,56,V_BLACK);
  vFillAttr(2,56,3,56,V_BLACK);
  par=vForm(form1,esckeys,"ESC = menu ,F2 = salvar");  /* despliega forma1 */
  switch(par)
   {
        case 0: mparam=0;
                break;
        case 1: clrscr();
                guarpar();
                vClear();
                vFormClear(form1);  /* limpia forma */
                mparam=1;
                break;
        default: return(FALSE);

        }
          }
    vFormClear(form);
  }
void guarpar(void)
{

        tolread();

        gtol_tol[pcama-1].gpcama=pcama;  /* guardo parametros */
        gtol_tol[pcama-1].gfca=fca;
        gtol_tol[pcama-1].gfcb=fcb;
        gtol_tol[pcama-1].gfra=fra;
        gtol_tol[pcama-1].gfrb=frb;
        gtol_tol[pcama-1].gpsa=psa;
        gtol_tol[pcama-1].gpsb=psb;
        gtol_tol[pcama-1].gta=ta;
        gtol_tol[pcama-1].gtb=tb;

        tolwrite();



  }

void tolread(void)
{
  int i;
  FILE *fc;
        if((fc=fopen("tol_par","r"))==NULL)    /* abre un archivo */
```

```
        {
          printf("no puedo abrir el archivo");
          return;
          }
        for(i=0;i<size;i++)
        {
          fread(&gtol_tol[i],sizeof(struct tol),1,fc);   /* lee los datos */
          }
        fclose(fc);        /* cierra el archivo */
        }

void tolwrite(void)
{
int i;
FILE *fc;
        if((fc=fopen("tol_par","w"))==NULL)
         {
          puts("no puedo abrir el archivo\n");
          return;
          }

        for(i=0;i<size;i++)
        {
         fwrite(&gtol_tol[i],sizeof(struct tol),1,fc);
          }
        fclose(fc);
}

                        /*  Función para dar de baja un paciente  */

void baja (void)
{

 int cual,wb,i,seguro,noseg;
noseg=1;
while(noseg==1)
{
 vBorder();
 vWriteStr(2,30,V_BRIGHT,"BAJA DE UN PACIENTE");
 vFillAttr(2,28,2,50,V_REVERSE);
 vFillAttr(3,29,3,50,V_BLACK);
 vFillAttr(2,50,3,50,V_BLACK);
 cual=vPopMultiselect(6,28,list,"Seleccione No. de cama",6); /* despliega caja de
dialogo */
 if(cual==0){noseg=1;break;}
```

```
wb=vWindowAdd(4,10,16,70);        /* crea una ventana */
vWindowGoto(wb);
vBorder2();                       /* dibuja borde de la nueva ventana */
vFillAttr(1,1,11,59,V_LIGHT_GREY);

abreread();

vPut(2,4,V_REVERSE,"CAMA No:%d\n",gpac_pac[cual-1].gcama);    /* depliega
datos */
vPut(4,4,V_REVERSE,"NOMBRE:%s\n",gpac_pac[cual-1].gnom);
vPut(6,4,V_REVERSE,"DOMICILIO:%s\n",gpac_pac[cual-1].gdom);
vPut(8,4,V_REVERSE,"EDAD:%d\n",gpac_pac[cual-1].gedad);
vPut(10,4,V_REVERSE,"PESO:%d\n",gpac_pac[cual-1].gpeso);
seguro=vPopYesNo(18,24,"¨Esta usted seguro?",1);
  switch(seguro)
  {
       case 0: noseg=0;
               vWindowDelete(wb);
               break;
       case 1: noseg=1;
               vClearAll();
               vWindowDelete(wb);
               break;
       case 2: limpia();
               noseg=0;
               vWindowDelete(wb);
               break;
        }
    }
}

                /*  Función que borra del archivo al paciente  */

void limpia(void)
{
int cual,i;
char bnom[2],bdom[2];
 gpac_pac[cual-1].gcama=0;
 strcpy(gpac_pac[cual-1].gnom,bnom);
 strcpy(gpac_pac[cual-1].gdom,bdom);
 gpac_pac[cual-1].gedad=0;
 gpac_pac[cual-1].gpeso=0;

 abrewrte();
 tolread();
```

```
        gtol_tol[cual-1].gpcama=0;  /* guardo parametros */
        gtol_tol[cual-1].gfca=0;
        gtol_tol[cual-1].gfcb=0;
        gtol_tol[cual-1].gfra=0;
        gtol_tol[cual-1].gfrb=0;
        gtol_tol[cual-1].gpsa=0;
        gtol_tol[cual-1].gpsb=0;
        gtol_tol[cual-1].gta=0;
        gtol_tol[cual-1].gtb=0;
    tolwrite();

}

                /*   Función que muestra los signos de un paciente   */

void signos(void)
 {
 int sistem;
 sistem=system("mbmrtc.exe");     /* sale al sistema operativo y ejecuta */
 if(sistem==-1){ printf("error");}

  }

                /*  Función que muestra los signos de los pacientes  */

void global(void)
{
 int sistem1;
 sistem1=system("sg.exe");
 if(sistem1==-1){printf("error");}
 }

                        /* Función que despliega la carátula */
 void cara(void)
 {
 int sistem2;
 sistem2=system("porta.exe");
 if(sistem2==-1){printf("error");}
 }

                    /* Función que imprime los datos del paciente */
 void imprime(void)
 {
  int cualimp,cualcama,i,sistem3;
  FILE *fd;
```

