



D-19980



T
610.28
PLA
P.2

ESCUELA SUPERIOR POLITECNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

TRABAJO DE GRADUACIÓN

"SIMULADOR CARDIACO DIGITAL"

Previo a la obtención del Título de:

**INGENIERO EN ELECTRICIDAD
ESPECIALIZACION ELECTRONICA**

Presentado por:

**VICENTE XAVIER PLAZA GUINGLA
MANUEL EDUARDO SANTIN LOAYZA**

GUAYAQUIL- ECUADOR

**AÑO
1999**

AGRADECIMIENTO

**A nuestros padres que nos dieron
incondicionalmente su apoyo.
A Denise y a Pedro, sin ellos no hubiéramos
culminado nuestro trabajo.**

DEDICATORIA

Dedico el presente trabajo a mi madre
que siempre me apoya.

A mi esposa por su comprensión y
sacrificio.

A mis hijos que me motivan.

A mi padre.

Manuel

A Dios.

A mis padres Elena y Manuel.

A mi hermano Douglas.

A Denise.

Vicente

DECLARACIÓN EXPRESA

“La responsabilidad del contenido de este Tópico de Grado, nos corresponden exclusivamente; y el patrimonio intelectual de la misma a la ESCUELA SUPERIOR POLITECNICA DEL LITORAL”



Vicente Plaza Guingla



Manuel SantIn Loayza

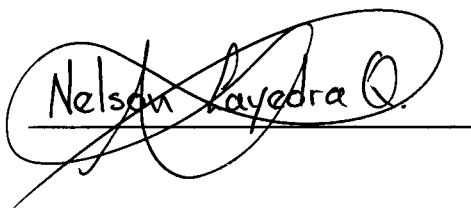
TRIBUNAL DE GRADUACION



**ING. CARLOS MONSALVE A.
SUBDECANO DE LA FIEC**



**ING. MIGUEL YAPUR
DIRECTOR DE TOPICO**



**ING. NELSON LAYEDRA
VOCAL**



**ING. WILMER NARANJO
VOCAL**

TRIBUNAL DE GRADUACION

**ING. SARA RIOS
VOCAL**

RESUMEN

El simulador cardíaco permite generar las mismas señales que se obtienen cuando un individuo se realiza un electrocardiograma.

Lo primero que se realizó fue digitalizar la señal cardíaca para esto se construyó un circuito adicional para convertir la onda cardíaca que es una señal analógica a una señal digital y se la almacena en una EPROM. Además se utilizó un microcontrolador que es el cerebro de todo el simulador, ya que tiene guardado dentro de él, las funciones que debe realizar.

El microcontrolador permite acceder a la memoria EPROM donde se encuentran los datos ya digitalizados de la señal cardíaca, y a la vez estos datos digitales pasan a un convertidor digital analógico que convierte la señal digital a analógica, donde podrá ser observada en un osciloscopio o en un monitor cardíaco.

Para cambiar la frecuencia cardíaca se utiliza un convertidor analógico a digital y un potenciómetro, que están conectados con el microcontrolador, como ya lo anotamos el microcontrolador es el cerebro del dispositivo.

INDICE GENERAL

	Pag.
RESUMEN	VI
INDICE GENERAL	VII
ABREVIATURAS	XII
SIMBOLOS.....	XIV
INDICE DE FIGURAS	XV
INDICE DE TABLAS	XVIII
INDICE DE ANEXOS.....	XIX
1. GENERALIDADES	2
1.1 EL SISTEMA CIRCULATORIO.....	2
1.2 POTENCIALES DE ACCION.....	5
1.3 CICLO CARDIACO.....	8
1.4 ELECTROCARDIOGRAMA.....	10
2. ANALISIS POR BLOQUE.....	14
2.1 EL MICROCONTROLADOR.....	14
2.2 LA MEMORIA	18
2.3 EL CONVERTIDOR DIGITAL A ANALOGICO.....	19
2.3.1 RESOLUCION.....	21
2.3.2 PRECISION.....	21
2.3.3 ERROR DE DESPLAZAMIENTO (OFFSET).....	23

2.4 EL AMPLIFICADOR.....	24
3. DESCRIPCION DEL FUNCIONAMIENTO DEL CAPTURADOR DE SEÑAL CARDIACA	25
3.2 DATOS TEORICOS AFINES.....	25
3.2.1 EL AMPLIFICADOR OPERACIONAL.....	26
3.2.2 EL AMPLIFICADOR DE PROPOSITO GENERAL μ A741.....	26
3.2.2.1 SIMBOLO Y TERMINALES DEL CIRCUITO.....	26
3.2.3 EL AMPLIFICADOR DIFERENCIAL BASICO.....	27
3.2.3.1 VOLTAJE EN MODO COMUN.....	29
3.2.4 MEJORAS EN EL AMPLIFICADOR DIFERENCIAL BASICO	31
3.2.4.1 INCREMENTO DE LA RESISTENCIA DE ENTRADA.....	31
3.2.4.2 GANANCIA AJUSTABLE.....	32
3.2.5 EL AMPLIFICADOR DE INSTRUMENTACION.....	34
3.2.5.1 OPERACIÓN DEL CIRCUITO.....	34
3.2.6 REGULADORES DE TRES TERMINALES.....	36
3.2.6.1 INTRODUCCION.....	36
3.2.6.2. CARACTERISTICAS.....	37
3.2.7 EL CONVERTIDOR ANALOGICO A DIGITAL.....	38
3.2.7.1. DESCRIPCION GENERAL DEL ADC0804.....	38
3.2.7.2. CUALIDADES.....	39

3.2.7.3. ESPECIFICACIONES.....	39
3.2.8 CONVERSION ANALOGICA A DIGITAL.....	40
3.2.8.1 TECNICA DE APROXIMACIONES SUCESIVAS.....	40
3.2.8.2 OPERACIÓN DEL CIRCUITO.....	41
3.2.8.3 ANALOGIA POR APROXIMACIONES SUCESIVAS	42
3.2.8.4 TIEMPO DE CONVERSION.....	44
3.2.9 EL BUFFER DE ACOPLAMIENTO 74LS244.....	44
3.2.9.1 DESCRIPCION GENERAL.....	44
3.2.9.2 CARACTERISTICAS.....	45
3.3 DESCRIPCION DEL FUNCIONAMIENTO DEL CSC.....	46
3.3.1 LA ADQUISICIÓN DE DATOS.....	46
3.3.2 LA AMPLIFICACION.....	46
3.3.3 LA CONVERSION ANALOGICA A DIGITAL.....	46
3.3.4 EL ACOPLAMIENTO.....	47
3.3.5 EL FILTRADO DE LA SEÑAL.....	48
3.3.6 LA FUENTE DE ALIMENTACION.....	48
3.4 DIAGRAMA ESQUEMATICO DEL CSC.....	49
4. ANALISIS DEL CIRCUITO PARA EL SIMULADOR CARDIACO.....	51
4.1 CARACTERISTICAS DEL MICROCONTROLADOR 8X51.....	52
4.1.1 ORGANIZACIÓN DE LA MEMORIA.....	53
4.1.1.1 MEMORIA DE PROGRAMA.....	55
4.1.1.2 CICLO DE MAQUINA.....	60

4.1.1.3 MEMORIA DE DATOS.....	61
4.1.1.3.1 BANCO DE REGISTROS.....	64
4.1.1.3.2 AREA DIRECCIONABLE DE BIT A BIT.....	65
4.1.1.3.3 AREA DE RAM DEL USUARIO.....	65
4.1.1.4 REGISTRO NO DIRECCIONABLE DE BIT A BIT.....	68
4.1.1.5 REGISTROS EXCLUSIVOS DE LOS 8XX2.....	69
4.1.2 TEMPORIZADOR / CONTADORES.....	73
4.1.3 COMUNICACIONES.....	73
4.2 CONEXIONES.....	75
4.2.1 CONEXION DEL MICROCONTROLADOR.....	75
4.2.2 CONEXION DE LA MEMORIA.....	77
4.2.3 CONEXIÓN DEL CONVERTIDOR DIGITAL ANALOGICO	78
5. MANUAL DEL USUARIO.....	79
5.1 MANUAL DEL USUARIO DEL CSC.....	79
5.1.1 GUIA GENERAL.....	79
5.1.2 FUNCION DE LAS BOTONERAS.....	80
5.1.3 REQUERIMIENTOS.....	81
5.1.4 MANEJO DEL CSC (HARDWARE).....	82
5.1.5 PRESENTACION (SOFTWARE).....	82
5.1.5.1 INTRODUCCION.....	82
5.1.5.2 FIJAR PARAMETROS DE MUESTREO.....	84
5.1.5.3 MUESTREO DE LA SEÑAL.....	87

5.1.5.4 CONGELAMIENTO DE LA SEÑAL.....	88
5.1.5.5 GRABACION DE LA SEÑAL.....	88
5.1.5.6 BUSQUEDA DE ARCHIVOS.....	91
5.1.5.7 PARA IMPRIMIR UNA SEÑAL.....	93
5.2 MANUAL DEL USUARIO DEL SIMULADOR CARDIACO.....	97
6. CONCLUSIONES Y RECOMENDACIONES.....	100
6.1 CONCLUSIONES DEL CAPTURADOR DE SEÑAL CARDIACA...	100
6.2 CONCLUSIONES DE SIMULADOR CARDIACO.....	101
ANEXOS.....	103
BIBLIOGRAFIA.....	170

ABREVIATURAS

μ C	Microcontrolador
μ P	Microprocesador
RAM	Memoria de acceso aleatorio
ROM	Memoria de solo lectura
PROM	Memoria ROM programable
EPROM	Memoria PROM borrable eléctricamente
E/S o I/O	Dispositivos de entrada y salida
D/A	Conversión digital a analógica
DAC	Convertidor digital a analógico
A/D	Conversión analógica a digital
ADC	Convertidor analógico a digital
INTEL	Empresa dedicada a la fabricación de circuitos integrados.
CPU	Unidad de procesamiento central
MCS-51	Familia de microcontroladores de INTEL
P.A.	Potencial de acción
S.A.	Sino Atrial
ECG	Electrocardiograma
P	Onda P, perteneciente a la señal cardíaca
QRS	Onda QRS, perteneciente a la señal cardíaca

T	Onda T, perteneciente a la señal cardíaca
CSC	Capturador de señal cardíaca
LSB	Bit menos significativo
MSB	Bit mas significativo
RA	Brazo derecho
LA	Brazo izquierdo
C	Pecho
RL	Pierna derecha
LL	Pierna izquierda
PC	Computadora personal
SAR	Registro de aproximaciones sucesivas
R	Resistencia
T	Período
RL	Resistencia de carga
Tc	Período de conversión
Vo	Voltaje de salida
Vsal	Voltaje de salida
Ven	Voltaje de entrada
μ s	microsegundos
ns	nanosegundos
E	Señal entrante
A	Amplificador operacional

SIMBOLOGIA

A AMPERIOS

Ω OHMIOS

μ F micro faradios

s segundos

V Voltios

$^{\circ}$ C grados centigrados

INDICE DE FIGURAS

	Pag.
Figura 1.1. REPOSO	5
Figura 1.2. DESPOLARIZACION	5
Figura 1.3. REPOLARIZACION	6
Figura 1.4. EL CORAZON	8
Figura 1.5. ONDA CARDIACA	10
Figura 2.1. AMPLIFICADOR INVERSOR	20
Figura 3.1. SIMBOLO Y TERMINALES DE UN OPAMP.....	27
Figura 3.2. AMPLIFICADOR DIFERENCIAL BASICO.....	28
Figura 3.3. GANANCIA DEL VOLTAJE EN MODO COMUN.....	30
Figura 3.5. ENTRADA DIFERENCIAL AL AMPLIFICADOR DE 32 SALIDA DIFERENCIAL CON GANANCIA AJUSTABLE	32
Figura 3.6. AMPLIFICADOR DE INSTRUMENTACION.....	35
Figura 3.7. DIAGRAMA ESQUEMATICO DE UN REGULADOR DE 36 TRES TERMINALES	36
Figura 3.8. DIAGRAMA DE BLOQUES DE UN ADC POR 41 APROXIMACIONES SUCESIVAS	41
Figura 3.9. DIAGRAMA ESQUEMATICO DE A/D	43
Figura 3.10. DIAGRAMA DE BLOQUES DEL C.S.C.	45
Figura 3.11. DIAGRAMA ESQUEMATICO DEL C.S.C.	49

Figura 3.12	DIAGRAMA ESQUEMATICO DE ALIMENTACIÓN DEL C.S.C.	50
Figura 4.1	ORGANIZACIÓN DE LAS MEMORIAS.....	54
Figura 4.2	ESQUEMA DE UNA INTERRUPCION.....	56
Figura 4.3	MEMORIA DE PROGRAMA.....	58
Figura 4.4	OPERACIÓN CON PROGRAMA EXTERNO.....	59
Figura 4.5	CICLO DE MAQUINA.....	61
Figura 4.6	MICROCONTROLADOR CON MEMORIA EXTERNA DE DATOS	62
Figura 4.7	ESPACIO DE MEMORIA RAM INTERNA Y FORMA DE DIRECCIONAMIENTO	62
Figura 4.8	LOS 128 BAJOS.....	64
Figura 4.9	SISTEMA DE ADQUISICION DE DATOS.....	74
Figura 5.1	VISTA FRONTAL DEL C.S.C.....	79
Figura 5.2	VISTA POSTERIOR DEL C.S.C.....	80
Figura 5.3	LA PANTALLA DEL EXPLORADOR DE WINDOWS.....	83
Figura 5.4	ABRIENDO ACCESO DIRECTO AL ADC0804.EXE.....	84
Figura 5.5	PANTALLA DEL MENU PRINCIPAL DE ADC0804.....	85
Figura 5.6	ABRIENDO PERIODO DE MUESTREO PARA CAMBIAR EL PARAMETRO	86
Figura 5.7	MUESTREO DE LA SEÑAL DEL SIMULADOR CARDIACO DIGITAL	87

Figura 5.8	NOMBRANDO UN ARCHIVO QUE SE HA GRABADO.....	89
Figura 5.9	AVISO DE QUE NO SE HA NOMBRADO EL ARCHIVO GRABADO	90
Figura 5.10	BUSCANDO EL ARCHIVO JUAN CARLOS MORENO.....	91
Figura 5.11	RESULTADO DE LA BUSQUEDA DEL ARCHIVO JUAN CARLOS MORENO	92
Figura 5.12	BUSQUEDA DE ARCHIVO EN EXCEL.....	94
Figura 5.13	LA SEÑAL CARDIACA GUARDADA VISTA EN EXCEL LISTA PARA SER IMPRESA	95
Figura 5.14	PANTALLA AL SALIR DE EXCEL.....	96

INDICE DE TABLAS

	Pag.
Tabla 1.1 VALORES DE AMPLITUD Y DURACION	12
Tabla 4.1 MIEMBROS DE LA FAMILIA MCS-51	52
Tabla 4.2 AREA DE REGISTRO DE PROPOSITO ESPECIAL SFR.....	66

INDICE DE ANEXOS

	Pag.
Anexo 1 CIRCULACION DE LA SANGRE	104
Anexo 2 SIMULADOR CARDIACO	105
Anexo 3 CODIGO FUENTE DEL PROGRAMA QUE CAPTURA LA SEÑAL CARDIACA	106
Anexo 4 DIAGRAMA DE BLOQUES.....	149
Anexo 5 CONFIGURACION BASICA DEL MICROCONTROLADOR..	150
Anexo 6 DIAGRAMA DE FLUJO.....	151
Anexo 7 PROCEDIMIENTO DELAY1	152
Anexo 8 PROCEDIMIENTO ENTRADAS	153
Anexo 9 PROCEDIMIENTO DELAY.....	154
Anexo 10 DIAGRAMAS ESQUEMATICOS DEL SIMULADIR.....	155
Anexo 11 CIRCUITO IMPRESO DEL C.S.C.....	159
Anexo 12 CIRCUITO IMPRESO DEL SIMULADOR	160
Anexo 13 CODIGO PARA PROGRAMAR EL MICROCONTROLADOR	161
Anexo 14 CODIGO HEXADECIMAL DEL PROGRAMA	164
Anexo 15 PROCEDIMIENTO ENTRADAS	79
Anexo 16 PROCEDIMIENTO DELAY.....	80
Anexo 17 CODIGO PARA PROGRAMAR EL MICROCONTROLADOR	100
Anexo 18 CODIGO HEXADECIMAL DEL PROGRAMA	164

Anexo 19	DATOS DIGITALES ALMACENADOS EN LA EPROM	165
Anexo 20	LISTA DE COMPONENTES	166
Anexo 21	LISTA DE PRECIOS	168

INTRODUCCIÓN

Nuestro corazón es uno de los órganos más importantes de nuestro cuerpo, y con el simulador tendremos la oportunidad de aprender un poco más acerca de su comportamiento.

El presente trabajo ha sido diseñado con la finalidad de proporcionar un dispositivo que permita simular la señal cardíaca de un individuo y variar la frecuencia de la misma, de manera que se ponga en práctica los conocimientos adquiridos al estudiar esta señal, y las alteraciones que se pueden presentar.

Este dispositivo podrá ser utilizado para comprobar el correcto funcionamiento de electrocardiografos, ya que permite determinar si la señal cardíaca simulada de la persona, tiene un ritmo cardiaco normal (60 latidos por minutos), si tiene bradicardia (menos de 60 latidos por minutos) , o taquicardia (más de 90 latidos por minutos)

Además permite fijar los limites de las pulsaciones cardiacas permisibles, por la persona (con las indicaciones medicas); es decir que la referencia puede ser incrementada a medida que el paciente (simulador) va restableciéndose en su rehabilitación.

CAPITULO I

1. GENERALIDADES

1.1 EL SISTEMA CIRCULATORIO

El corazón es un músculo hueco situado entre los pulmones. Se divide en dos partes la derecha contiene sangre venosa y a la izquierda, que alberga la arterial. Cada una se subdivide a su vez, en dos cavidades, superpuestas y llamadas aurícula y ventrículo respectivamente. El corazón del adulto pesa unos 275 gramos.

Los ventrículos están separados de sus articulaciones respectivas por un orificio; el paso a través del mismo lo regula una válvula. La que separa en el ventrículo derecho de la aurícula del mismo lado se denomina tricúspide, mientras que la del lado opuesto se llama mitral. Los ventrículos presentan además, otro orificio, mediante el cual comunican con las grandes arterias, la

aorta en el lado izquierdo y la pulmonar en el lado derecho. Dichos orificios también poseen sus válvula: la aórtica y la pulmonar, respectivamente.

Las aurículas son unas cavidades más pequeñas, situadas encima de cada ventrículo y comunicadas con ellos. La función de las aurículas es principalmente de entrada a los ventrículos, pero también impulsan debidamente la sangre para desplazarla desde las aurículas hacia los ventrículos.

La aurícula derecha presenta, además de la válvula de comunicación con el ventrículo, otros dos orificios donde desembocan las grandes venas que retornan la sangre hacia el corazón: son las venas cavas: superior e inferior, respectivamente. La aurícula izquierda posee cuatro orificios destinados a recibir, mediante las venas pulmonares, procedentes dos de cada pulmón, la sangre ya oxigenada.

Las arterias siempre llevan sangre del corazón hacia el cuerpo; y las venas del cuerpo al corazón. La arteria pulmonar, a pesar de llamarse arteria, lleva sangre venosa del ventrículo derecho hacia el pulmón, y las venas pulmonares devuelven la sangre oxigenada hacia la aurícula izquierda. El corazón también necesita un aporte especial de sangre que lleva sustancias nutritivas hasta su

última célula, hasta su última fibra muscular. Las arterias encargadas de esta misión se llaman coronarias, pues forman una verdadera corona colocada en su superficie externa.

Al circuito que nace en el ventrículo derecho y termina en la aurícula izquierda se llama circuito pulmonar o menor. Al que nace en el ventrículo izquierdo, y termina en la aurícula derecha, se lo llama circuito aórtico o mayor.

Los mecanismos especiales del corazón conservan el ritmo cardíaco y transmiten los potenciales de acción a toda la musculatura del órgano para iniciar su contracción: dichos mecanismos al contrario que los músculos ventriculares y auriculares que se contraen fuertemente, solo se contraen débilmente, ya que contienen muy pocas fibras contráctiles brindando un sistema excitatorio y de transmisión para la rápida conducción de impulsos a través del corazón.

1.2 POTENCIALES DE ACCIÓN

Los potenciales de acción son cambios repentinos de tipo pursátil del potencial de la membrana, que duran unas cuantas milésimas de segundo.

El potencial de acción puede ser despertado en una fibra nerviosa casi por cualquier factor que aumenta repentinamente la permeabilidad de la membrana a los iones de sodio.

Los potenciales de acción se presentan en dos etapas separadas :DESPOLARIZACIÓN Y REPOLARIZACIÓN

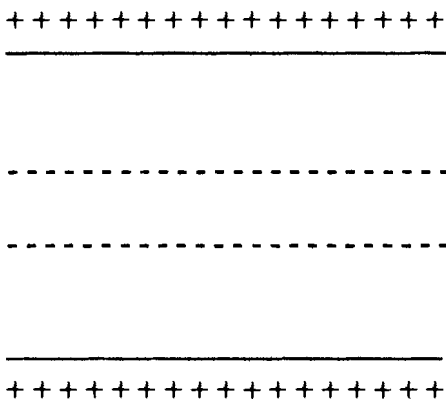


Fig 1.1
REPOSO

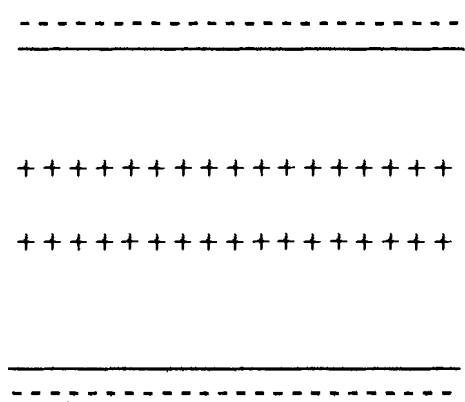


Fig 1.2
DESPOLARIZACIÓN

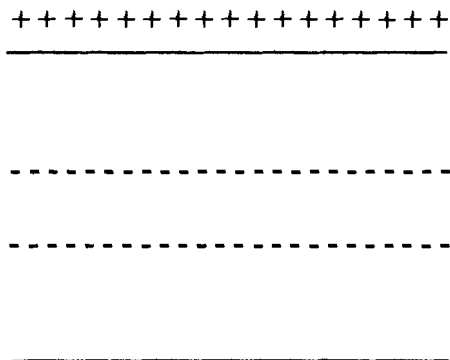


Fig 1.3

REPOLARIZACIÓN

En la (figura 1.1) se indica el estado de reposo de la membrana. Cuando la permeabilidad de la membrana para los iones de sodio aumenta bruscamente, muchos iones de sodio penetran en el interior de la fibra, llevando consigo suficientes cargas positivas, para causar la desaparición total del potencial normal de reposo y generalmente bastantes cargas para generar un estado positivo dentro de la fibra, este proceso es conocido como **DESPOLARIZACIÓN** (figura 1.2).

El potencial positivo que se desarrolla momentáneamente dentro de la fibra, se llama **POTENCIAL DE INVERSION**. Casi inmediatamente después que se lleva a cabo la despolarización, los poros de la membrana se hacen impermeables a los iones de sodio, pero al mismo tiempo, mucho más permeables de lo normal a los de potasio; por lo tanto los iones de sodio ya no pasan al interior de la fibra, en lugar de ellos salen los de potasio por su concentración alta en su interior. En consecuencia, como los iones de potasio tienen un exceso de cargas positivas dentro de la fibra, estos son transportados nuevamente al exterior para así recuperar su potencial de reposo; a este proceso se lo denomina **REPOLARIZACIÓN** (figura 1.3).

El músculo cardiaco tiene un tipo peculiar de P. A.; después de la espiga inicial, la membrana se conserva despolarizada durante 0.15 a 0.3 segundos, formando una meseta, seguida al término de la misma de una brusca repolarización, la presencia de esta meseta en el P.A. Hace que este dure de 20 a 50 veces más en el músculo cardiaco que en el músculo esquelético, y también origina un periodo de contracción bastante prolongado.

1.3 CICLO CARDIACO

El periodo que va desde el final de una contracción cardíaca hasta el final de una contracción siguiente se denomina CICLO CARDIACO.

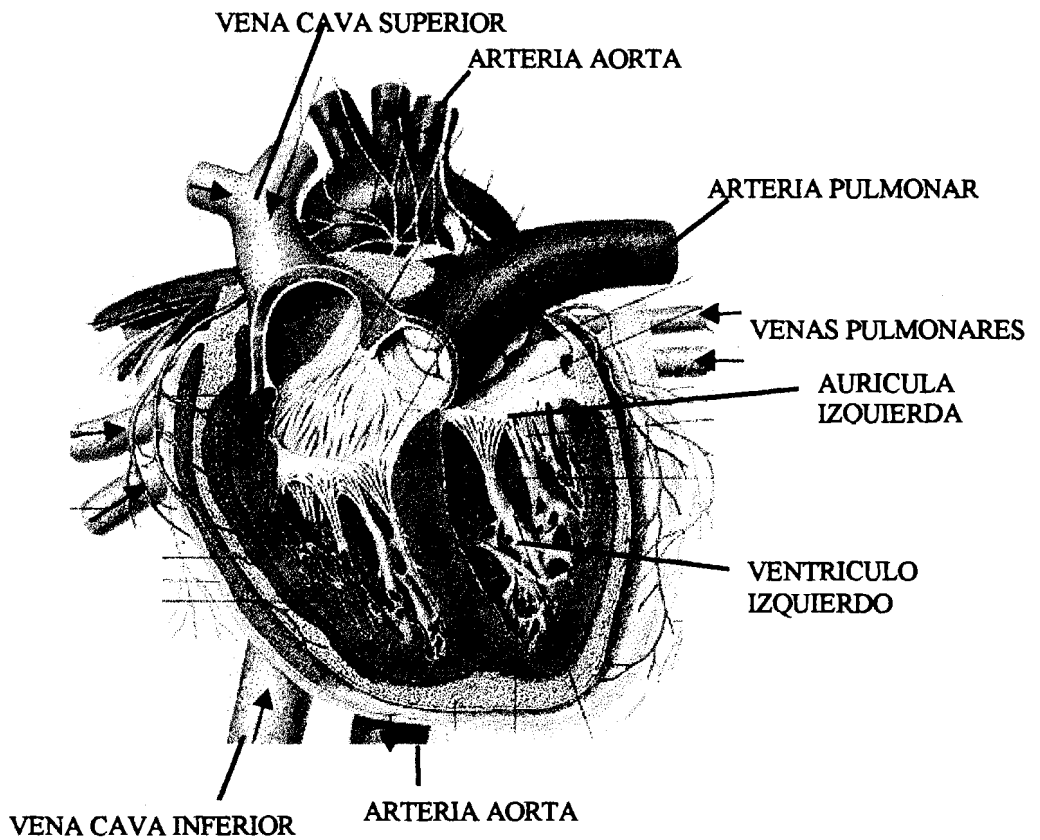


Fig. 1.4 EL CORAZÓN

Cada ciclo se inicia por lo general espontánea de un potencial de acción en el NODO SINO ATRIAL (NODO S.A.), el cual es el marcapasos natural del corazón y produce una señal eléctrica periódica de 72 ciclos por minuto normalmente en una persona adulta.

Este nodo se encuentra localizado en la parte posterior de la aurícula derecha, cerca de la abertura de la vena cava superior; el potencial viaja rápidamente por ambas aurículas hacia los ventrículos, hay un retraso de más de 0.1 seg entre el paso del impulso cardíaco a través de las aurículas y luego a través de los ventrículos; esto permite que las aurículas se contraigan antes que los ventrículos, con lo cual le impulsa sangre hacia los ventrículos antes de producirse la contracción ventricular enérgica.

Así las aurículas actúan como bombas de cebamiento para los ventrículos y estos luego proporcionan la fuerza mayor para desplazar la sangre por todo el sistema vascular.

El ciclo cardíaco incluye un período de relajación denominado **DIASTOLE** seguido de contracción llamado **SISTOLE**

1.4 EL ELECTROCARDIOGRAMA

Si las mediciones de potencial eléctrico entre el brazo derecho y el izquierdo es amplificado, la señal del ECG tendrá la apariencia de la figura

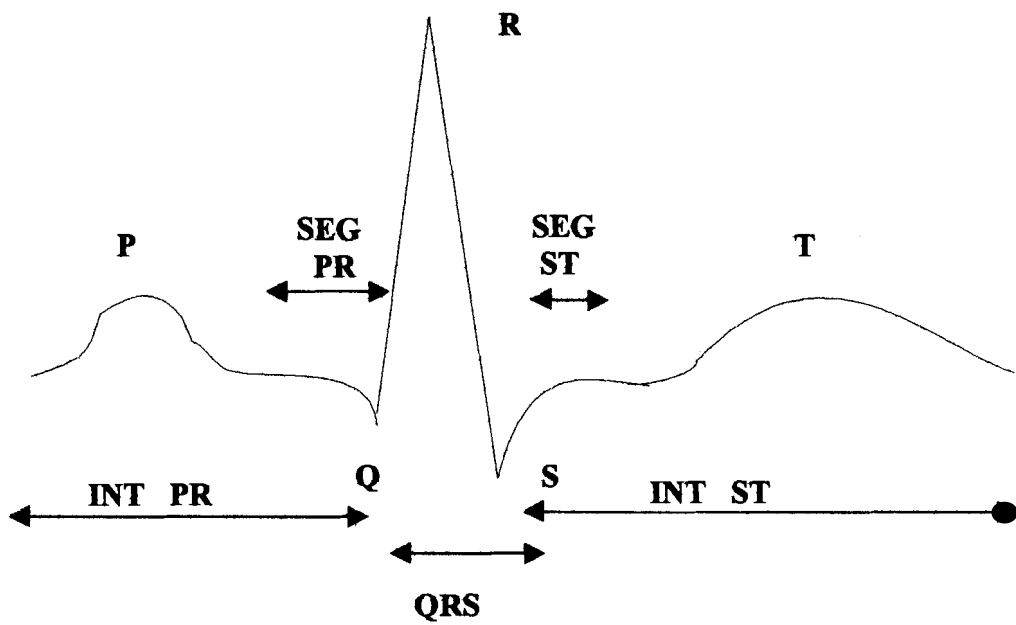


Fig. 1.5 ONDA CARDIACA

Cada una de las tres ondas completas P, QRS, y T corresponden a un evento electrofisiológico particular. El inicio del ciclo eléctrico toma lugar debido al nodo sino atrial (S.A), que es el marcapasos natural del corazón.

Se trata de un gráfico de las variaciones de voltaje del corazón en función del tiempo registrado por el electrocardiógrafo en la superficie corporal.

La onda P es causada por la disminución de la despolarización en la aurícula y va seguida por la contracción de esta cámara; ellos causan un ligero aumento en la curva de presión auricular inmediatamente después de la onda P.

Aproximadamente después de 0.16 segundos de iniciada la onda P aparecen las ondas QRS a consecuencia de la despolarización de los ventrículos, que inicia la contracción de los mismos y hace que su impresión en su interior empiece a elevarse según indica la figura por lo tanto el complejo QRS empieza poco antes de iniciarse la sístole ventricular.

Finalmente tenemos la onda T ventricular del electrocardiograma, que representa la etapa de repolarización de los ventrículos, cuando las fibras musculares correspondientes empiezan a relajarse, por lo tanto, la onda T ocurre antes de terminar la relajación ventricular.

Algunos valores normales para las amplitudes y duración de parámetros del ECG son los siguientes:



TABLA 1.1 VALORES DE AMPLITUD Y DURACION

El corazón humano adulto normalmente se contrae rítmicamente unas 72 veces por minutos, pero por diversos factores se producen ritmos cardíacos anormales.

Las variaciones cardiacas se las denominan arritmias que en términos generales es la pérdida del ritmo cardíaco. Las arritmias pueden ser de dos tipos. **TAQUICARDIA** significa frecuencia cardíaca rápida en principio mayor a 100 latidos por minuto, entre las causas generales de taquicardia, tenemos el aumento corporal, o estados tóxicos del corazón.

La frecuencia cardíaca aumenta aproximadamente 18 latidos por minuto por cada grado centígrado de aumento de la temperatura corporal hasta llegar aproximadamente a 41°C mas allá de esta temperatura, la frecuencia cardíaca puede en realidad disminuir por debilitación creciente del músculo a consecuencia de la fiebre.

BRADICARDIA que significa simplemente frecuencia cardíaca lenta, en principio con menos de 60 latidos por minuto.

CAPITULO II

ANALISIS POR BLOQUES

El simulador consta de cuatro partes fundamentales como se observa en el anexo diagrama de bloques:

- El microcontrolador
- La memoria
- El convertidor digital analógico
- El amplificador

2.1 El Microcontrolador (μ C).

Un microcontrolador μ C es una microcomputadora de pastilla porque contiene la mayoría de los elementos funcionales de una computadora, inclusive el procesador central, la memoria de acceso aleatorio (RAM), la memoria solo de lectura (ROM), y puertos de entrada y salida (E / S).

Aunque los microprocesadores y los microcontroladores tienen origen común, están diseñados para aplicaciones diferentes. La mayoría de los

microcontroladores se emplean en aplicaciones de intenso Hardware, de tiempo real, en las que se tienen señales tanto digitales como analógicas. Estas varían desde el control de codificación por teclado hasta el de motores, procesos industriales y motores de automóvil.

En contraste , el microprocesador (μP) que tiene mejoras continuas, se usa principalmente en aplicaciones de software intenso que se encuentran en computadoras personales, estaciones de trabajo para gráfico y en computadoras paralelas de nueva generación.

El diagrama simplificado en bloques ilustra un microcontrolador de una sola pastilla, de 8 bits, que contiene una unidad central de procesamiento (CPU) y memoria de datos RAM. Sin embargo también tienen funciones que no se encuentran en los microprocesadores, como memoria para programar (ROM o EPROM), reloj, contador de tiempo y eventos, y puertos programables en paralelo y en serie de E/S. Las ventajas que ofrecen el uso del microcontrolador en donde es apropiado incluyen ahorros en memoria externa y circuitos integrados de apoyo, espacio para tarjetas de circuito impreso y consumo de potencia.

El microcontrolador ocupa una pastilla de silicio que es aproximadamente del mismo tamaño que la de un microprocesador, por lo

que se tienen que aceptar situaciones de compromiso para acomodar las funciones adicionales. Se imponen, por ejemplo, límites de tamaño en las memorias RAM y ROM en pastilla. No obstante, algunos dispositivos μC avanzados pueden realizar funciones que requerían un microprocesador de generación anterior y cinco dispositivos adicionales de apoyo.

Los microcontroladores se obtienen con arquitectura de 4, 8 y 16 bits, con su funcionamiento relacionado en general con la longitud de bit. Se ofrecen en familias para dar al usuario opciones de rendimiento y de funciones. Por ejemplo los μC pueden tener diferentes cantidades de RAM y ROM en pastillas para facilitar la selección de acoplamiento de costo adecuado para u requisitos de aplicación. La parte principal de cada familia tiene una memoria ROM programada en fábrica. Una unidad μC de 8 bits puede ofrecer 1000 bytes de memoria ROM interna programada en fábrica y solo 64 bytes de memoria RAM interna, pero otro puede ofrecer 2000 bytes de ROM interna y 128 bytes de RAM interna.

Las primera unidades de μC con tecnología NMOS. Algunos diseños se han convertido a CMOS y se están introduciendo nuevos dispositivos en esa tecnología. Ofrecen menor consumo de potencia importante en aplicaciones de control. Las unidades μC han seguido los mismos pasos de

desarrollo que las μP , y se han beneficiado con los adelantos técnicos logrados en esos dispositivos.

Para desarrollo de productos se obtienen unidades μC sin ROM. El diseñador puede querer desarrollar software en memorias EPROM o EEPROM fuera de pastilla, para permitir cambios de programas más rápidos y económicos. Cuando se obtiene un programa depurado de errores, el usuario puede pedir unidades μC convencionales en cantidad, con memorias ROM programadas de fábrica.

En forma alterna algunos fabricantes de unidades μC , está, ofreciéndolas con EPROM y EEPROM en la pastilla, porque algunos clientes podrían no tener nunca la capacidad para hacer pedidos en volumen suficiente para justificar la compra de microcontroladores programados en fábrica. Se obtienen ya algunas unidades μC de 8 bits con convertidores analógico / digital. Esta opción permite conectar con interfaz directa a transductores analógicos tales como sensores de presión, temperatura y movimiento. Los microcontroladores de 8 bits con puerto en serie de E / S en la pastilla, permiten que trabajen juntas dos o más unidades μC en el sistema,. Algunas unidades μC de 16 bits y una sola pastilla tienen ambas cosas, puerto en serie de E / S y conversión analógica / digital (10 bits) de alta resolución.

2.2 La Memoria

Una EPROM puede ser programada por el usuario y también puede borrarse y reprogramarse tantas veces como se desee, una vez programada, la EPROM es una memoria no volátil que contendrá sus datos almacenados indefinidamente. El proceso para programar la EPROM implica la aplicación de niveles de voltaje especiales (comúnmente en el orden de 10 a 25 voltios) a las entradas adecuadas en el circuito en una cantidad de tiempo especificada (por lo general 50 milisegundos por localidad de dirección). El proceso de programación generalmente es efectuado por un circuito especial de programación que está separado del circuito en el cual la EPROM eventualmente trabajará. El proceso de programación completo puede llevar varios minutos para un microcircuito EPROM.

En una EPROM las celdas de almacenamiento son transistores MOSFET que tienen una compuerta de silicio sin ninguna conexión eléctrica (es decir, una compuerta flotante). En su estado normal, cada transistor está apagado y cada celda guarda un 1 lógico. Un transistor puede encenderse mediante la aplicación de un pulso de programación de alto voltaje, el cual inyecta electrones de alta energía en la región formada por la compuerta flotante. Estos electrones permanecen en esta región una vez que ha finalizado el pulso ya que no existe ninguna trayectoria de descarga. Esto

mantiene el transistor encendido de manera permanente, aún cuando se retire la potencia de alimentación del dispositivo y la celda guarda ahora un 0 lógico. Durante el proceso de programación, se emplean las direcciones y terminales de la EPROM para seleccionar las celdas de memoria que serán programadas como ceros, así como las que se dejarán como unos.

Una vez que se ha programado una celda de la EPROM, se puede borrar su contenido exponiendo la EPROM a la luz ultravioleta (UV) , la cual se aplica a través de la ventana que se encuentra sobre el encapsulado del circuito. La luz UV produce una fotocorriente que va desde la compuerta flotante hacia el sustrato de silicio; con esto se apaga el transistor y se lleva de nuevo la celda hacia el estado uno lógico. Este proceso de borrado requiere entre 15 a 20 minutos de exposición a los rayos UV. Desafortunadamente, no existe ninguna forma de borrar solo algunas celdas; la luz UV borra todas las celdas al mismo tiempo por lo que una EPROM borrada almacena solamente unos lógicos. Una vez borrada la EPROM puede reprogramarse.

2.3 El convertidor digital a analógico

Existen varios métodos y circuitos para producir la operación D / A. En este proyecto hemos usado un convertidor D / A en circuito integrado, y

describiremos las características significativas de la forma en que realiza la transformación digital a analógica.

En la siguiente figura se muestra el circuito básico de un convertidor D / A de 4 bits. Las entradas A, B, C, D son entradas binarias que se supone

$$V_{sal} = -\left(V_D + \frac{V_C}{2} + \frac{V_B}{4} + \frac{V_A}{8}\right)$$

tiene valores de 0 o 5V. El amplificador operacional que sirve de amplificador sumador, el cual produce la suma de los factores de ponderación de estos voltajes de entrada. Desarrollando la fórmula para el voltaje de salida tenemos:

El signo negativo está presente debido a que el amplificador sumador es un amplificador inversor.