```
vBorder();
vWriteStr(2,30,V_BRIGHT,"IMPRESION DE SIGNOS");
vFillAttr(2,28,2,50,V_REVERSE);
vFillAttr(3,29,3,50,V_BLACK);
vFillAttr(2,50,3,50,V_BLACK);
cualimp=vPopMultiselect(10,25,list1,"Seleccione tipo de impresora",2);
cualcama=vPopMultiselect(10,25,list,"Seleccione cama a imprimir",6);

gdat_dat.impre=cualimp;
gdat_dat.camapac=cualcama;
      if((fd=fopen("gdat_dat","w"))==NULL)
       {
        puts("no puedo abrir el archivo\n");
        return;
        }

       fwrite(&gdat_dat,sizeof(struct tol),1,fd);
      fclose(fd);

sistem3=system("impredat.exe");
if(sistem3==-1){printf("error");}
  }
```

Programa para graficación de signos vitales.

```
                                /* Archivos de cabecera */
#include <stdio.h>
#include <dos.h>
#include <graphics.h>
#include <conio.h>
#include "c:\len\tc\graf.c"
#include "c:\len\tc\gprintf.c"

                                /* Declaración de constantes */
#define size 14
#define LONG 600
#define INI_T 0x10
#define EOT 0x04
#define CLAVE1 0x01
#define CLAVE2 0x02
#define CLAVE3 0x03
#define CLAVE4 0x14
#define CLAVE5 0x05
#define CLAVE6 0x06
```

```
#define CLAVE7 0x07
#define CLAVE8 0x08
#define CLAVE9 0x09
#define CLAVE10 0x0a
#define CLAVE11 0x0b
#define CLAVE12 0x0c
#define CLAVE13 0x0d
#define CLAVE14 0x0e

                        /* Declaración de variables */
float tem,btem;

unsigned short int frecc,frecr,press,presd,cs,CLAVE,CHK;
unsigned short int bfrecc,bfrecr,bpress,bpresd;

int graphdr,graphmo,i,nucam,ca,clavec;
int xb1,xb2,x1a,x2a,x3a,x4a,x5a,x6a,x7a,xb;
int yb1,yb2,yb3,yb4,yb5,yb6,yb7,y1a,y2a,y3a,
   y4a,y5a,y6a,y7a,yb;

unsigned short int ecgp[LONG],ecgb[LONG],ecgp1[LONG],ecgb1[LONG];
unsigned short int ecgp2[LONG],ecgb2[LONG],ecgp3[LONG],ecgb3[LONG];
unsigned short int ecgp4[LONG],ecgb4[LONG],ecgp5[LONG],ecgb5[LONG];
unsigned short int ecgp6[LONG],ecgb6[LONG],ecgp7[LONG],ecgb7[LONG];
unsigned short int ecgp8[LONG],ecgb8[LONG],ecgp9[LONG],ecgb9[LONG];
unsigned short int ecgp10[LONG],ecgb10[LONG],ecgp11[LONG],ecgb11[LONG];
unsigned short int ecgp12[LONG],ecgb12[LONG],ecgp13[LONG],ecgb13[LONG];
unsigned short int ecgp14[LONG],ecgb14[LONG];

                        /* Declaración de funciones */
void abrear(void);
void inicio(void);
void proto(void);
void mues();
void param(void),mparam(void),alarma(void);
void termina(void);
void pausa(void);
void indivi(void),protoin(void),paramin(void);

FILE *fp;
FILE *fc;

                        /* Declaración de estructuras */
struct pac{
        int gcama;
```

```
        char gnom[35];
        char gdom[50];
        int gedad;
        int gpeso;
        }gpac_pac[size];

struct tol{
        int gpcama;
        int gfca;
        int gfcb;
        int gfra;
        int gfrb;
        int gpsa;  /* presion sistolica */
        int gpsb;  /* presion diastolica */
        int gta;
        int gtb;
        }gtol_tol[size];

                        /* Programa principal */
main()
   {
      abrear();  /* abre archivo de pacientes y parametros */

      graphdr=DETECT;                   /* inicializa modo gráfico*/
      initgraph(&graphdr,&graphmo,"c:\\tc");

      pcx("b:ecgdisp4.pcx");      /* dibujo de fondo */

      inicio();      /* inicializa puerto serie */
      proto();       /* envia protocolo */

      while(!kbhit()) /*mientras no se pida cama pide muestra y grafica*/
      {
       for(i=0;i<LONG;i++)
       {
         mues(i);
         }
         param();  /* activa alarmas */
       }

       termina();  /* envia el comando de fin de transmisión */

      pausa();
       scanf("%d",&ca);     /* lee el numero de cama y lo imprime */
       setcolor(BLACK);
```

```
settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
gprintf(338,462,"%d",ca);

switch(ca)      /* envia la clave de la cama deseada */
{
        case 1: CLAVE=CLAVE1;
                clavec=1;
                CHK=0x1f;
                indivi();   /* grafica un paciente */
                break;
        case 2: CLAVE=CLAVE2;
                clavec=2;
                CHK=0x2f;
                indivi();
                break;
        case 3: CLAVE=CLAVE3;
                clavec=3;
                CHK=0x3f;
                indivi();
                break;
        case 4: CLAVE=CLAVE4;
                clavec=4;
                CHK=0x4f;
                indivi();
                break;
        case 5: CLAVE=CLAVE5;
                clavec=5;
                CHK=0x5f;
                indivi();
                break;
        case 6: CLAVE=CLAVE6;
                clavec=6;
                CHK=0x6f;
                indivi();
                break;
        case 7: CLAVE=CLAVE7;
                clavec=7;
                CHK=0x7f;
                indivi();
                break;
        case 8: CLAVE=CLAVE8;
                clavec=8;
                CHK=0x8f;
                indivi();
                break;
```

```
                case 9: CLAVE=CLAVE9;
                        clavec=9;
                        CHK=0x9f;
                        indivi();
                        break;
                case 10: CLAVE=CLAVE10;
                         clavec=10;
                         CHK=0xaf;
                        indivi();
                        break;
                case 11: CLAVE=CLAVE11;
                         clavec=11;
                         CHK=0xbf;
                        indivi();
                        break;
                case 12: CLAVE=CLAVE12;
                         clavec=12;
                         CHK=0xcf;
                        indivi();
                        break;
                case 13: CLAVE=CLAVE13;
                         clavec=13;
                         CHK=0xdf;
                        indivi();
                        break;
                case 14: CLAVE=CLAVE14;
                         clavec=14;
                         CHK=0xef;
                        indivi();
                        break;
                default : break;
                }

        closegraph();   /* cierra el modo gráfico */

}       /*  fin del main  */

            /* Función que abre el archivo de datos de pacientes  */

void abrear(void)
{
        if((fp=fopen("internos","r"))==NULL)
        {
         printf("no puedo abrir el archivo");
         return;
```

```
        }
        for(i=0;i<size;i++)
        {
         fread(&gpac_pac[i],sizeof(struct pac),1,fp);
         }
        fclose(fp);

        if((fc=fopen("tol_par","r"))==NULL)
        {
         printf("no puedo abrir el archivo");
         return;
         }
        for(i=0;i<size;i++)
        {
         fread(&gtol_tol[i],sizeof(struct tol),1,fc);
         }
        fclose(fc);
        }

                    /* función para inicializar el puerto serie  */

void inicio(void)
{
  outp(0x3fb,0x80);  /*direcciona line_control para baudaje*/
  outp(0x3f8,0x0c);  /*LSB para 9600 Bauds*/
  outp(0x3f9,0x00);  /*MSB para 9600 Bauds*/
  outp(0x3fb,0x03);  /*inicializa el line_control en paridad non,
                1 bit stop, 8 bits de datos*/
  outp(0x3fc,0x0b);  /*inicializa contol de modem para interrupciones*/
  outp(0x3f9,0x0f);  /*habilita todas las clases de interrupciones*/

}

            /* función que envia el protocolo para todas las camas  */

void proto(void)
{
  if(gpac_pac[0].gcama==1)        /* envia el protocolo a las camas */
  {                                     /* que estan dadas de alta */
   outp(0x3f8,CLAVE1);
   check_rd_full();
   outp(0x3f8,INI_T);
   check_rd_full();
   }
  if(gpac_pac[1].gcama==2)
```

```
{
 outp(0x3f8,CLAVE2);
 check_rd_full();
 outp(0x3f8,INI_T);
 check_rd_full();
 }
if(gpac_pac[2].gcama==3)
{
 outp(0x3f8,CLAVE3);
 check_rd_full();
 outp(0x3f8,INI_T);
 check_rd_full();
 }
if(gpac_pac[3].gcama==4)
{
 outp(0x3f8,CLAVE4);
 check_rd_full();
 outp(0x3f8,INI_T);
 check_rd_full();
 }
if(gpac_pac[4].gcama==5)
{
 outp(0x3f8,CLAVE5);
 check_rd_full();
 outp(0x3f8,INI_T);
 check_rd_full();
 }
if(gpac_pac[5].gcama==6)
{
 outp(0x3f8,CLAVE6);
 check_rd_full();
 outp(0x3f8,INI_T);
 check_rd_full();
 }
if(gpac_pac[6].gcama==7)
{
 outp(0x3f8,CLAVE7);
 check_rd_full();
 outp(0x3f8,INI_T);
 check_rd_full();
 }
if(gpac_pac[7].gcama==8)
{
 outp(0x3f8,CLAVE8);
 check_rd_full();
```

```
 outp(0x3f8,INI_T);
 check_rd_full();
 }
if(gpac_pac[8].gcama==9)
{
 outp(0x3f8,CLAVE9);
 check_rd_full();
 outp(0x3f8,INI_T);
 check_rd_full();
 }
if(gpac_pac[9].gcama==10)
{
 outp(0x3f8,CLAVE10);
 check_rd_full();
 outp(0x3f8,INI_T);
 check_rd_full();
 }
if(gpac_pac[10].gcama==11)
{
 outp(0x3f8,CLAVE11);
 check_rd_full();
 outp(0x3f8,INI_T);
 check_rd_full();
 }
if(gpac_pac[11].gcama==12)
{
 outp(0x3f8,CLAVE12);
 check_rd_full();
 outp(0x3f8,INI_T);
 check_rd_full();
 }
if(gpac_pac[12].gcama==13)
{
 outp(0x3f8,CLAVE13);
 check_rd_full();
 outp(0x3f8,INI_T);
 check_rd_full();
 }
if(gpac_pac[13].gcama==14)
{
 outp(0x3f8,CLAVE14);
 check_rd_full();
 outp(0x3f8,INI_T);
 check_rd_full();
 }   }
```