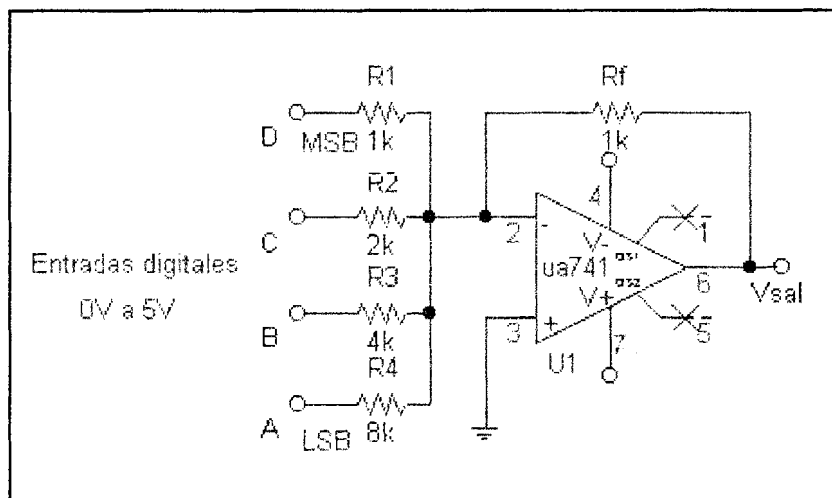


Fig.2.1 AMPLIFICADOR INVERSOR

La salida del amplificador sumador es un voltaje analógico que representa una suma de los valores de ponderación de las entradas digitales. Por ejemplo, si la entrada digital es 1010, entonces $V_D=V_B=5V$ y $V_C=V_A=0V$. Así, al utilizar la ecuación $V_{sal} = - (5V+0V+1/4 \times 5V+0V)$ por lo tanto $V_{sal} = -6.25V$. El voltaje de salida aumenta según se incremente el valor digital,

Las especificaciones del convertidor son: resolución precisión, offset.

2.3.1 Resolución

La resolución porcentual del convertidor depende únicamente del número de bits. Por esta razón los fabricantes por lo general especifican una resolución del convertidor por el número de bits. Un convertidor de 10 bits tiene una resolución más sensible (mayor exactitud) que uno de 6 bits. En nuestro proyecto utilizamos un convertidor de 8 bits.

2.3.2 Precisión

Los fabricantes tienen varias maneras de especificar la precisión. Las dos más comunes se llaman error de escala completa y error de linealidad, que normalmente se expresan como un porcentaje de la salida a escala completa del convertidor (%F.S.).

El error a escala completa es la máxima desviación de la salida del convertidor de su valor estimado (lineal); expresado a escala completa. Por ejemplo suponga que el convertidor tiene una exactitud de $\pm 0.01\%$ F.S. Como este convertidor tiene una salida a escala completa de 9.375 V, este porcentaje se convierte en $\pm 0.01\% \times 9.375$ V = ± 0.9375 mV.

Esto significa que la salida de este convertidor puede, en cualquier instante, variar 0.9375 mV de su valor esperado.

El error de linealidad es la desviación máxima en el tamaño de paso del tamaño de paso ideal. Por ejemplo, un convertidor que tenga un tamaño de paso estimado de 0.625 V. Si este convertidor tiene un error de linealidad de $\pm 0.01\%$ F.S., esto significaría que el tamaño de paso real podrían apartarse del estimado hasta 0.9375 mV.

Es importante entender que la precisión y resolución de un convertidor deben ser compatibles. Es lógico el tener una resolución de digamos 1% y una precisión de 0.1 por ciento y viceversa. Para ilustrar, un convertidor con una resolución de 1% y una salida a escala completa de 10 V. Puede producir un voltaje analógico de salida

dentro de 0.1 V de cualquier valor deseado asumiendo una exactitud perfecta. No tiene sentido tener una precisión costosa de 0.01 % de F.S. (o bien, 1 mV) si la resolución ya limita la proximidad al valor deseado a 0.12 V. Lo mismo puede decirse al tener una resolución muy pequeña (de muchos bits) en tanto que la exactitud es pobre, se trata de un desperdicio de bits de entrada.

2.3.2 Error de desplazamiento (offset)

En el caso ideal, la salida de un convertidor será de cero voltios cuando la entrada binaria es toda de ceros. Pero, en la práctica, habrá un voltaje de salida muy pequeño para esta situación y se llama error de desplazamiento. Este error de desplazamiento, si no se corrige, se sumará a la salida esperada de convertidor en todos los casos de entrada.

Por ejemplo digamos que un convertidor de cuatro bits tiene un error de desplazamiento de +2mV mayor que la esperada y esto se debe al error de desplazamiento; este puede ser tanto positivo como negativo.

Muchos convertidores tendrán un ajuste externo de desplazamiento que permitirá al usuario ponerlo en cero. Esto, por lo

general, se logra al aplicar todos los ceros a la entrada del convertidor y vigilar la salida mientras se gradúa un potenciómetro de ajuste de desplazamiento hasta que la salida esté lo mas cerca de 0 voltios como se requiere.

2.4 El Amplificador

Este último bloque es la amplificación de la señal, más concretamente es la conversión de corriente a voltaje, ya que el convertidor la señal de salida analógica es de corriente y mas no de voltaje, en nuestro trabajo colocamos el amplificador sobrecompensado denominado $\mu A741$ que es muy conocido.

La configuración de este amplificador operacional se denomina como amplificador inversor, ya que su salida es invertida con respecto a la entrada. No se va a profundizar en esto porque se explicará mas detalladamente en el capítulo III.

CAPITULO III
DESCRIPCION DEL FUNCIONAMIENTO DEL CAPTURADOR DE SEÑAL
CARDIACA (CSC)

3.1 Introducción.

Con la finalidad de obtener las formas de onda cardiaca lo más real posible, se implemento un dispositivo que sea capaz de receptor la información que se genera en el corazón, amplificarla, filtrarla y luego digitalizarla para poder almacenarla en la memoria de la computadora. Una vez guardado en la memoria de la computadora, con un software adecuado podemos observar cuantas veces que sea necesaria, la información obtenida, se puede tambien guardar un sinnúmero de muestras diferentes, gracias a una fuente de datos.

Esto hizo capaz que el simulador de onda cardiaca tenga en su memoria interna grabadas las ondas cardiacas con sus diferentes amplitudes de acuerdo a los puntos de muestreo, esto es LA, RA, C, LL, RL, en realidad la forma de onda es la misma en todos los casos lo unico que cambia es la amplitud.

Se puede decir que es una interface entre el paciente y la computadora, capaz de obtener la forma de onda idonea, y esta pueda ser observada en el monitor de su computadora, en donde se puede congelar la imagen para un mejor estudio de la misma, tal como se lo hace en el electrocardiografo.

3.2 Datos teoricos afines

3.2.1 El amplificador operacional

Los amplificadores operacionales (opam) son dispositivos muy usados actualmente en todos los circuitos electrónicos por su gran versatilidad, su fácil uso y su bajo costo, no es necesario conocer su estructura interna para poder usarlos, y si se comete algún error en el cableado esto no causa daño al dispositivo porque cuenta con diseño que lo protege de esta eventualidad.

3.2.2 El amplificador operacional de proposito general 741

3.2.2.1 Simbolo y terminales del circuito.

El simbolo del amplificador operacional que se muestra en la figura 3.1 es un triángulo que apunta en la dirección del flujo de la señal. Este componente tiene un número de identificación dentro del

simbolo del triángulo. El 741C que se muestra aqui es un amplificador operacional de proposito general que se utilizara en el diseño del captador de señal cardiaca.

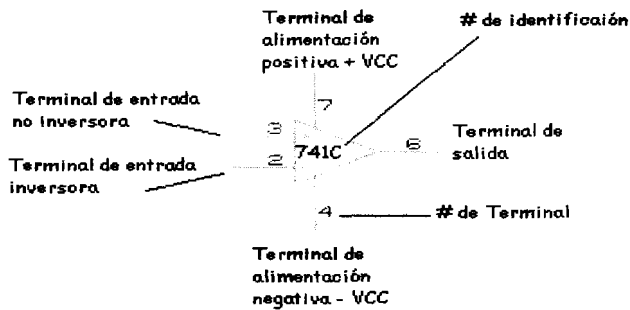


Figura 3.1 Símbolo y terminales de un opam

Todos los amplificadores tiene por lo menos 5 terminales: (1) la terminal de fuente de poder V_{cc} ó $+V$ en el pin 7, (2) la terminal de fuente negativa V_{EE} ó $-V$ en el pin 4, (3) la terminal de salida en el pin 6, (4) la terminal de entrada inversora en el pin 2, (5) la terminal de entrada no inversora en el pin 3, algunos amplificadores operacionales cuentan con más de 5 terminales.

3.2.3 Amplificador diferencial básico.

El amplificador diferencial básico es capaz de medir y amplificar pequeñas señales que pueden quedar ocultas en señales más intensas, como por

ejemplo el ruido, es por esto que este tipo de dispositivo es usado muy frecuentemente cuando se trata de adquirir datos.

El amplificador diferencial básico esta formado por 4 resistencias de precisión (1%) y un amplificador operacional (opam), como se muestra en la figura 3.2, hay dos terminales de entrada, denominadas entrada (-) y entrada (+), correspondientes a la terminal más cercana del opam. Si E1 es reemplazado por un cortocircuito, el resultado es un amplificador inversor con ganancia de $-m$. Por lo tanto, el voltaje de salida debido a E2 es igual a $-mE2$. Si se pone E2 en cortocircuito. E1 se divide entre R y mR para aplicar un voltaje de $mE1/(1+m)$. A la entrada (+) del opam. Esto da como resultado un opam noinversor con una ganancia igual a $(m+1)$. El voltaje de salida debido a E1 es el voltaje dividido, $mE1/(1+m)$, multiplicado por la ganancia del opam no inversor $(1+m)$, lo cual da $mE1$.

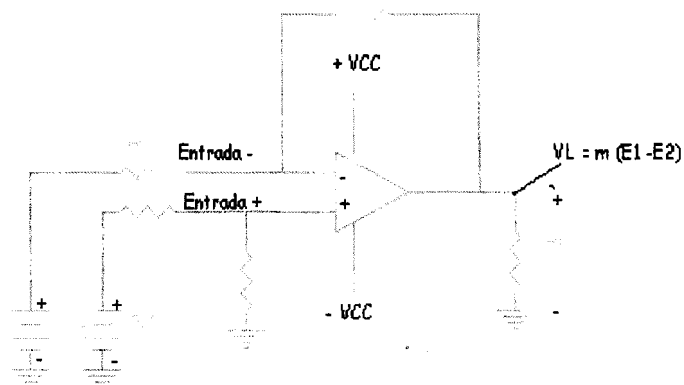


Figura 3.2 Amplificador diferencial básico

Por lo tanto, E_1 es amplificado a la salida por el multiplicador m a mE_1 . Cuando E_1 y E_2 están presentes en las entradas (+) y (-) respectivamente, V_o es $mE_1 - mE_2$, o

$$V_o = mE_1 - mE_2 = m(E_1 - E_2) \quad (3.1)$$

En esta ecuación se muestra que el voltaje en la salida del amplificador diferencial, V_o es proporcional a la diferencia de voltajes aplicados en las entradas (+) y (-). El factor multiplicador m se llama ganancia diferencial y se establece con la relación entre resistencias.

De la ecuación 3.1 nos damos cuenta que en el caso de que en ambas entrada se encuentren un mismo voltaje ($E_1 = E_2$) el voltaje de la salida será 0V.

3.2.3.1 Voltaje en modo común.

En la sección anterior vimos que para voltajes de entrada iguales es decir ($E_1 = E_2$) el voltaje de salida será 0V, estas entradas se las puede llamar voltaje común.

El modo más simple de aplicar voltajes iguales es conectar ambas entradas a una misma fuente de voltaje (figura 3.3). En

dicho caso el voltaje de entrada se llama voltaje de entrada en modo común ECM. Ahora V_o será $0V$, si las relaciones de las resistencias son iguales (mR a R para la ganancia del amplificador inversor es igual a mR a R del circuito divisor de voltaje). Prácticamente, las relaciones de resistencias se igualan mediante la instalación de un potenciómetro en serie con una resistencia, como se muestra en la figura 3.3. El potenciómetro se ajusta hasta que V_o se reduce a un valor despreciable. Esto provoca que la ganancia de voltaje en modo común V_o/ECM , se aproxime a $0V$. Esta es la característica de un amplificador diferencial que permite que una señal débil se capte extrayéndola de una señal de ruido más intensa. Es posible arreglar el circuito de modo que la señal más intensa no deseada, sea el voltaje de entrada común y la pequeña señal el voltaje de entrada diferencial. Entonces el voltaje de salida del amplificador amplificara solamente el voltaje diferencial de entrada

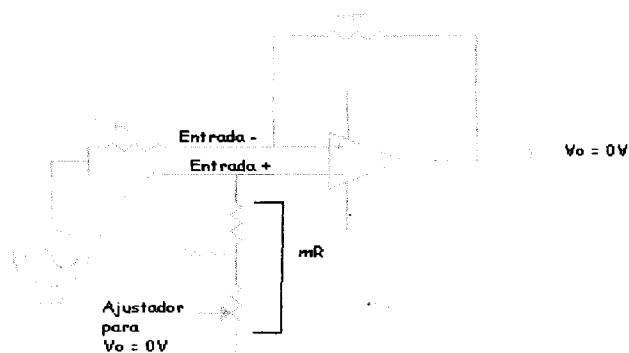


Figura 3.3 Ganancia de voltaje en modo común

3.2.4 Mejoras en el amplificador diferencial básico.

3.2.4.1 Incremento de la resistencia de entrada.

El amplificador diferencial tiene dos desventajas, tiene baja resistencia de entrada y el cambio de ganancia es difícil, porque las relaciones entre las resistencias deben igualarse estrechamente. La primera desventaja se elimina al aislar las entradas con seguidores de voltaje. Esto se realiza con dos opams conectados como seguidores como se muestra en la figura 3.4. La salida del opam A1 con respecto a tierra es E1 y la salida del opam A2 con respecto a tierra es E2. El voltaje diferencial de salida V_o se desarrolla a través de la resistencia R_L . V_o es igual a la diferencia entre E1 y E2 ($V_o = E1 + E2$). Obsérvese que la salida del amplificador diferencial básico de la figura 3.2 es una salida de extremo único, es decir el extremo de R_L está conectado a tierra y V_o se mide desde la terminal de salida del opam a tierra. El amplificador diferencial aislado de la figura 3.4 tiene salida diferencial; es decir, ningún extremo de R_L está conectado a tierra y V_o se mide sólo a través de R_L .

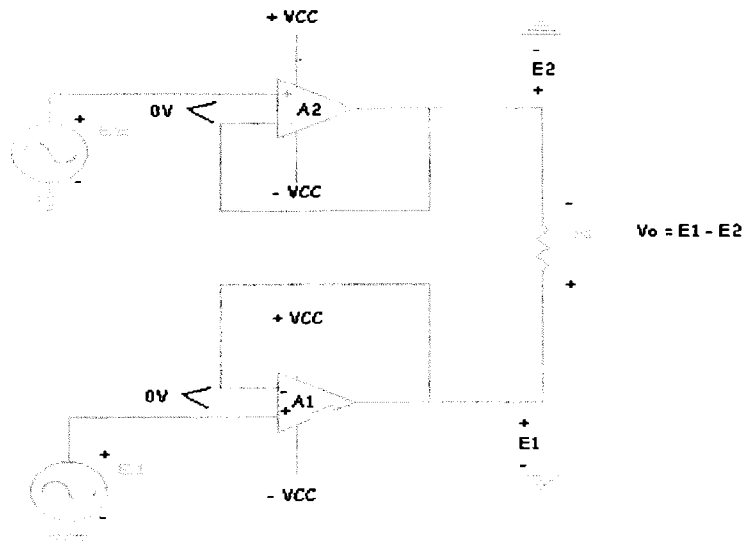


Figura 3.4 Entrada diferencial aislada al amplificador de salida diferencial

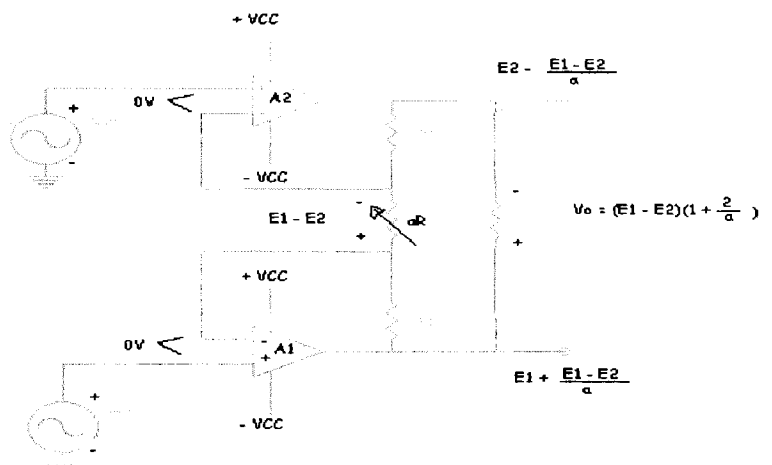


Figura 3.5 Entrada enicial al amplificador de salida diferencial con ganancia ajustable.

3.2.4.2 Ganancia ajustable.

La otra desventaja del amplificador diferencial básico es la falta de ganancia ajustable. Este problema se elimina al agregar tres

resistencias al amplificador aislado. El amplificador resultante, amplificador de entrada diferencial y salida diferencial, con ganancia ajustable se muestra en la figura 1.5 la alta resistencia de entrada se mantiene con los seguidores de voltaje.

Ya que el voltaje diferencial de entrada de cada opam es de 0V, los voltajes en los puntos 1 y 2 (con respecto a tierra) son iguales a E_1 y E_2 . Por lo tanto el voltaje a través de la resistencia aR es $E_1 - E_2$. La resistencia aR puede ser fija a un potenciómetro que se utiliza para ajustar la ganancia. La corriente a través de aR es:

$$I = \frac{E_1 - E_2}{aR} \quad (3.2)$$

Para cambiar la ganancia del amplificador, sólo tiene que ajustarse una resistencia única aR . Sin embargo este amplificador diferencial tiene una desventaja, solamente puede conectarse a cargas flotantes. Cargas flotantes son aquellas que no tiene terminal conectado a tierra. Para manejar cargas a tierra debe agregarse un circuito que convierta el voltaje diferencial de entrada en un voltaje de salida referido a tierra. Dicho circuito es el amplificador diferencial básico.

3.2.5 Amplificador de instrumentación

3.2.5.1 Operación del circuito.

El amplificador de instrumentación es de los más útiles, precisos y versátiles disponibles en la actualidad. Está conformado por tres opams y siete resistencias, como se muestra en la figura 3.6. Para simplificar el análisis del circuito, observe que este amplificador en realidad se hace conectando un amplificador aislado (figura 3.5) a un amplificador diferencial básico (figura 3.2).

El amplificador operacional A3 y sus cuatro resistencias iguales R forman un amplificador diferencial con una ganancia unitaria. Sólo las resistencias de A3, tienen que igualarse. La resistencia marcada como R', puede hacerse variable, para balancear eliminando cualquier voltaje en modo común, como se muestra en la figura 3.3. Sólo una resistencia aR, se usa para establecer la ganancia de acuerdo con la ecuación (3.3).

$$\frac{V_o}{E_1 - E_2} = 1 + \frac{2}{a} \quad (3.3)$$

Donde $a = aR/R$

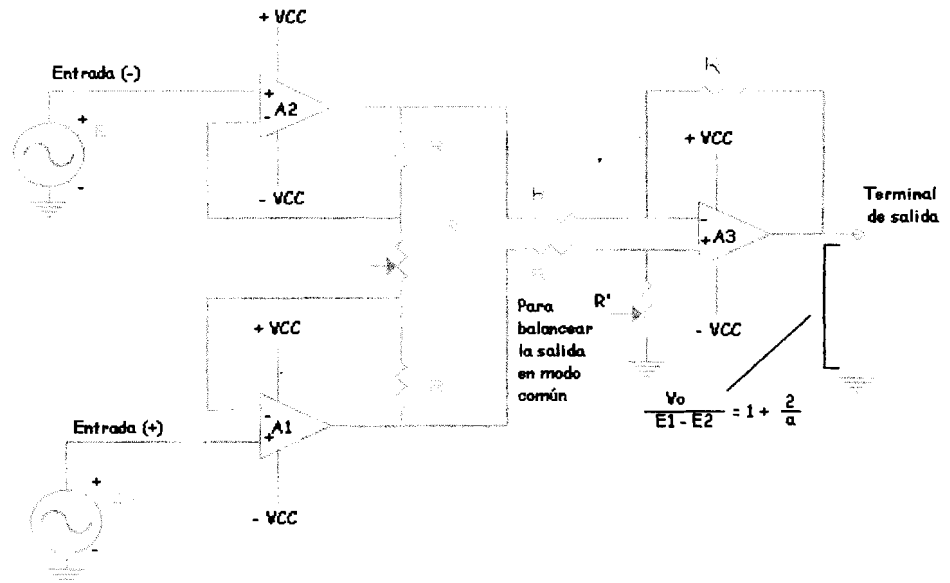


Figura 3.6 Amplificador de instrumentación

E_1 se aplica a la entrada (+) y E_2 a la entrada (-). V_o es proporcional a la diferencia entre los voltajes de entrada. Las características del amplificador de instrumentación se resumen así:

- 1 La ganancia de voltaje, desde la entrada diferencial ($E_1 - E_2$) a la salida del extremo único, se establece con una resistencia
- 2 La resistencia de entrada de ambas entradas es muy alta y no cambia al variar la ganancia.
- 3 V_o no depende del voltaje común a E_1 y E_2 (voltaje en modo común), sólo en su diferencia.