```
        /*  función que toma las muestras de ECG de todas las camas  */

void mues(int k)
{
    int j;
    xb1=12;xb2=333;
    yb1=102;yb2=159;yb3=216;yb4=273;yb5=330;yb6=387;yb7=445;
    j=k/4;

     if(gpac_pac[0].gcama==1)
     {
        outp(0x3f8,0x1f);       /* envia chk */
        check_rd_full();        /* checa si el registro de llegada */
        ecgp1[k]=inp(0x3f8);    /* tiene algo y lee el valor */
        putpixel(xb1+j,yb1-(ecgb1[k]/7),BLACK);
        putpixel(xb1+j,yb1-(ecgp1[k]/7),GREEN);
        ecgb1[k]=ecgp1[k];
        }
     if(gpac_pac[1].gcama==2)
     {
        outp(0x3f8,0x2f);
        check_rd_full();
        ecgp2[k]=inp(0x3f8);
        putpixel(xb2+j,yb1-(ecgb2[k]/7),BLACK);
        putpixel(xb2+j,yb1-(ecgp2[k]/7),GREEN);
        ecgb2[k]=ecgp2[k];
        }
     if(gpac_pac[2].gcama==3)
     {
        outp(0x3f8,0x3f);
        check_rd_full();
        ecgp3[k]=inp(0x3f8);
        putpixel(xb1+j,yb2-(ecgb3[k]/7),BLACK);
        putpixel(xb1+j,yb2-(ecgp3[k]/7),GREEN);
        ecgb3[k]=ecgp3[k];
        }
     if(gpac_pac[3].gcama==4)
     {
        outp(0x3f8,0x4f);
        check_rd_full();
        ecgp4[k]=inp(0x3f8);
        putpixel(xb2+j,yb2-(ecgb4[k]/7),BLACK);
        putpixel(xb2+j,yb2-(ecgp4[k]/7),GREEN);
        ecgb4[k]=ecgp4[k];
        }
```

```
if(gpac_pac[4].gcama==5)
{
   outp(0x3f8,0x5f);
   check_rd_full();
   ecgp5[k]=inp(0x3f8);
   putpixel(xb1+j,yb3-(ecgb5[k]/7),BLACK);
   putpixel(xb1+j,yb3-(ecgp5[k]/7),GREEN);
   ecgb5[k]=ecgp5[k];
   }
if(gpac_pac[5].gcama==6)
{
   outp(0x3f8,0x6f);
   check_rd_full();
   ecgp6[k]=inp(0x3f8);
   putpixel(xb2+j,yb3-(ecgb6[k]/7),BLACK);
   putpixel(xb2+j,yb3-(ecgp6[k]/7),GREEN);
   ecgb6[k]=ecgp6[k];
   }
if(gpac_pac[6].gcama==7)
{
   outp(0x3f8,0x7f);
   check_rd_full();
   ecgp7[k]=inp(0x3f8);
   putpixel(xb1+j,yb4-(ecgb7[k]/7),BLACK);
   putpixel(xb1+j,yb4-(ecgp7[k]/7),GREEN);
   ecgb7[k]=ecgp7[k];
   }
if(gpac_pac[7].gcama==8)
{
   outp(0x3f8,0x8f);
   check_rd_full();
   ecgp8[k]=inp(0x3f8);
   putpixel(xb2+j,yb4-(ecgb8[k]/7),BLACK);
   putpixel(xb2+j,yb4-(ecgp8[k]/7),GREEN);
   ecgb8[k]=ecgp8[k];
   }
if(gpac_pac[8].gcama==9)
{
   outp(0x3f8,0x9f);
   check_rd_full();
   ecgp9[k]=inp(0x3f8);
   putpixel(xb1+j,yb5-(ecgb9[k]/7),BLACK);
   putpixel(xb1+j,yb5-(ecgp9[k]/7),GREEN);
   ecgb9[k]=ecgp9[k];
   }
```

```
if(gpac_pac[9].gcama==10)
{
   outp(0x3f8,0xaf);
   check_rd_full();
   ecgp10[k]=inp(0x3f8);
   putpixel(xb2+j,yb5-(ecgb10[k]/7),BLACK);
   putpixel(xb2+j,yb5-(ecgp10[k]/7),GREEN);
   ecgb10[k]=ecgp10[k];
   }
if(gpac_pac[10].gcama==11)
{
   outp(0x3f8,0xbf);
   check_rd_full();
   ecgp11[k]=inp(0x3f8);
   putpixel(xb1+j,yb6-(ecgb11[k]/7),BLACK);
   putpixel(xb1+j,yb6-(ecgp11[k]/7),GREEN);
   ecgb11[k]=ecgp11[k];
   }
if(gpac_pac[11].gcama==12)
{
   outp(0x3f8,0xcf);
   check_rd_full();
   ecgp12[k]=inp(0x3f8);
   putpixel(xb2+j,yb6-(ecgb12[k]/7),BLACK);
   putpixel(xb2+j,yb6-(ecgp12[k]/7),GREEN);
   ecgb12[k]=ecgp12[k];
   }
if(gpac_pac[12].gcama==13)
{
   outp(0x3f8,0xdf);
   check_rd_full();
   ecgp13[k]=inp(0x3f8);
   putpixel(xb1+j,yb7-(ecgb13[k]/7),BLACK);
   putpixel(xb1+j,yb7-(ecgp13[k]/7),GREEN);
   ecgb13[k]=ecgp13[k];
   }
if(gpac_pac[13].gcama==14)
{
   outp(0x3f8,0xef);
   check_rd_full();
   ecgp14[k]=inp(0x3f8);
   putpixel(xb2+j,yb7-(ecgb14[k]/7),BLACK);
   putpixel(xb2+j,yb7-(ecgp14[k]/7),GREEN);
   ecgb14[k]=ecgp14[k];
  }
```

```
}

                /* Función que pide parámetros de todas las camas */

void param(void)
{

    if(gpac_pac[0].gcama==1)
    {
       mparam();
       nucam=0;
       x6a=244;y6a=97;x7a=284;y7a=97;x2a=203;y2a=79;
       x3a=244;y3a=79;x5a=203;y5a=97;
       alarma();
       }
    if(gpac_pac[1].gcama==2)
    {
        mparam();
        nucam=1;
        x6a=566;y6a=97;x7a=606;y7a=97;x2a=524;y2a=79;
        x3a=566;y3a=79;x5a=524;y5a=97;
        alarma();
      }
    if(gpac_pac[2].gcama==3)
    {
        mparam();
        nucam=2;
        x6a=244;y6a=155;x7a=284;y7a=155;x2a=203;y2a=137;
        x3a=244;y3a=137;x5a=203;y5a=155;
        alarma();
      }
    if(gpac_pac[3].gcama==4)
    {
        mparam();
        nucam=3;
        x6a=566;y6a=155;x7a=606;y7a=155;x2a=524;y2a=137;
        x3a=566;y3a=137;x5a=524;y5a=155;
        alarma();
      }
    if(gpac_pac[4].gcama==5)
    {
        mparam();
        nucam=4;
        x6a=244;y6a=213;x7a=284;y7a=213;x2a=203;y2a=193;
```

```
      x3a=244;y3a=193;x5a=203;y5a=213;
      alarma();
    }
 if(gpac_pac[5].gcama==6)
 {
      mparam();
      nucam=5;
      x6a=566;y6a=213;x7a=606;y7a=213;x2a=524;y2a=193;
      x3a=566;y3a=193;x5a=524;y5a=213;
      alarma();
    }
 if(gpac_pac[6].gcama==7)
 {
      mparam();
      nucam=6;
      x6a=244;y6a=270;x7a=284;y7a=270;x2a=203;y2a=250;
      x3a=244;y3a=250;x5a=203;y5a=270;
      alarma();
    }
 if(gpac_pac[7].gcama==8)
 {
      mparam();
      nucam=7;
      x6a=566;y6a=270;x7a=606;y7a=270;x2a=524;y2a=250;
      x3a=566;y3a=250;x5a=524;y5a=270;
      alarma();
    }
 if(gpac_pac[8].gcama==9)
 {
      mparam();
      nucam=8;
      x6a=244;y6a=325;x7a=284;y7a=325;x2a=203;y2a=307;
      x3a=244;y3a=307;x5a=203;y5a=325;
      alarma();
    }
 if(gpac_pac[9].gcama==10)
 {
      mparam();
      nucam=9;
      x6a=566;y6a=325;x7a=606;y7a=325;x2a=524;y2a=307;
      x3a=566;y3a=307;x5a=524;y5a=325;
      alarma();
    }
 if(gpac_pac[10].gcama==11)
 {
```

```
            mparam();
            nucam=10;
            x6a=244;y6a=383;x7a=284;y7a=383;x2a=203;y2a=365;
            x3a=244;y3a=365;x5a=203;y5a=383;
            alarma();
          }
      if(gpac_pac[11].gcama==12)
      {
            mparam();
            nucam=11;
            x6a=566;y6a=383;x7a=606;y7a=383;x2a=524;y2a=365;
            x3a=566;y3a=365;x5a=524;y5a=383;
            alarma();
          }
      if(gpac_pac[12].gcama==13)
      {
            mparam();
            nucam=12;
            x6a=244;y6a=440;x7a=284;y7a=440;x2a=203;y2a=420;
            x3a=244;y3a=420;x5a=203;y5a=440;
            alarma();
          }
      if(gpac_pac[13].gcama==14)
      {
            mparam();
            nucam=13;
            x6a=566;y6a=440;x7a=606;y7a=440;x2a=524;y2a=420;
            x3a=566;y3a=420;x5a=524;y5a=440;
            alarma();
          }
}
void mparam(void)
{
unsigned short int tempe;
          outp(0x3f8,0x26);
          check_rd_full();
          tempe=inp(0x3f8);
          tem=(tempe*0.195312);

          outp(0x3f8,0x27);
          check_rd_full();
          frecc=inp(0x3f8);

          outp(0x3f8,0x28);
          check_rd_full();
```

```
        frecr=inp(0x3f8);

        outp(0x3f8,0x29);
        check_rd_full();
        cs=inp(0x3f8);

        outp(0x3f8,0x30);
        check_rd_full();
        press=inp(0x3f8);

        outp(0x3f8,0x31);
        check_rd_full();
        presd=inp(0x3f8);

}

        /* función que envía un EOT para indicar que termino la adqusición */

void termina(void)
{
 for(i=0;i<10;i++)
 {
  outp(0x3f8,EOT);
  }
 }

                        /* función que despliega un solo paciente  */

void indivi(void)
{
int i;
int x2a,x3a,x5a,x6a,x7a,y2a,y3a,y5a,y6a,y7a;
        pcx("b:ecgdisp5.pcx");
        setcolor(BLACK);
        settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
        gprintf(110,375,"%d",gpac_pac[clavec-1].gcama);
        gprintf(110,420,"%s",gpac_pac[clavec-1].gnom);
        protoin();
        xb=21;
        yb=208;

        while(!kbhit())
        {
        for(i=0;i<599;i++)
         {
```

```
            outp(0x3f8,CHK);
            check_rd_full();
            ecgp[i]=inp(0x3f8);
            putpixel(xb+i,yb-(ecgb[i]/2),BLACK);
            putpixel(xb+i,yb-(ecgp[i]/2),GREEN);
            ecgb[i]=ecgp[i];
            }
            paramin();
           }
           termina();
           pausa();
  }