3.2.6 Reguladores de voltaje de tres terminales.

3.2.6.1 Introducción.

En la actualidad resulta muy común encontrar en las tarjetas electrónicas reguladores de voltaje, estos son muy populares por su fácil aplicación, estos dispositivos hacen posible contar con voltajes que deseamos para ciertos elementos en la placa, de esta manera en una misma placa pueden actuar elementos que trabajen a diferentes niveles de voltaje gracias a los reguladores, estos dispositivos estan diseñados normalmente para que trabajen a corrientes por debajo de 1.5 A. El simbolo esquemático del regulador de tres terminales se puede ver en la figura 3.7

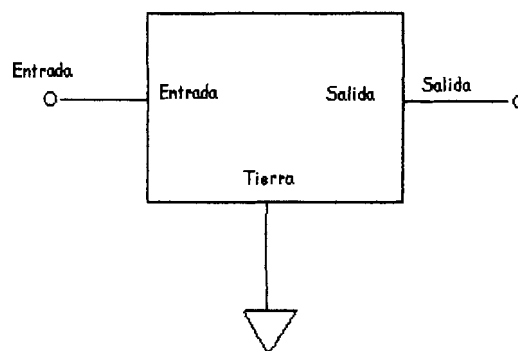


Figura 3.7 Diagrama esquemático de un regulador de tres terminales

3.2.6.2 Características.

Las características de los reguladores de voltaje de tres terminales se pueden resumir en tres puntos:

1. El voltaje de salida para muchos de los reguladores de voltaje de tres terminales es fijado en un valor específico por el fabricante, existen en el mercado reguladores de diferentes voltajes de salida, para satisfacer todas las necesidades.
2. El voltaje no regulado en la entrada es siempre mayor que el voltaje de salida regulado en valor absoluto. La cantidad en que excede el voltaje no regulado de la entrada se denomina voltaje de caída. Para muchos de estos reguladores el voltaje de caída está alrededor de 2 a 3 voltios, por ejemplo:

$$V_{sal} = 5 \text{ V} \quad E_{en \text{ min}} = 7\text{V}$$

$$V_{sal} = - 5 \quad E_{en \text{ min}} = - 7 \text{ V}$$

3. La corriente de salida o de la carga puede ser muy variada puede ir desde cero hasta un valor máximo de corriente de salida. Este valor máximo puede ser fijado por las necesidades del circuito, se supone que hablamos de pequeños circuitos, los valores de

corriente de salida estan limitados por varios factores entre los que se destaca la temperatura, pues a mayor corriente mayor temperatura. En el mercado podemos encontrar dispositivos con corrientes de salida máxima que van desde 0.1 A hasta 10 A.

4. Estos dispositivos cuentan con una protección contra altas temperaturas, para esto se encuentran provistos de sensores de calor. Cuando la temperatura se encuentra usualmente entre 125° C a 150° C el dispositivo se apaga y de esta manera se protege.

3.2.7 El convertidor analógico digital ADC0804.

3.2.7.1 Descripción general.

El ADC0804 es un dispositivo CMOS de 8bit que para realizar la conversión utiliza la técnica de aproximaciones sucesivas usando un arreglo resistivo similar al arreglo 256R. Este convertidor esta diseñado para que sus pines de entrada / salida pueden interactuar con cualquier microprocesador sin que sea necesario tener una interface lógica. Es un convertidor lo suficientemente rápido para este tipo de aplicación pues su tiempo de conversión es de 100us.

El voltaje de referencia de entrada puede ser ajustado, para aceptar cualquier señal pequeña de voltaje medida con una resolución de 8 bit.

3.2.7.2 Cualidades de ADC0804.

- Compatible con controladores 8080 y similares, no necesita interface lógica, tiempo de acceso de 135 ns.
- Fácil interface con cualquier microprocesador
- Voltaje analógico diferencial de entrada
- Entradas lógicas y salidas aptas en niveles de voltaje según las especificaciones entre MOS y TTL.
- Voltajes de trabajo de 2.5V (LM336) voltaje de referencia
- Posee un generador de pulsos.
- Voltaje de entrada en un rango de 0V a 5V, con 5V de alimentación.
- No requiere ajustar el cero

3.2.7.3 Especificaciones.

- 8 bit de resolución
- Error total $\pm 1/4$ LSB, $\pm 1/2$ LSB y ± 1 LSB.
- Tiempo de conversión 100 μ s.

Para más información técnica sobre el dispositivo ADC0804 podrá encontrar en el internet en la dirección www.national.com

3.2.8 Conversión analógica a digital.

3.2.8.1 Técnica de aproximaciones sucesivas.

En la figura 3.8 se muestra el diagrama de bloques de un convertidor analógico a digital usando (ADC). Consta de un convertidor analógico, un comparador y un registro de aproximaciones sucesivas (SAR). Se necesita una terminal para el voltaje de entrada analógica V_{in} . La salida digital esta disponible; en forma serie o paralela. Se requiere un mínimo de tres terminales de control: inicia conversión, da inicio la secuencia A/D, fin de conversión indica cuando se termina la conversión y una terminal externa de reloj establece el tiempo para completar cada conversión, en el caso del ADC0804 ya cuenta en su configuración interna con un generador de pulsos.

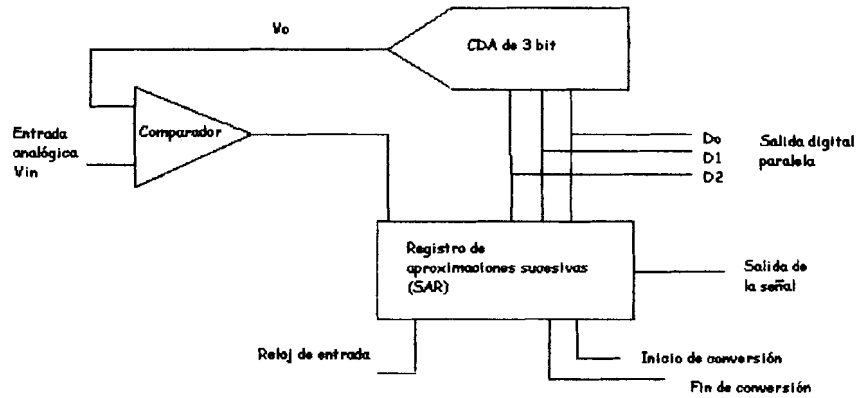


Figura 3.8 Diagrama de bloques de un convertidor analógico a digital por aproximaciones sucesivas

3.2.8.2 Operación del circuito

De la figura 3.8 la orden de inicio conversión, inicia el ciclo de conversión analógico a digital. El SAR conecta la secuencia de números digitales, un número por cada bit, a las entradas del convertidor digital a analógico.

El convertidor digital a analógico transforma cada número digital en una salida analógica V_o . El voltaje analógico de entrada se compara con V_o . El comparador le dice a SAR cuando V_{in} es mayor o menor que la salida del convertidor digital a analógico, V_o para cada bit de salida de 3 bits, deben efectuarse tres comparaciones.

Las comparaciones se hacen comenzando por el bit más significativo y terminan con el bit menos significativo. Al terminar la comparación el registro SAR envía la señal que finalizó la conversión. El equivalente digital de V_{en} está ahora presente en la salida digital del registro.

3.2.8.3 Analogía por aproximaciones sucesivas.

Suponga que se tienen los pesos de 1, 2 y 4 lbs. (SAR) y una balanza (comparador y convertidor digital a analógico). Considere el peso de 1lb como LSB y el peso más significativo de 4 lbs como MSB. Consulte las figuras 3.8 y 3.9. V_{en} corresponde a un peso desconocido.

Convertimos $V_{en} = 6.5 V$ en una salida digital (peso desconocido = 6.5 lb) se pondrá el peso desconocido en un platillo de la balanza y el de 4 lbs en el otro para comparar si el peso desconocido (V_{en}) excede a 4 lbs. El SAR utiliza un pulso de reloj para aplicar 100 bits (MSB) al convertidor digital a analógico en la figura 3.9 su salida, $V_o = 4V$, se compara con V_{en} , el bit más significativo se hace 1 si $V_{en} > V_o$. Esto es como dejar el peso de 4 lbs en la balanza.

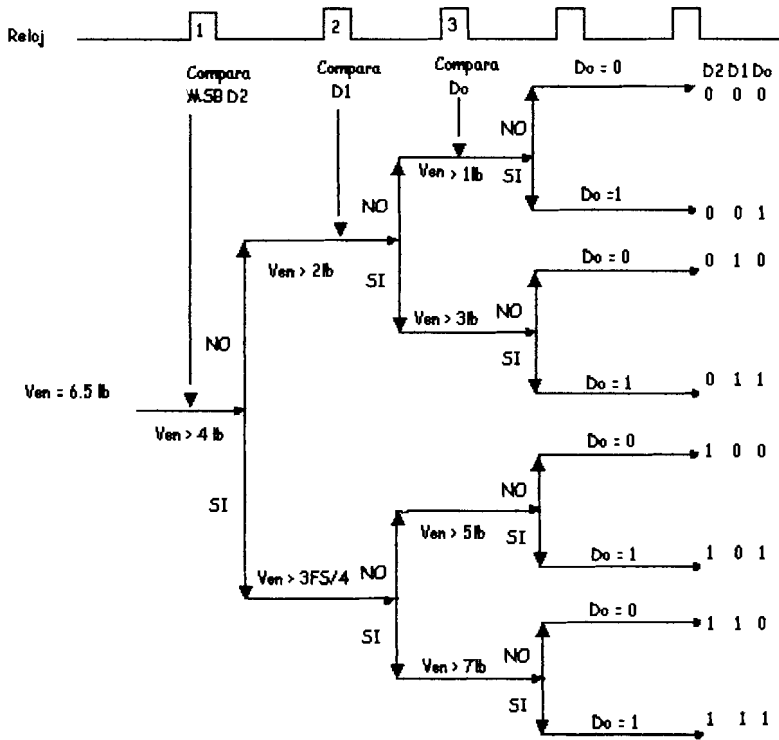


Figura 3.9 Diagrama explicativo de conversión analógica a digital

El SAR aplica después 110 (se agrega un peso de 2lbs.) al convertidor digital a analógico; D_1 se pone a 1 puesto que $V_{en} = 6.5 \text{ V}$ es mayor que 6 V . Por último, el registro aplica 111 al convertidor digital a analógico (se agrega 1 lb). Dado que $V_{en} = 6.5 \text{ V}$ es menor que 7 V , se pone D_0 a cero (se elimina el peso de 1 lb).

3.2.8.4 Tiempo de conversión.

La figura 3.9 muestra que se necesita un pulso de reloj para que el SAR compare cada bit. No obstante, casi siempre se requiere un pulso adicional para restablecer el registro antes al llevar a cabo la conversión. El tiempo de conversión que tarda una conversión analógica a digital dependerá tanto del periodo de reloj T como del número de bits n , la relación es :

$$T_c = T(n+1) \quad (3.4)$$

3.2.9 El buffer de acoplamiento 74 LS244

3.2.9.1 Descripción general.

Este dispositivo esta diseñado para optimizar el rendimiento entre las señales ingresantes y su PC, este buffer funciona como un controlador principal del administrador de direccionamiento de memoria del computador, del reloj y del bus de orientación transmisión / recepción. Con este dispositivo de acoplamiento nos aseguramos de no tener problemas con las PCs cuyos puertos paralelos no sean bidireccionables.

3.2.9.2 Características.

- Las entradas PNP reducen la carga DC en la línea de bus
- Reduce el margen de ruido en las entradas
- Retardo de tiempo típico enable / disable de 18 ns.

Más información técnica podrá encontrar en el internet en la dirección www.fairchildsemi.com

3.3 Descripción del funcionamiento del proyecto

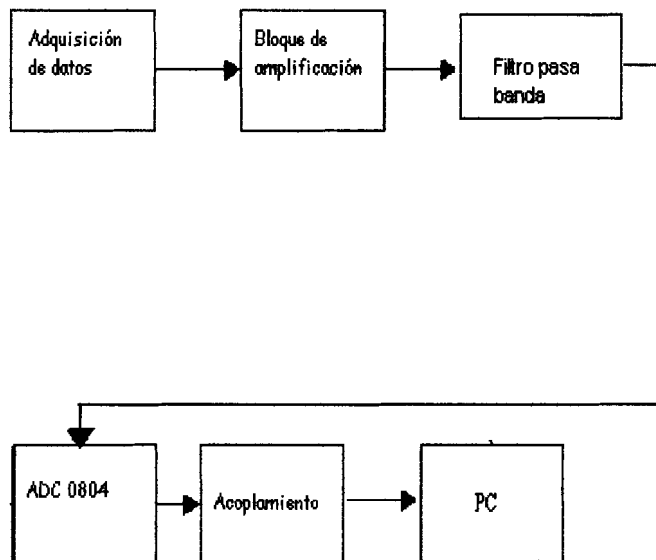


Figura 3.10 Diagrama de bloques del CSC

3.3.1 La adquisición de datos:

El bloque de la adquisición de datos esta compuesto por un amplificador diferencial de instrumentación, debido a que este arreglo es el más idoneo, por que discrimina el ruido de la señal de interes, aprovechando la cualidad de rechazo al voltaje de modo común.

3.3.2 La amplificación .

Es necesario que la señal a capturarse sea amplificada para que pueda ser manejada y apreciada por el convertidor analógico digital, puesto que la señal que se obtiene esta en en el oden de los milivoltios lo cual es muy pequeño para que pueda ser captada por el convertidor, en este caso debemos amplificarla al rango de los voltios maximo hasta 5V (el pico mas alto de la bonda). Para locual utilizamos dos opams, con el primer opam logramos amplificar la señal 100 veces y con el segundo opam 20 veces dando un factor de amplificación total de 2000 veces.

3.3.3 La conversión analógica / digital

Esto se realiza con la utilización de un convertidor analogico / digital como es el ADC 0804 de 8 bit , este convertidor tiene la ventaja de que en su configuración interna cuenta con su propio reloj, esto disminuye la

circuiteria en la tarjeta, dando mayor espacio físico. Este es un dispositivo rápido, su tiempo de conversión es de 100 us.

El ADC0804 consta de un arreglo resistivo, un reloj, un registro de aproximaciones sucesivas un convertidor digital analógica, un comparador. Tal como se mostro en la figura 3.8 (para mayor información consultar e la dirección www.national.com)

3.3.4 El acoplamiento

El acoplamiento se lo hace posible gracias al buffer de acoplamiento de 10 bits el registro de acoplamiento 74LS244, permite que podamos usar un computador, cuyo puerto paralelo sea unidireccional, esto es muy útil cuando tenemos un PC no actualizado, cuando este no sea el caso el buffer de acoplamiento puede ser despreciado.

El buffer de acoplamiento 74LS244 tiene 10 bits, de los cuales 8bits corresponden a los datos de la señal que se quiere capturar y los 2 bits restantes corresponden a las señales de control, esta señales de control son fundamentales cuando nuestra computadora no cuenta con un puerto paralelo bidireccional.

Luego del buffer de acoplamiento le sigue el conector DB25, aqui la señal ya digitalizada ingresa por los pines 7,8,9,10,12,13,15,y18 del

conector DB25, para cada punto que compone la señal se puede establecer un valor de 0 a 255 para la amplitud. Ese valor es colocado en el puerto paralelo bajo la forma de un valor binario de 8bit.

3.3.5 El filtrado de la señal.

El filtrado de la señal se lo realiza por medio de software, este simula la labor de un filtro pasabajas. Lo que el programa hace es muestrear la señal y el ruido que esta pueda tener es eliminado al tomar en cuenta los valores promedios de la misma, practicamente lo mismo que haria un arreglo filtro pasa baja en el hardware.

3.3.6 La fuente de alimentación

El sistema de alimentación que se aplica para el CSC, consiste en un transformador de 110V a 12V conectado a un puente rectificador de onda completa, una vez rectificada la onda, el votaje de las salidas tanto positiva, como negativa son reguladas por los reguladores 7812 y el 1912 positivo y negativo respectivamente, como se puede ver en la figura 3.12 con esto logramos tener dos voltajes de + 12V y de -12V que son necesarios para la polarización de los cuatro opam que se utilizan en el CSC. Se puede ver que existen capacitores conectados en

las salidas de los voltajes tanto positivo y negativo, esto se hace para obtener una señal DC pura.

En la parte positiva conectamos un regulador de + 5V, (7805) pero antes se coloca cuatro resistencias de 120Ω con lo que reducimos el voltaje de entrada al regulador, de esta manera evitamos que el 7805 se caliente demasiado, el voltaje de 5V resultante es necesario para polarizar el ADC0804 y el 74LS244.

3.4 Diagrama esquemático del CSC

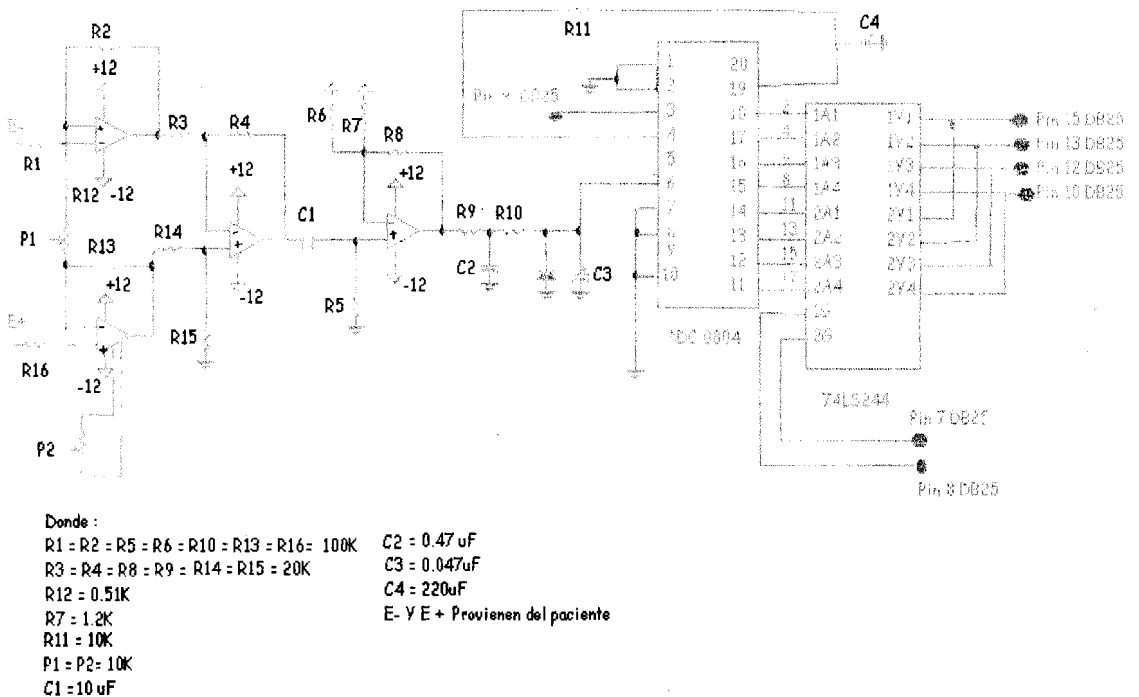
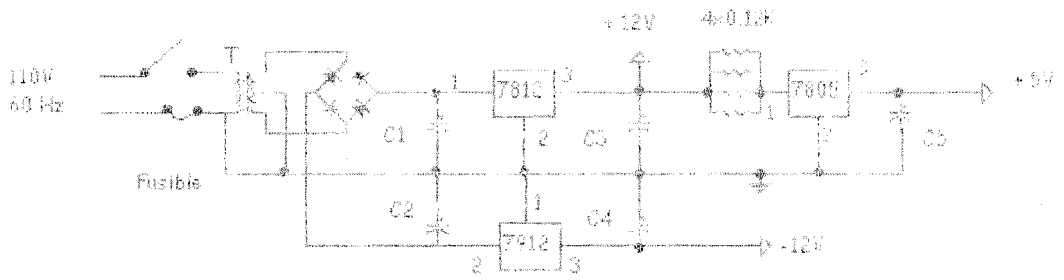


figura 3.11 Diagrama esquemático del CSC



Donde:
 $C1 = C2 = 470\mu\text{F}$
 $C3 = C5 = 4.7\mu\text{F}$
 $C4 = 1\mu\text{F}$
 $T = 110 / 12 \text{ V}$

Figura 3.12 Diagrama esquemático de alimentación del CSC

CAPITULO IV

ANALISIS DEL CIRCUITO PARA EL SIMULADOR CARDIACO

En nuestro proyecto nos referiremos a uno de los miembros más populares de los microcontroladores de la familia 51: los 8XX1, y más concretamente, los 8X51. Las diferencias entre estos miembros se muestran en la tabla 4.1.