                /* función que envía el protocolo a una sola cama */

void protoin(void)
{
 outp(0x3f8,CLAVE);
 check_rd_full();
 outp(0x3f8,INI_T);
 check_rd_full();

}

                /* Función que pide parámetros */

void paramin(void)
{
unsigned short int tempe;
        outp(0x3f8,0x26);
        check_rd_full();
        tempe=inp(0x3f8);
        tem=(tempe*0.195312);
        outp(0x3f8,0x27);
        check_rd_full();
        frecc=inp(0x3f8);
        outp(0x3f8,0x28);
        check_rd_full();
        frecr=inp(0x3f8);
        outp(0x3f8,0x29);
        check_rd_full();
        cs=inp(0x3f8);
        outp(0x3f8,0x30);
        check_rd_full();
        press=inp(0x3f8);
```

```
        outp(0x3f8,0x31);
        check_rd_full();
        presd=inp(0x3f8);
        setcolor(LIGHTGRAY);
        settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
        gprintf(268,308,"%d",bfrecr);
        gprintf(268,252,"%d",bfrecc);
        gprintf(117,252,"%d",bpress);
        gprintf(117,308,"%d",bpresd);
        gprintf(415,252,"%-2.1f",btem);
        setcolor(BLACK);
        settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
        gprintf(268,308,"%d",frecr);
        gprintf(268,252,"%d",frecc);
        gprintf(117,252,"%d",press);
        gprintf(117,308,"%d",presd);
        gprintf(415,252,"%-2.1f",tem);
        nucam=clavec-1;
 bfrecc=frecc;bpress=press;btem=tem;
 bfrecr=frecr;bpresd=presd;
 x2a=600;y2a=325;x3a=600;y3a=300;x5a=600;y5a=380;
 x6a=600;y6a=355;x7a=600;y7a=410;
 alarma();
}

            /* función que detecta si el registro de llegada tiene algo */

check_rd_full()
 {
  for(;;)
  {
  if( inp(0x3fd) & 0x01 == 0x01) return;
   if(kbhit()) return;
   }
 }

                        /* función que iniciliza el !kbhit() */

void pausa(void)
{
 int c;
 c=getch();
 if(0==c)
 {
```

```
 c=getch();
 }
}

            /* función que detecta datos anormales y enciende alarmas */

void alarma(void)
{
        if(frecc>gtol_tol[nucam].gfca || frecc<gtol_tol[nucam].gfcb)
          {
            setfillstyle(SOLID_FILL,RED);
            floodfill(x6a,y6a,BLACK);
            sound(300);delay(100);nosound();
          }
        else  setfillstyle(SOLID_FILL,LIGHTGRAY);
              floodfill(x6a,y6a,BLACK);
        if(frecr>gtol_tol[nucam].gfra || frecr<gtol_tol[nucam].gfrb)
          {
            setfillstyle(SOLID_FILL,RED);
            floodfill(x7a,y7a,BLACK);
           sound(300);delay(100);nosound();
          }
        else  setfillstyle(SOLID_FILL,LIGHTGRAY);
              floodfill(x7a,y7a,BLACK);
        if(press>gtol_tol[nucam].gpsa)
          {
            setfillstyle(SOLID_FILL,RED);
            floodfill(x2a,y2a,BLACK);
           sound(300);delay(100);nosound();
          }
        else  setfillstyle(SOLID_FILL,LIGHTGRAY);
              floodfill(x2a,y2a,BLACK);
        if(presd<gtol_tol[nucam].gpsb)
          {
            setfillstyle(SOLID_FILL,RED);
            floodfill(x3a,y3a,BLACK);
           sound(300);delay(100);nosound();
          }
        else  setfillstyle(SOLID_FILL,LIGHTGRAY);
              floodfill(x3a,y3a,BLACK);
        if(tem>gtol_tol[nucam].gta || tem<gtol_tol[nucam].gtb)
          {  setfillstyle(SOLID_FILL,RED);
             floodfill(x5a,y5a,BLACK);
           sound(300);delay(100);nosound();
             }
```

```
        else  setfillstyle(SOLID_FILL,LIGHTGRAY);
              floodfill(x5a,y5a,BLACK);
      }
```

Programa para la impresión de signos vitales.

```
                                /* Archivos de cabecera */
#include <stdio.h>
#include <graphics.h>
#include "c:\len\tc\grafpcx.c"
#include "c:\len\tc\gprintf.c"

                                /* Declaración de constantes */
#define size 14
#define ACKN 0x06
#define EOT 0x04
#define LONG 600
#define INI_T 0x10
#define CLAVE1 0x01
#define CLAVE2 0x02
#define CLAVE3 0x03
#define CLAVE4 0x14
#define CLAVE5 0x05
#define CLAVE6 0x16
#define CLAVE7 0x07
#define CLAVE8 0x08
#define CLAVE9 0x09
#define CLAVE10 0x0a
#define CLAVE11 0x0b
#define CLAVE12 0x0c
#define CLAVE13 0x0d
#define CLAVE14 0x0e

                                /* Declaración de variables */
float tem,temper;
int graphdr,graphmo,i,cama,tipoimp,xb,yb,frecc,freccs,frecr,press,presd,cs;
unsigned short int ACK,CLAVE,CHK,ack,clav;
unsigned short int ecgp[LONG],ecgb[LONG];

int format(double posicion)
{
  int width=6;
  if(posicion<1000.0) width--;
```

```
 if(posicion<100.0) width--;
 if(posicion<10.0) width--;
 return(width);
 }

                        /* Declaración de funciones */

void mues(void),epson(void),laser(void);
void inicio(void);
void param(void);
void proto(void);
FILE*fd;
FILE*fp;

                        /* Declaración de estructuras */
struct dat{
     int camapac;
     int impre;
     }gdat_dat;

struct pac{
  int gcama;
  char gnom[35];
  char gdom[50];
  int gedad;
  int gpeso;
  }gpac_pac[size];


                        /* Programa principal */
main()
   {
        graphdr=DETECT;
        initgraph(&graphdr,&graphmo,"c:\\tc");

       if((fd=fopen("gdat_dat","r"))==NULL)
        {
         puts("no puedo abrir el archivo\n");
         return;
         }

        fread(&gdat_dat,sizeof(struct dat),1,fd);
       fclose(fd);
        if((fp=fopen("internos","r"))==NULL)
        {
```

```
 printf("no puedo abrir el archivo");
 return;
 }
for(i=0;i<size;i++)
{
 fread(&gpac_pac[i],sizeof(struct pac),1,fp);
 }
fclose(fp);

inicio();
cama=gdat_dat.camapac;
switch(cama)
{
 case 0: break;
 case 1: CLAVE=CLAVE1;CHK=0x1f;mues();
         tipoimp=gdat_dat.impre;
          switch(tipoimp)
          {
              case 0: break;
              case 1: epson();
                      break;
              case 2: laser();
                      break;
                      }
         break;
 case 2: CLAVE=CLAVE2;CHK=0x2f;mues();
         tipoimp=gdat_dat.impre;
          switch(tipoimp)
          {
              case 0: break;
              case 1: epson();
                      break;
              case 2: laser();
                      break;
                      }
     break;
 case 3: CLAVE=CLAVE3;CHK=0x3f;mues();
         tipoimp=gdat_dat.impre;
          switch(tipoimp)
          {
              case 0: break;
              case 1: epson();
                      break;
              case 2: laser();
                      break;
```

```
                }

    break;
case 4: CLAVE=CLAVE4;CHK=0x4f;mues();
        tipoimp=gdat_dat.impre;
        switch(tipoimp)
        {
            case 0: break;
            case 1: epson();
                    break;
            case 2: laser();
                    break;
                    }
    break;
case 5: CLAVE=CLAVE5;CHK=0x5f;mues();
        tipoimp=gdat_dat.impre;
        switch(tipoimp)
        {
            case 0: break;
            case 1: epson();
                    break;
            case 2: laser();
                    break;
                    }
    break;
case 6: CLAVE=CLAVE6;CHK=0x6f;mues();
        tipoimp=gdat_dat.impre;
        switch(tipoimp)
        {
            case 0: break;
            case 1: epson();
                    break;
            case 2: laser();
                    break;
                    }
    break;
case 7: CLAVE=CLAVE7;CHK=0x7f;mues();
        tipoimp=gdat_dat.impre;
        switch(tipoimp)
        {
            case 0: break;
            case 1: epson();
                    break;
            case 2: laser();
                    break;
```

```
                }
    break;
case 8: CLAVE=CLAVE8;CHK=0x8f;mues();
        tipoimp=gdat_dat.impre;
        switch(tipoimp)
        {
            case 0: break;
            case 1: epson();
                    break;
            case 2: laser();
                    break;
                    }
    break;
case 9: CLAVE=CLAVE9;CHK=0x9f;mues();
        tipoimp=gdat_dat.impre;
        switch(tipoimp)
        {
            case 0: break;
            case 1: epson();
                    break;
            case 2: laser();
                    break;
                    }
    break;
case 10: CLAVE=CLAVE10;CHK=0xaf;mues();
        tipoimp=gdat_dat.impre;
        switch(tipoimp)
        {
            case 0: break;
            case 1: epson();
                    break;
            case 2: laser();
                    break;
                    }
        break;
case 11: CLAVE=CLAVE11;CHK=0xbf;mues();
        tipoimp=gdat_dat.impre;
        switch(tipoimp)
        {
            case 0: break;
            case 1: epson();
                    break;
            case 2: laser();
                    break;
                    }
```

```
            break;
        case 12: CLAVE=CLAVE12;CHK=0xcf;mues();
                tipoimp=gdat_dat.impre;
                switch(tipoimp)
                {
                    case 0: break;
                    case 1: epson();
                            break;
                    case 2: laser();
                            break;
                            }
            break;
        case 13: CLAVE=CLAVE13;CHK=0xdf;mues();
                tipoimp=gdat_dat.impre;
                switch(tipoimp)
                {
                    case 0: break;
                    case 1: epson();
                            break;
                    case 2: laser();
                            break;
                            }
            break;
        case 14: CLAVE=CLAVE14;CHK=0xef;mues();
                tipoimp=gdat_dat.impre;
                switch(tipoimp)
                {
                    case 0: break;
                    case 1: epson();
                            break;
                    case 2: laser();
                            break;
                            }
                break;
                }

        closegraph();
}


void inicio(void)
{
/*funcion para inicializar el puerto serie*/
 outp(0x3fb,0x80);/*direcciona line_control para baudaje*/
 outp(0x3f8,0x0c);/*LSB para 9600 Bauds*/
```