Como se puede observar existen tres versiones para cada uno: Una versión sin ROM, cuyo prefijo es 80 y sufijo 31, y dos versiones con ROM, con sufijo 51, la versión EPROM tiene el prefijo 87. Existe un miembro un poco más completo de esta familia, el cual cambia el número 1 al 2, y posee mayor capacidad de ROM, de RAM y de temporizadores / contadores.

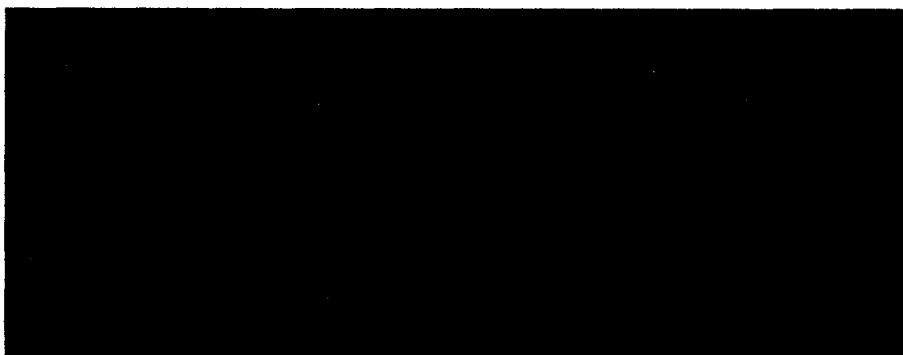


TABLA 4.1 MIEMBROS DE LA FAMILIA MCS-51

4.1 Características de los Microcontroladores 8X51

La arquitectura básica de esta familia de microcontroladores se muestra en la figura y sus características más importantes son:

- CPU de 8 bits, optimizada para aplicaciones de control
- Procesador Booleano (operaciones sobre bits)
- Espacio de memoria de programa de 64 Kbytes
- Espacio de memoria de datos de 64 Kbytes
- Cuatro Kbytes de memoria interna de programa
- 128 bytes de memoria RAM interna.

- 32 Líneas de entrada / salida direccionables bit a bit.
- 2 temporizadores / contadores de 16 bits
- Comunicación asincrónica. Full Duplex.
- 5 Fuentes de interrupción.
- Oscilador interno

4.1.1 Organización de la memoria

Todos los dispositivos de la familia MCS-51 tienen separados los espacios de direcciones de datos y de programas, como se muestra en la figura 4.1.

La separación lógica de ambas memorias permite que a través de la memoria de datos se acceda a direcciones de 8 bits, lo cual permite que los datos puedan ser manipulados y almacenados más rápidamente por el CPU de 8 bits.

La memoria de programa solamente puede ser leída, no escrita, y se puede acceder hasta 64 Kbytes. Las versiones ROM y EPROM los más bajos de 4 K a 8 K están dentro del chip dependiendo del tipo de dispositivo.

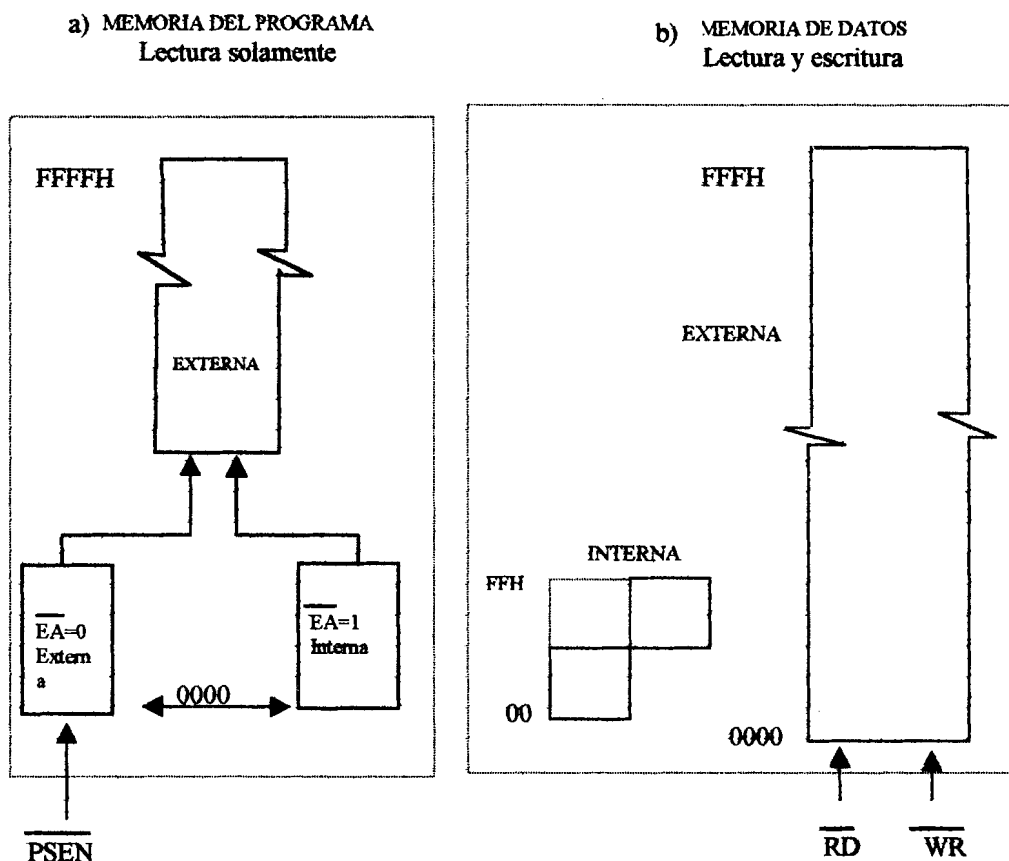


Fig. 4.1 ORGANIZACIÓN DE LAS MEMORIAS

La memoria externa de datos puede ser direccionada hasta 64 Kbytes y ella puede ser escrita o leída para lo cual la CPU genera las señales WR y RD respectivamente.

Las memorias externas de datos y externas de programa pueden coexistir, si se desea aplicando las señales RD y PSEN a las entradas de una compuerta AND y utilizando la salida de la compuerta como señal de sincronización a la memoria externa de programa datos.

4.1.1.1 Memoria de Programa

La memoria de programa puede ser una combinación externa e interna o totalmente externa. En cualquiera de los dos casos anteriores en la parte baja de memoria de programas se encuentran una serie de posiciones de memoria, especiales para el tratamiento de las interrupciones.

Una interrupción se describe como una señal que genera un dispositivo para indicarle a la CPU que se requiere su atención. Las interrupciones nacen de la necesidad de ejecutar un proceso en un instante preciso y ante la existencia de varias pueden crearse prioridades entre ellas de tal forma que una interrupción de baja prioridad puede ser interrumpida por una de más alta pero no se presenta el proceso inverso.

El CPU atiende la interrupción mediante una rutina especial realizada en el programa, la cual es llamada Rutina de Servicio una vez la interrupción es atendida, el CPU regresa al punto en el cual había suspendido sus labores, como se muestra en la siguiente figura 4.2.

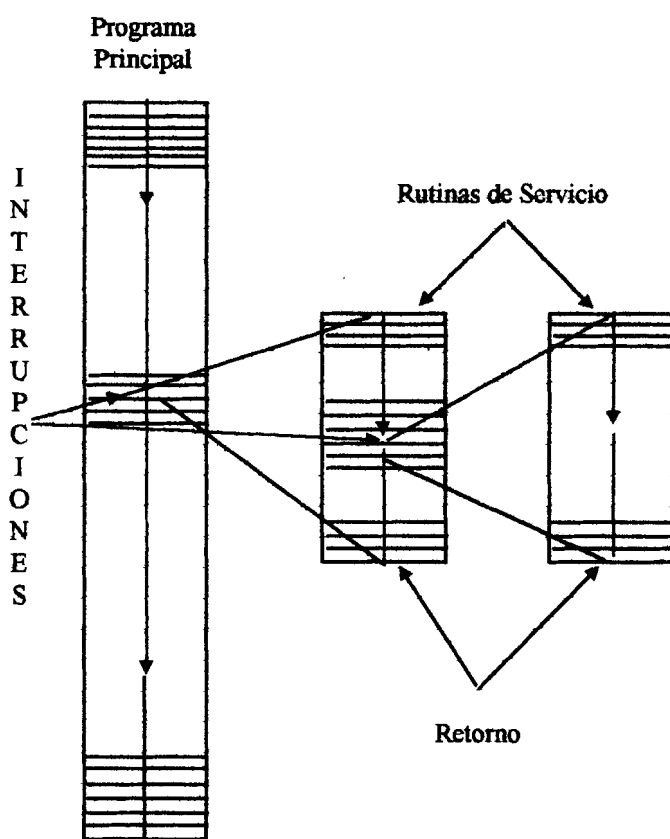


FIG. 4.2 ESQUEMA DE UNA INTERRUPCIÓN

Este proceso se lleva a cabo con la ayuda de la memoria RAM en la cual se almacena provisionalmente la posición de la instrucción que estaba por ejecutarse antes del llamado.

Como puede existir un buen número de interrupciones, generadas por diferentes condiciones de los dispositivos cada una de ellas debe tener su propia rutina de servicio, localizada en una posición específica de la memoria de programa. Las interrupciones pueden ser internas y externas dependiendo que si la fuente que la origina está dentro del microcontrolador o por fuera de este.

Como se muestra en la figura 4.3, a cada interrupción se le asigna una dirección fija de la memoria de programa la interrupción hace que el CPU salte a esta localidad donde comienza la ejecución de una rutina de servicio. La posición 0003H por ejemplo está asociada a la interrupción externa 0. Si la interrupción externa va a ser utilizada, la rutina de servicio debe empezar en esta dirección. Si la rutina no se utiliza, la posición de memoria puede ser empleada como memoria de programa.

Las rutinas de servicio de la interrupción están separadas por intervalos de 8 bytes, si estas no son, los suficientes cortas las rutinas pueden ser utilizadas introducciones de salto a una zona

más amplia de la memoria de programas con capacidad para contener el tratamiento completo a la interrupción.

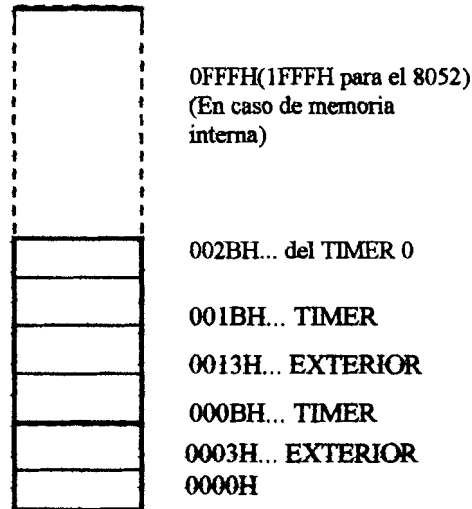


Fig. 4.3 MEMORIA DE PROGRAMA

Como se ha dicho, la parte baja de la memoria de programa puede ser de 4 K a 8 K, la cual se puede seleccionar dentro de la memoria interna o la externa. La selección se puede realizar mediante la comutación del pin EA (Externa Access) a Vee o Vss; si este pin se conecta a Vcc, la parte baja se buscará en la memoria interna del microcontrolador, si se conecta lo hará en la memoria externa, en ambos casos la parte superior de la memoria de programa (60 K o 56 K). Se buscará en la memoria externa en los dispositivos sin ROM, este pin debe conectarse a Vss para que el programa se

ejecute apropiadamente. Una señal de lectura de la ROM externa PSEN se utiliza para todas las búsquedas de memoria externa y no es activada en ningún caso, para búsqueda en la memoria interna.

En la figura 4.4 se muestra la configuración a utilizar para la ejecución del programa de memoria externa. Se puede observar que 16 líneas de I/O (puerto 0 y 2) son dedicadas a las funciones de bus durante la ejecución de la memoria externa. El puerto P0 es multiplexado en el tiempo como bus de direcciones y de datos. Este emite la parte baja del contador de programas PCL como una dirección y luego pasa a un estado flotante, esperando la llegada del código proveniente de la memoria de programa.

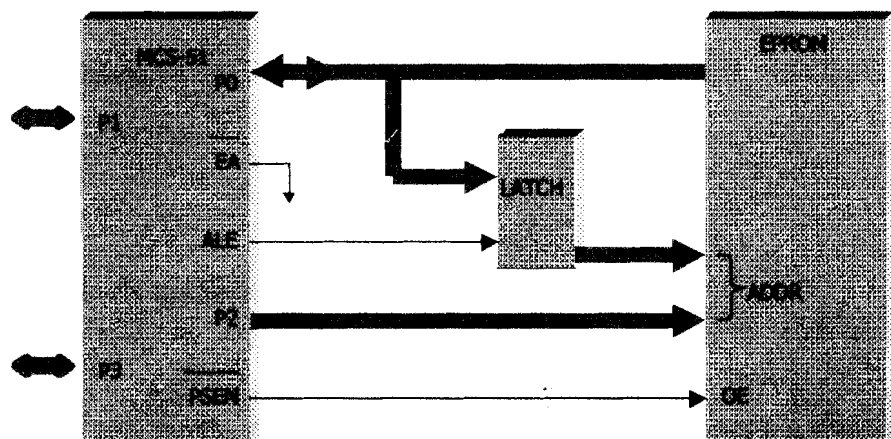


FIG. 4.4 OPERACION CON PROGRAMA EXTERNO

Durante el tiempo en el cual el byte bajo del contador de programas es estable en P1, la señal ALE fija este bite en el cerrojo de direcciones mientras tanto el puerto P2 emite la parte alta del contador de programas PCH. Finalmente se emite el pulso de sincronización PSEN y el código es leído por el microcontrolador el direccionamiento de programas es siempre de 16 bits de ancho, aunque el total disponible de memoria de programas sea menor a esta cantidad. La ejecución de programas externos sacrifica dos de los cuatro puertos de ocho bits P1 y P2, para direccionar la memoria de programas.

Las secuencias de búsqueda y ejecución de las instrucciones no dependen de la utilización de la memoria interna o externa lo que quiere decir, en ambos casos, los tiempos de ejecución serán los mismos.

4.1.1.2 Ciclo de máquina

En esta familia de microcontroladores un ciclo de máquina consiste en una secuencia de seis estados, numerados desde S1 hasta S6. Cada estado está formado por dos periodos de la señal de reloj y se denominan fases P1 y P2 por lo tanto un ciclo de máquina toma 12

períodos de reloj. Si la frecuencia del reloj es de 12 MHz, el ciclo de máquina tendrá una duración de un microsegundo como se muestra en la figura.

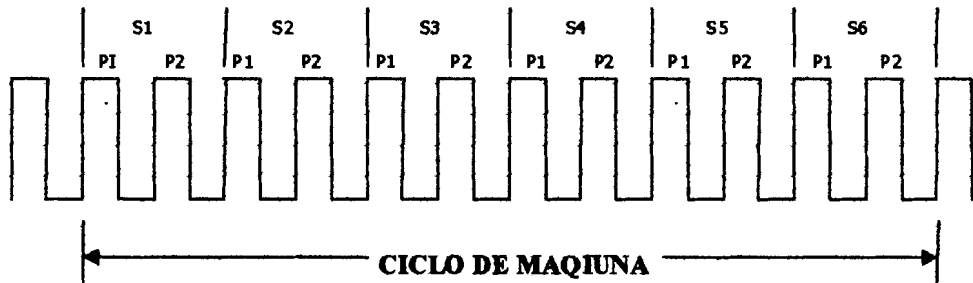


FIG. 4. 5 CICLO DE MAQUINA

4.1.1.3 Memoria de datos

La familia MCS-51 puede acceder hasta 64 Kbytes de memoria externa de datos. La configuración de la figura 4.6 podrá utilizar hasta 2 Kbytes de memoria externa.

En este caso la ejecución se está realizando desde la ROM interna. De nuevo el puerto P1 esta siendo multiplexado como bus de direcciones y datos de la RAM y tres líneas del puerto P2 se utiliza para paginarla. El CPU genera las señales RD y WR necesarias durante el acceso de la RAM externa. El direccionamiento de la

memoria externa puede ser de 1 o 2 bytes de ancho. El direccionamiento de un byte se logra por el puerto P0 en conjunto con una o más líneas I/O para paginar la RAM como se muestra en la figura 4.6. También se puede utilizar el direccionamiento de 2 bytes en cuyo caso el byte alto de dirección es emitido por el puerto P2.

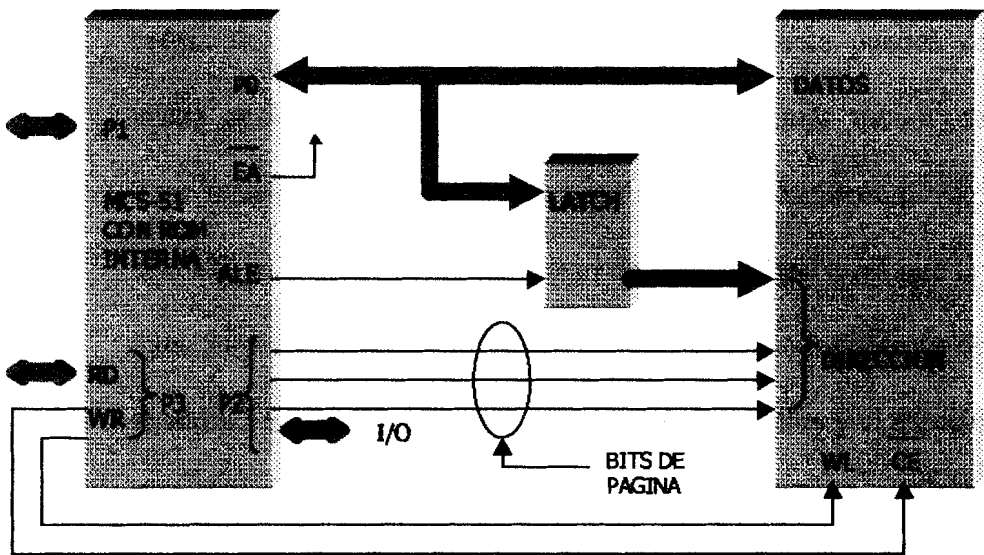


FIG. 4.6 MICROCONTROLADOR CON MEMORIA EXTERNA DE DATOS

El mapa de memoria interna se puede observar en la figura 4.7.

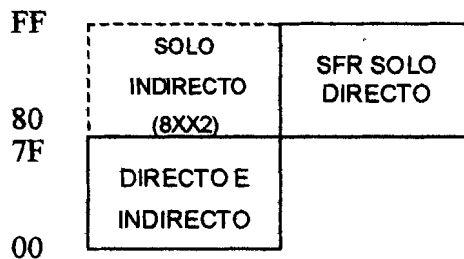


FIG. 4.7 ESPACIO DE MEMORIA RAM INTERNA Y FORMA DE DIRECCIONAMIENTO

El espacio de memoria se muestra dividido en tres bloques, los cuales son llamados por lo general como los 128 bajos, los 128 altos y el espacio de los SFR o registros de propósito especial.