```
 outp(0x3f9,0x00);/*MSB para 9600 Bauds*/
 outp(0x3fb,0x03);/*inicializa el line_control en paridad non,
                    1 bit stop, 8 bits de datos*/
 outp(0x3fc,0x0b);/*inicializa contol de modem para interrupciones*/
 outp(0x3f9,0x0f);/*habilita todas las clases de interrupciones*/
 }

                /* Función que solicita signos vitales y los grafica */
void mues(void)
{
   int i,xb,yb;
   xb=21;yb=208;

 setcolor(WHITE);
 setlinestyle(SOLID_LINE,0,NORM_WIDTH);
 rectangle(0,0,639,479);
 rectangle(20,10,622,210);
 for(i=0;i<LONG;i++)
 {
  outp(0x3f8,CHK);
  check_rd_full();
  ecgp[i]=inp(0x3f8)/2;
  if(ecgp[i]>0xc8)
  {
   ecgp[i]=0xc8;
   }
      setcolor(WHITE);
      line(xb+i,yb-ecgp[i],xb+i-1,yb-ecgp[i-1]);
  }
  param();
        for(i=0;i<10;i++){
         outp(0x3f8,EOT); }

  }

 check_rd_full()
{
            /*función que detecta si el registro de llegada tiene algo*/
  for(;;){
  if( inp(0x3fd) & 0x01 == 0x01) return;
   if(kbhit()) return;
   }
  }
```

```
void param(void)
{
unsigned short int tempe;

        outp(0x3f8,0x26);
        check_rd_full();
        tempe=inp(0x3f8);
        tem=(tempe*0.195312);
        outp(0x3f8,0x27);
        check_rd_full();
        freccs=inp(0x3f8);
        frecc=freccs+freccs/10;
        outp(0x3f8,0x28);
        check_rd_full();
        frecr=inp(0x3f8);
        outp(0x3f8,0x29);
        check_rd_full();
        outp(0x3f8,0x30);
        check_rd_full();
        press=inp(0x3f8);
        outp(0x3f8,0x31);
        check_rd_full();
        presd=inp(0x3f8);

         setcolor(WHITE);
         settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
         gprintf(260,308,"Frec. Resp. : %d",frecr);
         gprintf(260,252,"Frec. Car. : %d",frecc);
         gprintf(100,252,"Pres. Sist. : %d",press);
         gprintf(100,308,"Pres. Diast. : %d",presd);
         gprintf(420,252,"Temp. : %-2.1f",tem);
         gprintf(100,375,"Cama No.: %d",gpac_pac[cama-1].gcama);
         gprintf(100,420,"Nombre : %s",gpac_pac[cama-1].gnom);

   }

  /* Función que imprime signos vitales en una impresora de matriz de puntos */

void epson(void)
{

  char m;
  int i,j,k,Msb,Lsb,
     MaxX=getmaxx(),
     MaxY=getmaxy();
```

```
  setviewport(0,0,MaxX,MaxY,0);

  fprintf(stdprn,"\x1B%x",31);

  Lsb=MaxY & 0x00FF;
  Msb=MaxY>>8;
  for(j=0;j<=MaxX;j+=8)
  {
   fprintf(stdprn,"\x1B*%c%c%c",0,Lsb,Msb);
   for(i=MaxY;i>=0;i--)
   {
     m=0;
     for(k=0;k<8;k++)
     {
        m<<=1;
        if(getpixel(j+k,i))m++;
        }
     fprintf(stdprn,"%c",m);
     }
   fprintf(stdprn,"\x0D\x0A");
   }
   }

          /* Función que imprime signos vitales en una impresora láser */

void laser(void)
{


  char m,resolucion[3];
  int i,j,k,p,q,xasp,yasp,
     MaxX=getmaxx()+1,
     MaxY=getmaxy()+1;
  static char graph_end[]="\x1B*rB";
  static char graph_ini[]="\x1B*E\x1B&11H\x1B&10\x1B*p0X\x1Bp0Y\x1B*t";
  double xprint, yprint,prstep,AspR;

  getaspectratio(&xasp,&yasp);
  AspR=(double)xasp/(double)yasp;

  setviewport(0,0,MaxX,MaxY,0);

  xprint=690.0,
  yprint=500.0,
```

```
strcpy(resolucion,"100");
prstep=7.2/AspR;

fprintf(stdprn,"%s%sR",graph_ini,resolucion);

for(j=0;j<=MaxY;j++)
{
 fprintf(stdprn,"\x1B&a%-*.1fh%-*.1fv",format(xprint),xprint,
         format(yprint),yprint);
 yprint +=prstep;
 fprintf(stdprn,"\x1B*r1A\x1B*b%dW",MaxX/8);

 for(i=0;i<MaxX/8;i++)
 {
  m=0;
  for(k=0;k<8;k++)
  {
     m<<=1;
     if(getpixel(i*8+k,j))m++;
     }
  fprintf(stdprn,"%c",m);
  }
 fprintf(stdprn,"%s",graph_end);
 }
 fprintf(stdprn,"\x0C\x1B&10\x1B(8U\x1B(sp10h12vsb3T\x1B&11H");
 }
```