El direccionamiento de la memoria es siempre de un byte de ancho, lo que implica que el espacio de direcciones es de solamente de 256 bytes. Sin embargo, los modos de direccionamiento pueden de hecho acomodar 384 bytes usando dos diferentes modos de direccionamiento de las instrucciones. El direccionamiento directo más alto que 7FH utiliza un espacio de memoria, mientras que un direccionamiento indirecto 7FH accede a un espacio diferente de memoria. Por esto la figura 3.7 muestra los 128 altos y el espacio de las SFR compartiendo el mismo bloque de direcciones, 80H hasta FFH aunque ellos son físicamente entidades separadas. Vale la pena aclarar que el área de direccionamiento solo indirecto (posiciones de RAM 80H a FFH) únicamente está presente en los 8XX2.

Los 128 bajos están presentes en todos los dispositivos MCS-51 su mapa se muestra en la figura 4.8. Como se puede observar se ha dividido en tres segmentos principales:

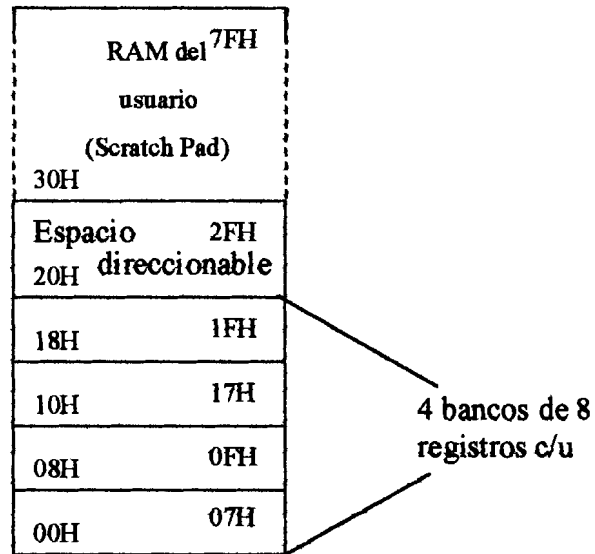


FIG. 4.8 LOS 128 BAJOS

4.1.1.3.1 Bancos de registros

Los 32 bits más bajos (00 a 1FH) se agrupan en 4 bancos de 8 registros las instrucciones de programas llaman a estos registros R0 hasta R7. Dos bits en la palabras de estados (PSW) seleccionan cual banco de registros están en uso.

Esto permite un uso más eficiente del espacio de código así como instrucciones que involucran registros más cortos que aquellos que utilizan direccionamiento

indirecto. Por defecto cuando se aplica un RESET al uC el banco en uso es el 0.

4.1.1.3.2 Area direccionable de bit a bit

Los 16 bites siguientes (20H a 2FH), conformar un bloque en el que se puede direccionar directamente bit por bit. El conjunto de instrucciones de los MCS 51 incluyen una amplia variedad de instrucciones orientadas hacia bits, y los 128 bits de esta área se pueden manejar directamente por éstas. Las direcciones de los bits de esta área van desde 00H hasta 7FH.

4.1.1.3.3 Area de RAM del usuario

Las posiciones restantes de la memoria de RAM interna (30H a 7FH) conforman la RAM de trabajo del usuario o Scratch Pad .

En la tabla 4.2 muestra la disposición de los registros de propósito especial o SFR los cuales se pueden acceder solamente por direccionamiento directo. Aquellos registros que se encuentran en la columna izquierda de la tabla también son direccionables bit a bit. Los registros encerrados existen en lo 8XX2 no en los 8XX1, notará

que no todas las direcciones están ocupadas, las direcciones desocupadas no están implementadas en el integrado. Al realizar lecturas en estas direcciones se obtendrán datos aleatorios, mientras que escribir en ellas no tendrán efecto alguno estas posiciones se reservan para futuros productos MCS-51 que poseerán nuevas características.

F8								FF
F0	B							F7
E8								EF
E0	ACC							E7
D8								DF
D0	PSW							D7
C8	(T2CON)		(RCAP2L)	(RCAP2H)	(TL2)	(TH2)		CF
C0								C7
B8	IP							BF
B0	P3							B7
A8	IE							AF
A0	P2							A7
98	SCON	SBUF						9F
90	P1							97
88	TCON	TMOD	TL0	TL1	TH0	TH1		8F
80	P0	SP	DPL	DPH			PCON	87

TABLA 4.2 AREA DE REGISTRO DE PROPÓSITO ESPECIAL SFR

La función de cada uno de los registros de propósito especial se describe a continuación empezando por los direccionables bit a bit.

ACC es el registro acumulador, sobre el cual se realizan las operaciones lógicas, aritméticas de transferencia.

B Se usa durante operaciones de multiplicación y división. Para otras instrucciones este puede ser tratado como cualquier otro registro del Scratch Pad.

PSW Program Status Word o Palabras de Estados del Programa. Contiene información del Estado del Programa y las banderas que son afectadas como resultados de operaciones.

P0, P1, P2 y P3. Puertos 0 a 3 son los cerrojos de los puertos 0,1,2 y 3 respectivamente.

IE. Habilitador de interrupciones. Permite que se atiendan las interrupciones que el usuario considere convenientes.

IP- Prioridad de interrupciones. Cada interrupción puede programarse con un grado de prioridad, de tal manera que una interrupción con alta prioridad no puede ser interrumpida por otra de prioridad más baja, pero si se presenta el efecto inverso.

SCON. Control del puerto serial. Establece los parámetros para la transmisión o recepción de datos a través del puerto serie (ancho del dato, bit de arranque y parada paridad, velocidad).

TCON. Control de temporizador / contador. Controla los modos de operación de los temporizadores y maneja las interrupciones asociadas, así como los flancos de activación.

4.1.1.4 Registros no Direccionables bit a bit

SBUF. Buffer serial de datos son dos registros buffer que se utilizan para la transmisión y recepción de datos a través del puerto serial

TMOD. Control de Modo Temporizador / Contador. Selecciona los modos de operación de los timers, si actúan como temporizadores o como contadores, etc.

SP. Puntero de Pila. Este se incrementa antes que un dato sea almacenado a causa de una instrucción **PUSH** o **CALL**, la pila es un área de la memoria RAM que se utiliza para almacenar temporalmente información básica del programa, antes de atender una interrupción, la pila puede residir en cualquier sitio de la memoria RAM pero el puntero se inicializa después del Reset con 07. Esto ocasiona que la pila se inicie en la dirección 08.

DPTR. Puntero de Datos. Está constituido por una parte alta DPH y una parte baja DPL y sostiene una dirección de 16 bits. Se puede manejar como un registro de 16 bits o como 2 registros independientes de 8.

T0 y T1. Temporizadores. Son registros de 16 bits que pueden actuar como temporizadores o contadores incluyen los registros de 8 bits TH0, TLO, TH1 y TL1.

PCON. Control de Poder. Posee un bit que cambia la velocidad de comunicación serial y en versiones CHMOS de esta familia, puede modificar el régimen de trabajo del microcontrolador, reduciendo el consumo.

4.1.1.5 Registros exclusivos de los 8XX2

T2CON. Lo mismo que TMOD y TCON, pero aplicado al timer adicional que posee esta versión uC.

RCAP2. Registro de captura incluye dos registros de 8 bits (RCAP2H y RCAP2L) que trabajan en conjunto con el TIMER 2.

TH2. La misma función de T0 y T1 descritos anteriormente

En los anexos se tiene una estructura más detallada de la familia de MCS 51. Se pueden observar además de los bloques y los buses,

los pines de conexión. Estos pines se muestran en los anexos, específicamente en la hoja de datos del microcontrolador, correspondientes a la descripción de los pines del microcontrolador

Vcc es la alimentación del integrado y se debe conectar por tanto a +5 voltios.

Vss tierra o referencia del circuito

ALE/PROG. Habilitador del cerrojo de direcciones. Por este pin se emite la señal para enclavar el byte bajo de las direcciones, cuando se accede a la memoria externa. También es la entrada de los pulsos de programación de la memoria EPROM.

PSEN. Habilitador del programa almacenado. Es la señal de sincronización para leer la memoria externa de programa PSEN solo se origina cuando se lee la memoria externa.

EA/VPP. Dirección externa. Cuando se mantiene en un nivel lógico alto, solo se ejecuta el programa de la memoria interna, a menos que el contador de programas exceda de 0FFFH para el 8X51 o 1FFFH para los 8X52. Si este pin se mantiene a un nivel bajo, el programa que se ejecuta es el de la memoria externa independientemente de la dirección del programa. Durante la

programación de la memoria PROM este pin recibe un voltaje que oscila entre 5 voltios y la tensión de programación

XTAL1 y XTAL2. Entrada y salida del oscilador, respectivamente. Se puede utilizar un cristal de cuarzo o un resonador cerámico.

RESET. Entrada para reinicialización del sistema. EL RESET se produce cuando se coloca este pin a un nivel alto, o por lo menos durante 2 ciclos de máquina.

Puertos. Todos son bidireccionales de 8 bits cuando se describen 1's a los pines. Estos se pueden usar como entrada de alta impedancia. Como salida cada pin del puerto P0 puede manejar 8 cargas TTL-LS el resto de los puertos puede manejar solo 4 cargas TTL-LS estos puertos también pueden actuar de la siguiente manera.

Puerto 0 (P0) es multiplexado como la parte baja de la dirección y del bus de datos durante el acceso de la memoria externa. Además emite los códigos durante la programación de la EPROM y los recibe en la verificación del programa

Puerto 1 (P1). Durante la programación y verificación de la EPROM recibe la parte baja de las direcciones. Sus 2 bits más bajos tienen una función adicional en los 8XX2

P1.0 :T2. Temporizador 2. Entrada externa para este temporizador contador solamente en los 8XX2

P1. 1: T2EX. Captura y recarga de disparo para el TIMER 2.

Puerto 2 P2. Emite la parte alta de las direcciones durante el acceso externo de memoria y cuando el acceso de datos tiene 16 bits asimismo durante la programación y verificación de la EPROM .

Puerto 3 (P3). Este tiene funciones alternas especiales bien determinadas tales como

P3.0: RXD. Entrada del puerto serie.

P3.1: TXD. Salida del puerto serie

P3.2: INT0. Entrada de la Interrupción Externa 0

P3.3 : INT1. Entrada de la interrupción Externa 1

P3.4 : T0. Entrada Externa del Temporizador / Contador 0

P3.5 : T1. Entrada Externa del temporizador / Contador 1

P3.6: WR: Señal de escritura para elementos externos.

P3.7 : RD: Señal de lectura para elementos externos

4.1.2 Temporizadores / Contadores

Un **TIMER** trabaja como un temporizador cuando incrementa o decrementa su cuenta con base en los impulsos recibidos por el reloj de instrucción del equipo. Lo hará como un contador, cuando su cuenta cambia ante la variación de una condición externa al dispositivo. Los dos estados son excluyentes y pueden ser programados por el usuario, con varias opciones de operación y activación.

4.1.3 Comunicaciones

El puerto serie trabaja en modo Full Duplex, lo que significa que puede transmitir datos simultáneamente. La familia MCS-51 posee cuatro modos de comunicación, dentro de los cuales se encuentra sincrónicos y asincrónicos, con velocidades variables por el usuario.

Estos modos de comunicación no solamente le permiten dialogar con otros microcontroladores de la misma familia, sino con la de otros fabricantes, con memorias seriales e incluso con computadoras personales de tipo PC.

Con la anterior estructura, es posible llegar a implementar sistemas de adquisición de datos como el que se muestra en la figura 4.9. Este sistema puede funcionar autónomamente proveniente de ocho entradas diferentes para posteriormente vía RS-232, ser transmitidas a un computador tipo PC, con el cual se pueden realizar los análisis estadísticos y gráficos más adecuados.

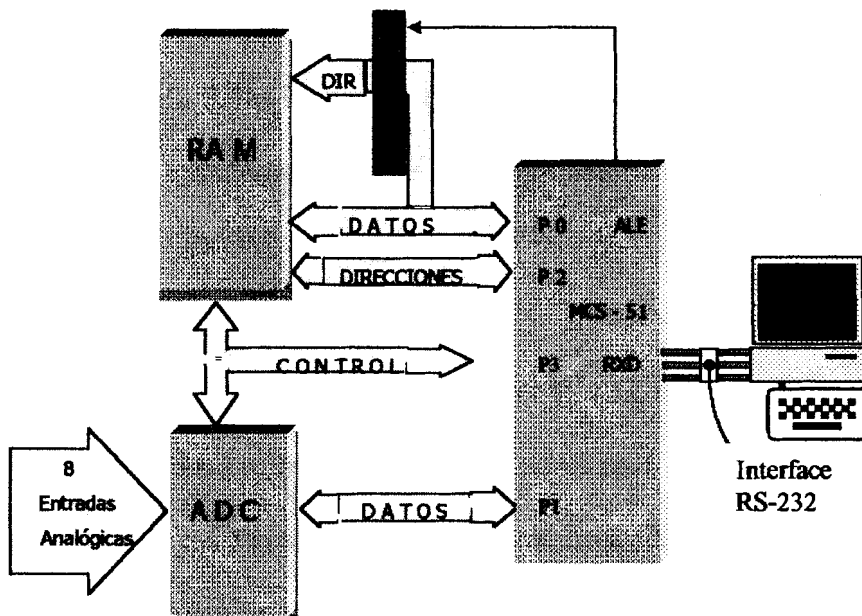


Fig. 4.9 SISTEMA DE ADQUISICION DE DATOS

4.2 Conexiones.

4.2.1 Conexión del microcontrolador

Después de revisar todas las características del microcontrolador 8751, ahora anotaremos, como se encuentran conectado el microcontrolador,.

Primero fue necesario aprender a programar el microcontrolador, mediante el software llamado RCHPSIM, ya este microcontrolador tiene sus propias interrupciones. Este software es el ensamblador, es decir , que se edita un programa con extensión ASM, luego se lo ensambla y se crea un programa HEX (hexadecimal). Este archivo hexadecimal es el que se copia en la EPROM del microcontrolador. La forma en que se graba el archivo hexadecimal en el microcontrolador es mediante un grabador universal de EPROM.

Hemos utilizado los puertos 0, 1, 2 como ya sabemos estos puertos son bidireccionales. El puerto 1 sirve para direccionar los datos de la EPROM. La EPROM 27C256 tiene una capacidad de 32 KB, por lo tanto posee una barra de 15 líneas de dirección, y una barra de 8 líneas de datos.

El puerto 2 sirve para adquirir los datos del ADC que se lee del potenciómetro y que sirva para variar la frecuencia cardíaca. De este puerto se utilizó los pines P2.0 para activar la señal START del convertidor, en general este pin es la señal SOC. Los pines P2.1 hasta P2.7 se utilizan para adquirir datos ya digitales que aumentan o disminuyen según se varíe el potenciómetro POT como se puede ver en el anexo SIMULADOR CARDIACO, y esto nos hará variar la frecuencia cardíaca.

El puerto 0 sirve para indicar al microcontrolador si la señal que se desea es NORMAL o es una PATOLOGIA. En caso de ser una patología se podrá seleccionar por medio de un interruptor doble cual de las tres patologías se desea, TAQUICARDIA, BRADUCARDIA o ARRITMIA. Específicamente se ha usado:

- P0.0 Para indicar si es NORMAL o PATOLOGIA
- P0.1 Para indicar la PATOLOGIA 1
- P0.2 Para indicar la PATOLOGIA 2

El pin 31 que es EA/Vpp el habilitador de dirección externa. Este pin está a +Vcc para indicar que solo se ejecuta el programa de la memoria interna.

El pin 9 es el RESET, que es una botonera normalmente abierta mas una circuitería con resistencias y capacitores, para eliminar los rebotes. Esta botonera al pulsarla hace inhibir la señal cardíaca.

Los pines 18 y 19 van conectados a un cristal de oscilación de 15 MHz acompañados de 2 capacitores de 10 nF.

4.2.2 Conexión a la memoria

La EPROM 27C256 tiene una capacidad de 32 KB, por lo tanto posee una barra de 15 líneas de dirección, y una barra de 8 líneas de datos. De las 15 líneas de dirección solo 8 van al microcontrolador, específicamente al puerto 1. Las otras 7 líneas van a +Vcc, es decir se considera a la parte alta como 1's lógicos. La barra de 8 líneas de datos O0 a O7 van conectados al DAC.

Los datos digitales de la señal cardíaca se la obtiene mediante el CSC, los datos binarios se colocan en forma hexadecimal y se lo graba en disquetes. El CSC me permite hacer esto ya que tiene una interface con la computadora, y me genera el archivo en forma digital binaria luego se la debe convertir a hexadecimal para grabarlo en la EPROM, de igual forma que grabamos el microcontrolador.

El pin 20 de la EPROM es el CE que es la entrada de habilitación del circuito y va conectado a GND.

El pin 22 de la EPROM es el OE es la entrada que habilita las salidas y se emplea para controlar los buffers de salida de datos, los que permiten al dispositivo pueda conectar al canal de datos del microcontrolador sin conflicto por el canal.

El pin 1 de la EPROM es el Vpp que es el voltaje especial de programación que se encuentra conectado a +Vcc debido aunque la usamos en modo de lectura después de grabada.

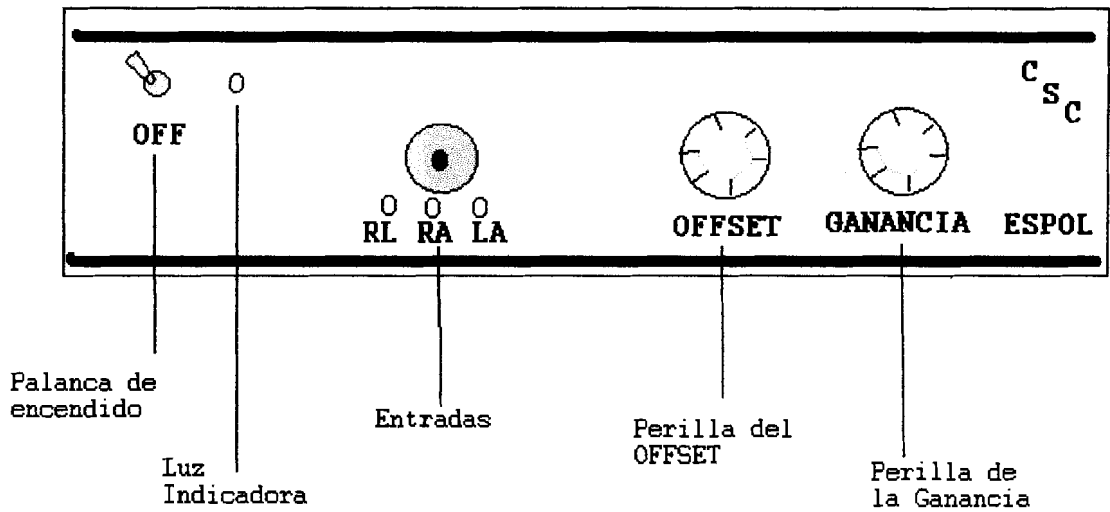
4.2.3 Conexión del DAC

El convertidor utilizado en nuestro circuito es el DAC0807, que es un convertidor de 8 bits con un error de escala completa de $\pm 0.19\%$, que utiliza una red en escalera R-2R.

La salida de esta conversión no es voltaje sino mas bien corriente y se la amplifica por medio de un amplificador operacional $\mu A741$, cuyas características ya se mencionaron en el capítulo III.

CAPITULO V
MANUAL DEL USUARIO

5.1 Manual del usuario del CSC



5.1.1 Guia General.

Figura 5.1 Vista frontal del CSC

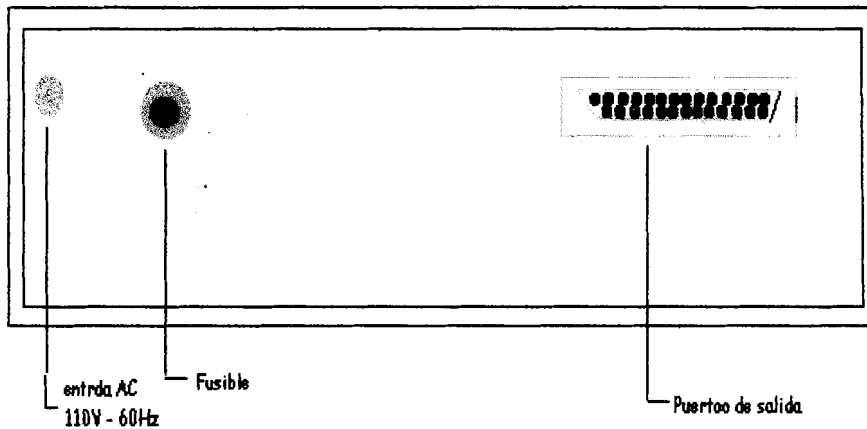


Figura 5.2 Vista posterior del CSC

5.1.2 Función de las botoneras.

1. **La palanca de encendido.** La palanca de encendido tiene dos posiciones on y off con que se enciende o se apaga el capturador de señal cardíaca, respectivamente.
2. **Luz indicadora.** Indica que el capturador de señal cardíaca se encuentra en funcionamiento.
3. **La perilla de OFFSET.** Esta perilla sirve para eliminar la distorción en la señal debido al voltaje OFFSET presente en la entrada.

4. **La perilla de ganancia.** Esta perilla sirve para agustar la ganancia de la señal a capturarse, sirve para una mejor visualización de la onda, al aumentar la amplitud.
5. **Puerto de entrada.** Es por donde ingresamos la señal que se desea capturar.
6. **Puerto de salida.** Es por donde la señal a capturar sale del capturador cardiaco hacia el computador., aqui se conecta el cable con el conector DB25 al puerto paralelo del computador.
7. **Entrada AC.** Es por donde ingresa la señal de alimentación al capturador de señal cardiaca, debe ser de 110V a 60Hz.

5.1.3 Requerimientos

- Voltaje de alimentacion de 110V a 60hz
- Rango de señales de entrada (a capturarse) en mV
- PC desde 386 Mhz minimo
- Copilador de lenguaje C (borlandc)
- Electrodo de buena calidad y en buen estado (mismo fabricante)

5.1.4 Manejo del CSC (hardware)

El manejo del captador de onda cardica es muy sencillo debido a los pocos botones de control. Antes de encender el dispositivo tenemos que tener precaución de asegurar que el voltaje de alimentación sea de 110V. Que los conectores tanto de entrada como de salida este bien conectados.

Cuando encendemos el aparato (palanca en posición ON) la luz indicadora se prendera y se mantendra asi mientras este prendido el captador cardiaco. Las botoneras tanto de OFFSET y de GANANCIA nos ayudan a obtener una señal idoneas para una mejor visualización. La forma de onda puede se vista en el monitor de su computador gracias a un software que hace esto posible.

5.1.5 Presentación. (software)

5.1.5.1 Introducción.

Como ya lo mencionamos anteriormente para poder ingresar la señal al computador esta debe ser digitalizada, pero el computador maneja esta información gracias a un software desarrollado en lenguaje C, este permite que el usuario observe en tiempo real la señal que se esta muestreando, ademas tiene las opciones de: guardar, congelar y reproducir la señal cardiaca muestreada. El programa se puede ver en el anexo 3.

Para acceder a la aplicación que permite visualizar, y capturar la señal, debemos seguir los siguientes pasos:

1. Abrir el explorador de windows.
2. Luego hacemos click en la carpeta borlandc y nos situamos en el subdirectorio bin ver figura 5.3

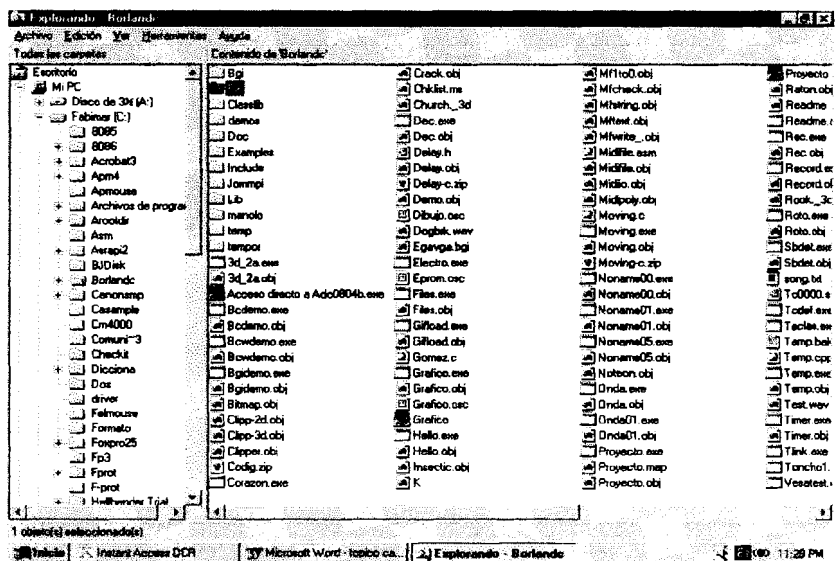


Figura 5.3 La pantalla del explorador de windows

la PC y este puede ser variado por el usuario de acuerdo a sus necesidades en un rango de 2 a 40, normalmente se encuentra en 12 , la velocidad de muestreo maxima será 2 y la más lenta 40.

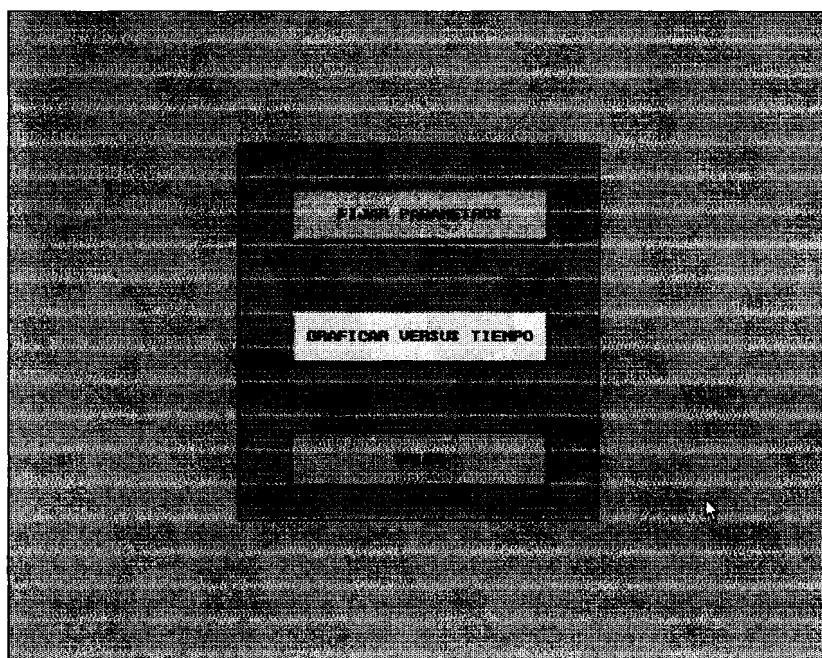


Figura 5.5 Pantalla del menú principal de Adc0804

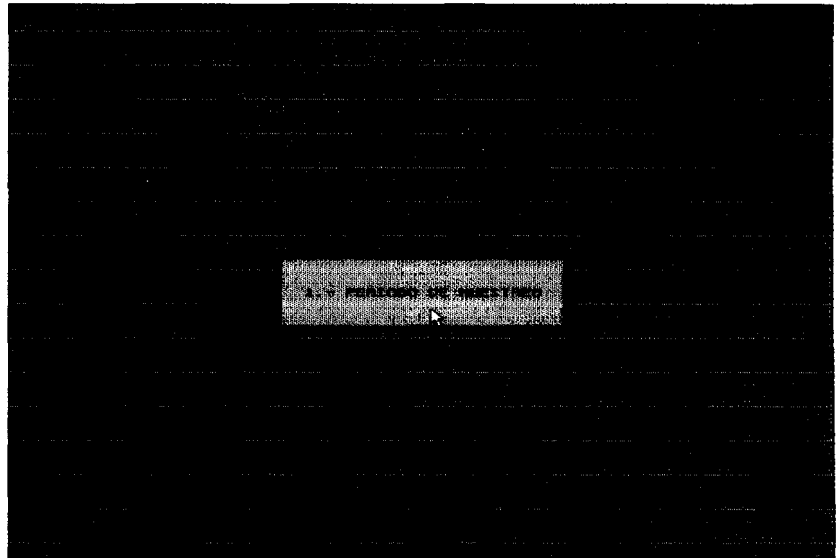


Figura 5.6 Abriendo periodo de muestreo para cambiar el parametro

Para variar la velocidad de muestreo de la señal hacemos click en la barra fijar parametros o presionamos la tecla P, y en la pantalla aparecera una nueva ventana (ver figura 5.6), hacemos click en la barra periodo de muestreo y en la pantalla aparecera un cuadro pidiendo el nuevo parametro de muestreo, por ejemplo si queremos poner el parametro en 12 tenemos que escribir en el cuadro el número 12 y luego damos un ENTER, teniendo en cuenta que el rango variable esta entre 2 y 40.

5.1.5.3 Muestreo de la señal.

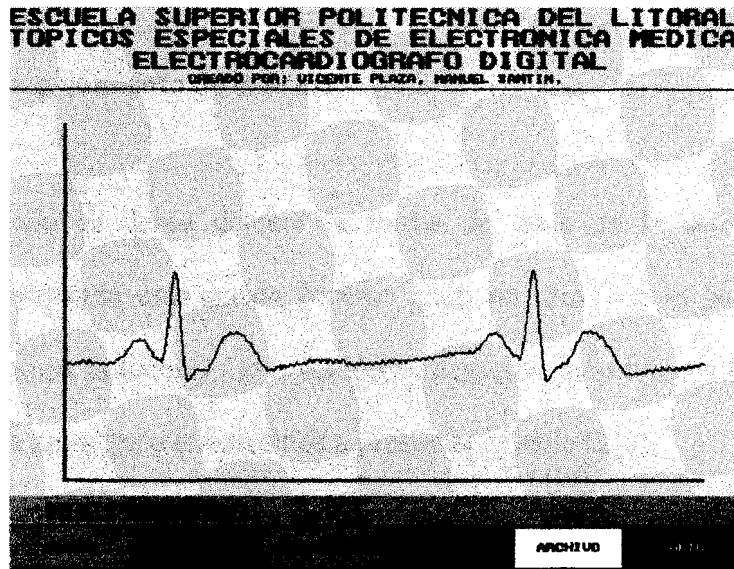


Figura 5.7 Muestreo de la señal del simulador cardiaco digital

Una vez que la palanca de encendido del CSC, esta en la posición On, y el terminal de salida esta debidamente conectado y hemos seguido los pasos del 1 al 4 de la sección 5.1.5.1, y tenemos en la pantalla las tres opciones, como se muestra en la figura 5.5, hacemos clip en la barra graficar versus tiempo, o presionamos la tecla G, y la pantalla nos mostrara una ventana como se muestra en la figura 5.7

En la figura 5.7 se puede observar que el CSC esta muestreando una señal cardiaca proveniente del simulador de ritmo cardiaco digital, ademas

también aparecen en la parte inferior de la ventana las opciones, grabar, parar, reproducir, congelar, archivo y salir.

5.1.5.4 Congelamiento de la señal.

Cuando se desea detener la forma de onda de la señal que se está muestreando esto puede hacerse con un clic en la barra congelar o presionar la tecla C, esta opción se desactiva si volvemos a hacer clic en la barra congelar o si volvemos a presionar la tecla C.

Esto es importante cuando se quiere examinar más detenidamente la forma de onda de la señal muestreada.

5.1.5.5 Grabación de la señal.

Si deseamos guardar una o varias señales, esto no es problema para el capturador de señales, pues es capaz de guardar un gran número de señales, en realidad su capacidad de almacenamiento depende de la memoria del computador.

Para grabar la señal que se muestrea hacemos clic en la barra guardar o presionamos la tecla G, en la parte inferior de la ventana aparecerá la palabra grabando en vez de muestreando, para terminar la grabación

hacemos clic en la barra parar o presionamos la tecla P y en la pantalla se pedira el nombre del archivo que contendra a la señal, ademas podemos ingresar la fecha como se muestra en la figura 5.8.El programa permite que se pueda utilizar una gran cantidad de caracteres para asignar los nombres a los archivos que continen las formas de las señales capturadas.

Este archivo se almacena en una base de datos donde estara a la disposición del usuario cuando requiera observar.

ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL
TOPICOS ESPECIALES DE ELECTRONICA MEDICA
ELECTROCARDIOGRAMAS DIGITAL

REGISTRACION

NOMBRE:
BOLTORA ESPINDOZA

FECHA:

REGISTRO#
3

REGISTRAR

CARGANDO...

ARCHIVO

Figura 5.8 Nombrando un archivo que se a grabado

En la figura 5.8 podemos observar los espacios que se llenan con los datos (nombre del archivo y fecha), en la parte derecha de la ventana existe un cuadro donde se enumera los archivos grabados, esto no tiene que ser llenado por el usuario, por que el programa lo asigna automaticamente, en este caso el archivo se llama Bolivar Espinoza, fue grabado el 25/11/99 y ocupa el registro 3.

El programa se asegura que la señal grabada sea debidamente nombrada, evitando que salgamos de esta opción sin haber nombrado al archivo como se muestra en la figura 5.9

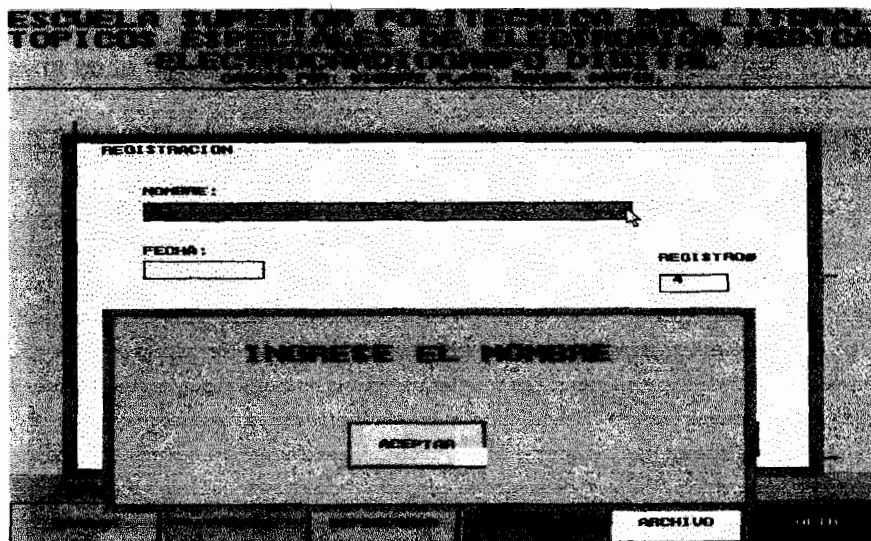


Figura 5.9 Aviso de que no se a nombrado el archivo grabado

Una vez que se ha nombrado al archivo que contiene la señal que se guardo, la pantalla nos vuelve a mostrar la ultima señal que se grabo,

estando en esta opción podemos ver la secuencia de la señal cuantas veces sea necesario haciendo clip en la barra reproducir, que graficara la onda desde el comienzo de la grabación hasta el final de la misma.

5.5.6 Búsqueda de archivos

Seguimos los pasos de 1 al 4 de la sección 5.1.5.1, luego escogemos la opción graficar versus tiempo y en la pantalla aparecera una ventana como se muestra en la figura 5.7, pero sin la señal que ahí se ve, en esta ventana en la parte inferior encontramos 6 opciones, hacemos clip en la barra de archivos, y lo que tendremos en la pantalla sera una ventana como se muestra en la figura 5.10

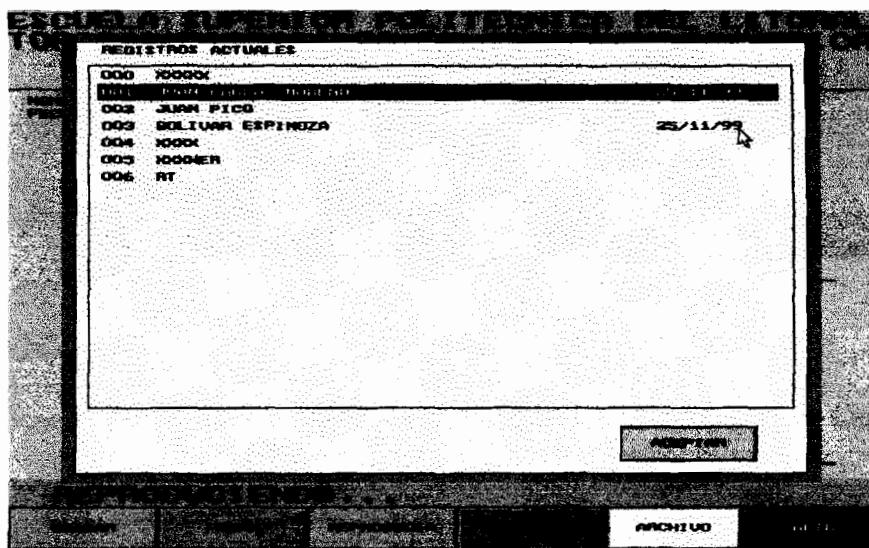


Figura 5.10 buscando el archivo Juan Carlos Moreno

Como se puede ver en la figura 5.10 encontramos un listado de todos los archivos que tenemos guardados en la computadora, en este ejemplo queremos ver lo que se encuentra en el archivo Juan Carlos Moreno, para esto seleccionamos este archivo y damos aceptar o ENTER, como resultado visualizaremos lo que se encuentra grabado en ese archivo como se muestra en la figura 5.11 (en este archivo no existe señal)

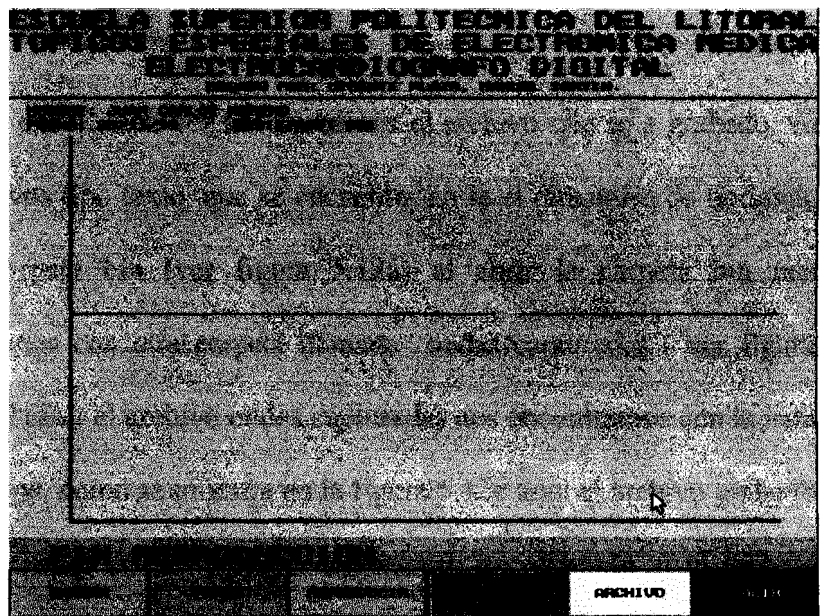


Figura 5.11 Resultado de la búsqueda del archivo Juan Carlos Moreno

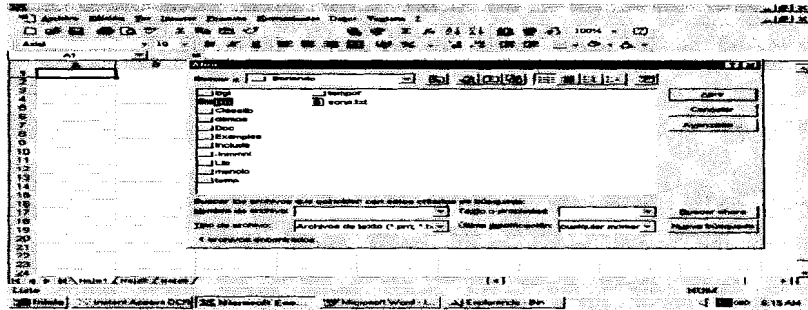
Para salir de esta ventana elegimos la barra salir y nos llevara a la ventana de la figura 5.5 en donde tendremos las tres opciones principales.

5.1.5.7 Para imprimir una señal.

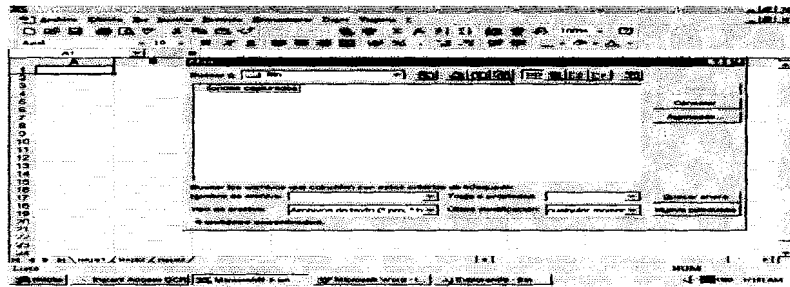
Para imprimir una señal debemos seguir los siguientes pasos:

1. Guardar la señal en un archivo, ver sección 5.1.5.5.
2. Hacemos click en la barra salir, se presenta la ventana que se muestra en la figura 5.5 y volvemos a ser click en la barra salir de esta manera salimos del programa.
3. Abrimos excel , una vez ahí buscamos el archivo que se grabado, en los archivos tipo texto, que se encuentra en la el directorio de borlandc en la carpeta bin (ver figura 5.12a) al abrir la carpeta bin nos encontramos con otra carpeta llamada ondas capturadas (ver figura 3.12b), al abrir el archivo ondas capturadas nos encontramos con la lista de archivos, como se muestra en la figura 5.12c aquí el archivo grabado lo encontramos por el número de registro, por ejemplo si el archivo Juan Carlos Moreno se grabo en el registro 001 dentro de excel lo encontramos como. Ecmtb001.txt. es decir que los archivos en excel estan nombrados de esta manera:

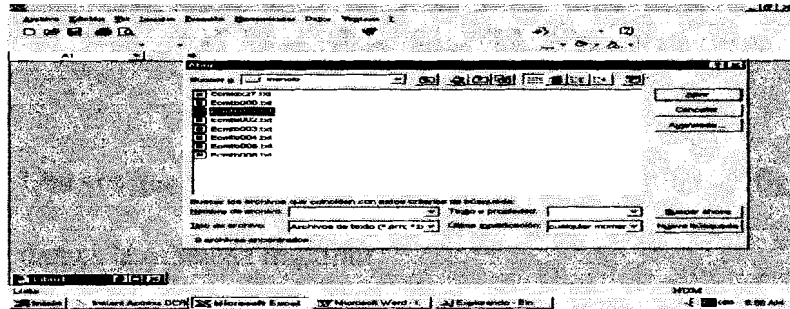
Ecmtb(registro).txt



a) Buscando en el directorio borlanc



b) Encontramos la carpeta ondas capturas



c) Resultado de la búsqueda

Figura 5.12 a,b y c Búsqueda de archivos en excel

4. Al abrir el archivo que nos interesa nos encontramos con una hoja típica de excel en que en la primera columna se encontraran los valores de la señal en forma decimal, como se puede apreciar en la figura 5.12, para obtener el grafico de la señal tal cual fue grabada, damos a los datos de la columna el mismo tratamiento como cualquier dato de una hoja de excel, de la cual se desea obtener información gráfica.

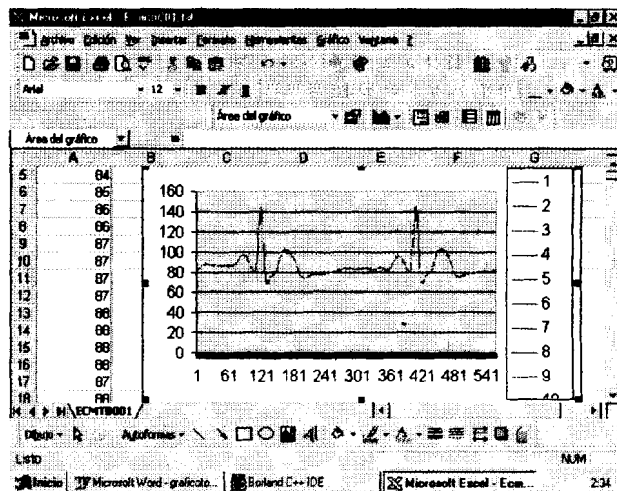


Figura 5.13 La señal cardiaca guardada vista en excel lista para ser impresa

5. Una vez que tenemos la onda en forma satisfactoria (ver figura 5.13), damos la orden de imprimir.

Nota importante al salir de excel el computador seguramente le preguntara si desea actualizar los cambios realizados en el archivo, el usuario debera hacer clip en la opción NO, como se muestra en la figura

5.14 de esta manera nos aseguramos que la onda no sufra ningún cambio en su forma, caso contrario se dañaría la información existente en ese archivo.

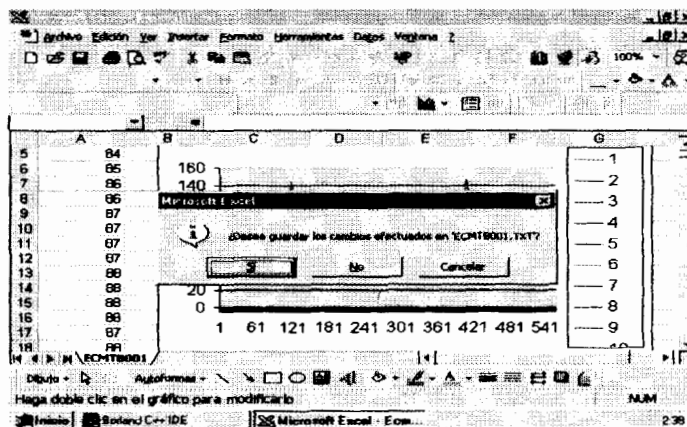


Figura 5.14 Pantalla al salir de excel

5.2 MANUAL DEL USUARIO DEL SIMULADOR CARDIACO

Para darle un funcionamiento óptimo al dispositivo y para no causarle daños al equipo se detalla a continuación, cada uno de los controles que tiene el simulador y sus respectivas precauciones. La numeración siguiente se refiere a la que esta anotada en el anexo llamado SIMULADOR CARDIACO.

1.- INTERRUPTOR DE ENCENDIDO. (ON)

Presione el interruptor para encender el equipo, la indicación "ON" se encuentra a la vista del usuario.

2.- INTERRUPTOR PARA SELECCIONAR ONDA CARDIACA NORMAL (NORMAL).

Con este interruptor se selecciona onda cardíaca normal o una de las tres patologías.

3.- CONTROL DE FRECUENCIA PARA LA ONDA CARDIACA NORMAL (POT).

Este control es un potenciómetro, con este se aumenta la frecuencia cardíaca si giramos hacia la derecha, y disminuimos si giramos del lado contrario. Este control solo se encuentra activo cuando se ha seleccionado el interruptor NORMAL.

4.- CONTROL PARA SELECCIONAR EL TIPO DE PATOLOGIAS (ARR/BRAD/TAQ).

Las patologías que se encuentran disponibles en este simulador son ARRITMIAS (ARR), BRADICARDIA (BRAD) y TAQUICARDIA (TAQ) como se indica en el equipo.

5.- BOTONERA (RESET).

Esta botonera inhibe la salida de la señal cardíaca.

6.- SALIDA (RA).

Esta salida proporciona la salida de la señal cardíaca simulada del brazo derecho. Su color es blanco.

7.- SALIDA (LA).

Esta salida proporciona la salida de la señal cardíaca simulada del brazo izquierdo. Su color es negro.

8.- SALIDA (C).

Esta salida proporciona la salida de la señal cardíaca simulada del pecho. Su color es amarillo.

9.- SALIDA (RL):

Esta salida proporciona la salida de la señal cardíaca simulada de la pierna derecha. Su color es verde y además sirve como referencia para las demás señales.

10.- SALIDA (LL).

Esta salida proporciona la salida de la señal cardíaca simulada de la pierna izquierda. Su color es café.

11.- SALIDA AL OSCILOSCOPIO.

Esta es una salida que puede ser observada en el osciloscopio. El simulador tiene un conector especial que se conecta a un cable ya lista para conecta a las puntas del osciloscopio.

La salida 6, 7, 8, 9 y 10 no pueden ser observadas en el osciloscopio, ya que son señales simuladas de un individuo y estas necesitan ser amplificadas para ser observadas en el osciloscopio.

CAPITULO VI

CONCLUSIONES Y RECOMENDACIONES

6.1 CONCLUSIONES DEL CAPTURADOR DE SEÑAL CARDIACA

Se comprobó que para un sistema de adquisición de datos no hay mejor arreglo electrónico que el amplificador diferencial de instrumentación.

El ruido que interfiere en las señales muestreadas, se puede filtrar por medio de software, con el mismo resultado de un filtro pasabajo.

El trabajo constante con la circuitería del CSC y del simulador de ritmo cardíaco digital, y con el software que se aplicó, afianzo nuestros conocimientos y se ganó valiosas experiencias prácticas.

Se recomienda que al utilizar el capturador de señales cardíacas, se tenga especial cuidado de que los electrodos esten nuevos o en buenas condiciones, para obtener una buena señal.

Se recomienda también, que en caso de que la señal se distorsione, se mueva suavemente la perilla OFFSET hasta eliminar la distorsión.

6.2 CONCLUSIONES DEL SIMULADOR CARDIACO DIGITAL

El objetivo del proyecto se logró satisfactoriamente, la calidad de la señal cardíaca observada en el osciloscopio es muy buena, sin ruido, y se presenta la opción de variar la frecuencia de la señal.

Una de las aplicaciones es usar el simulador para dar clases a estudiantes de Medicina, ya que podrían observar las condiciones normales y patologías que sufren los pacientes, esto ayudaría a dar un diagnóstico acertado.

Otra aplicación, es que el simulador serviría en la reparación de equipos médicos que utilicen como entradas la señal cardíaca, tales como monitores cardíacos y electrocardiógrafos, es decir no se necesitaría obligatoriamente un paciente para probar los equipos.

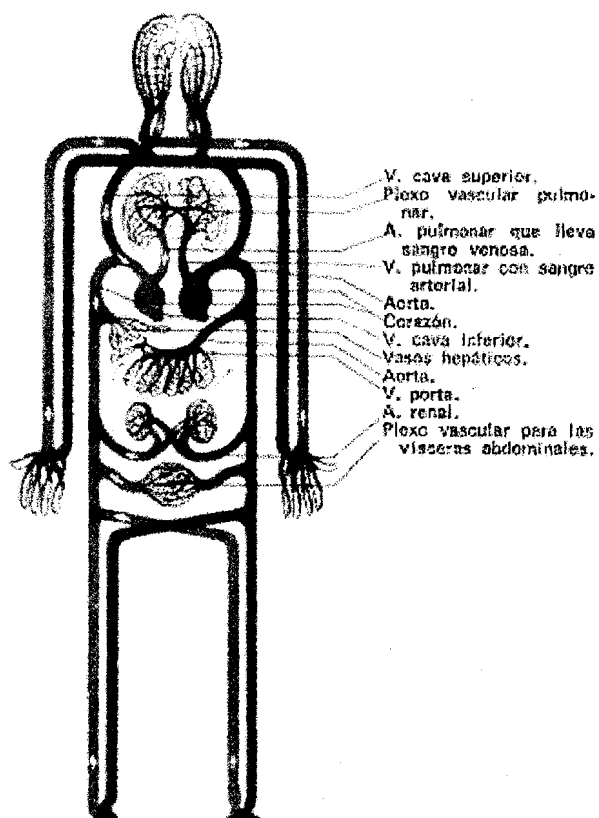
Como el cerebro del simulador es el microcontrolador, tuvimos que aprender a programarlo, ya que este dispositivo tiene su propio lenguaje de programación llamado RCHPSIM, pero que tiene una similitud al TASM aprendido en la materia de Microprocesadores.

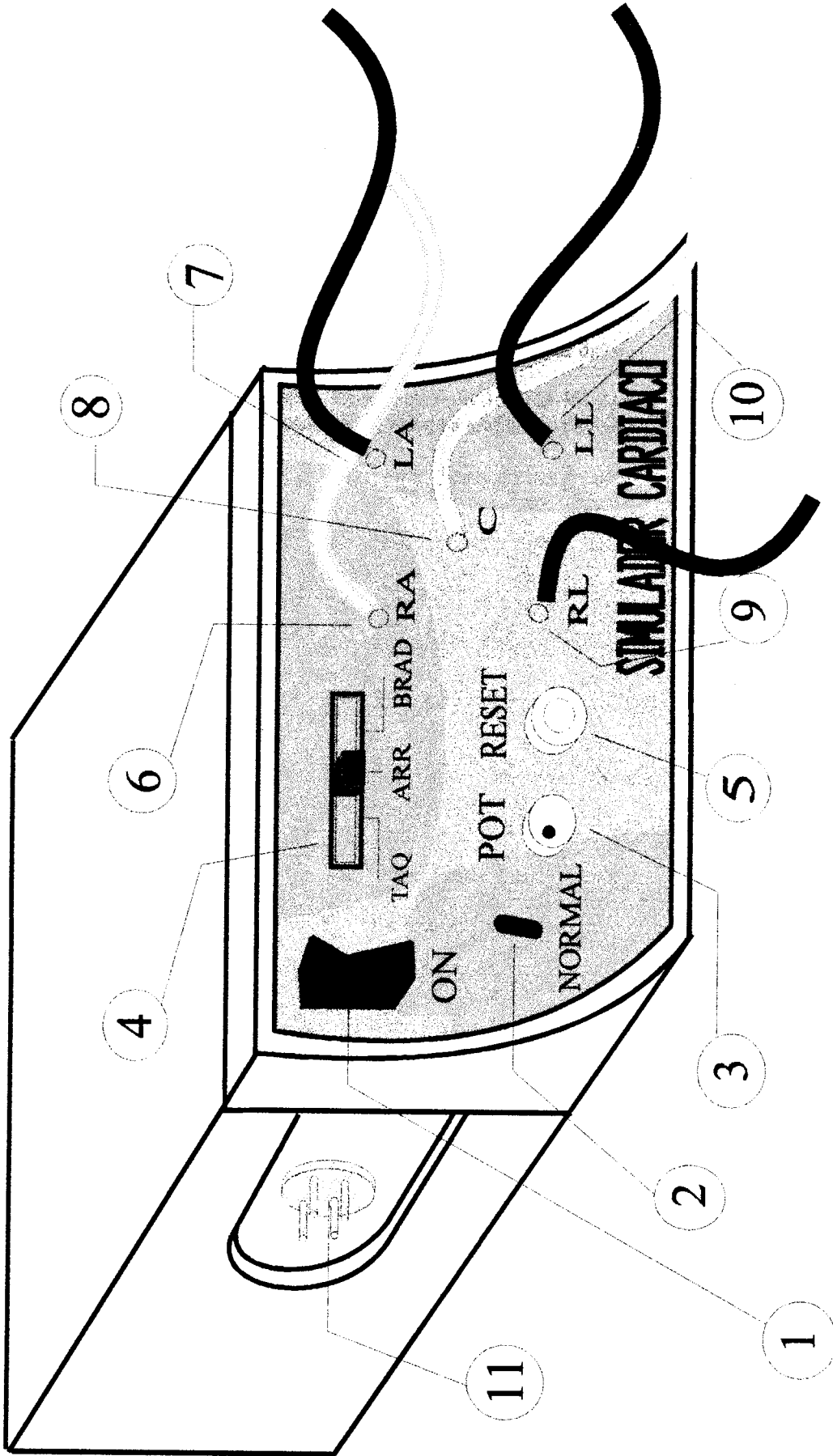
Los circuitos integrados utilizados no son muy fáciles de adquirir debido a su costo económico, esto debido a que el proyecto lo tratamos de reducir en elementos, pero esto hizo que los elementos usados sean más costosos.

Como recomendación se podría añadir una etapa mas en la que este mismo simulador se lo utilice también como electrocardiógrafo con interfase a una computadora personal, debido a que los elementos utilizados en el proyecto son totalmente digitales.

ANEXOS

CIRCULACION DE LA SANGRE





SIMULADOR CARDIACO

ANEXO 3

```
// ADC0804B.CPP
□
// Esta version toma muestras de una entrada de voltaje del ADC0804
□
// y lo grafica en pantalla en funcion del tiempo (entre 0mv y A mv).
□
// Y guarda un maximo de 16384 muestreos en un archivo de datos
□
// en formato de texto numerico decimal, ejemplo: 156 168 199 211 ....
// Permite fijar la amplitud maxima del voltaje a graficar en pantalla.

#include "conio.h"
#include "time.h"
#include "graphics.h"
#include "math.h"
#include "stdlib.h"
#include "string.h"
#include "dos.h"
#include "stdio.h"
#include "ctype.h"
#include "stdarg.h"
#include "alloc.h"
#include "errno.h"

#define UP 328
#define DOWN 336
#define SPACE 32
#define CR 13
#define ESC 27
#define BSPACE 8
#define TAB 9
#define PGDOWN 337
#define PGUP 329
#define DELETE 339

union REGS inr,outr;

char *msg[]={ "FIJAR PARAMETROS","GRAFICAR VERSUS TIEMPO",
              "REALIZAR GRAFICO X-Y","SALIR","1.- PERIODO DE MUESTREO","2.-
NOMBRE DE 1ra VARIABLE",
              "3.- NOMBRE DE 2da VARIABLE","4.- FACTOR CONVERSION 1ra VAR","5.-
FACTOR CONVERSION 2da VAR",
              "ACEPTAR","ACEPTAR",
```

```

        "GRABAR", "PARAR", "REPRODUCIR", "CONGELAR", "ARCHIVO", "SALIR",
        "SI", "NO"
    };

struct ventan {
    int top;
    int bottom;
    int left;
    int right;
} vp;

int colorboton[30];
int datafield[30][2];
int bot[30][4];
char cad1[80], cad2[13];
int maxx;
int maxy;
char nombre[14];
int opc=0;
FILE *FCOD ;
FILE *FCOD1 ;
FILE *FCOD2 ;
void far *buf;
int rax,rbx,rcx,rdx;
int posxx,posyy;
int ventana;
int operacion=1;

void sombras (int,int,int,int);
void boton1 (int,char *);
void aplastar (int,int,int);
void desaplastar (int,int,int);
void ventanamsg( char *mgs);
int ventanamsg2( char *mgs);
void inimodgraf();
void rectang(int x1,int y1,int x2,int y2,int str1,int str2);
int getkey();
int getkey1();
void saveimag(int x1,int y1,int x2, int y2,int a) ;
void restoreimag(int x1,int y1,int x2,int y2,int a) ;
void poneblancos(int x,int y,int lon,int color) ;
void ir(int x1,int y1);
void printstr(char *str2, int size) ;
int longitud_campo(int j);
char *ingresa_cadena(int lon,int size) ;
char *ingresa_cadena1(int lon,int size) ;
int ingresa_numero(int a , int lon) ;
char *ingresa_numero2(int a , int lon) ;
void printchar(char str2, int size) ;
void printnum(int str2, int size) ;
void pantallagraficar();

```



```

void llamaraton(int regax,int regbx,int regcx,int regdx)
{
    union REGS inregs,outregs;
    inregs.x.ax = regax;
    inregs.x.bx = regbx;
    inregs.x.cx = regcx;
    inregs.x.dx = regdx;
    int86(0x33,&inregs,&outregs);
    rax = outregs.x.ax;
    rbx = outregs.x.bx;
    rcx = outregs.x.cx;
    rdx = outregs.x.dx;
}

void ratonini(void)
{
    llamaraton(0,0,0,0);
}

void ratonpres(void)
{
    llamaraton(1,0,0,0);
}

int ratonclick(void)
{
    llamaraton(5,0,0,0);
    return (rbx);
}

void ratonesc(void)
{
    llamaraton(2,0,0,0);
}

int ratonaplas()
{
    llamaraton(3,0,0,0);
    if( rbx & 1 ) return(1);
    else return(0);
}

void muevebot(int i)
{
    ratonesc();
    aplastar (i,1,2);
    delay(200);
    ratonpres();
    while(ratonaplas());
    ratonesc();
    desaplastar (i,1,2);
}

```

```

void muevebot(int i)
{
    ratonesc();
    aplastar (i,1,2);
    delay(200);
    ratonpres();
    while(ratonaplas());
    ratonesc();
    desaplastar (i,1,2);
    ratonpres();
}

/*

void chpalette (void)
{
    int Colors[16][3]={0,0,0, 10,10,10, 5,5,5, 10,10,10, 2,3,7, 2,3,7, 2,3,7, 5,4,1, 5,4,1, 5,4,1, 5,4,1, 5,4,1,
1,4,8, 1,5,8, 3,0,3, 4,4,4};
    int ctes[16][3]= {0,0,0, 2,2,2, 6,6,6, 2, 2, 2, 3,3,3, 3,3,3, 3,3,3, 1,1,1, 1,1,1, 1,1,1, 1,1,1, 1,1,1,
1,1,1, 0,0,0, 1,1,1, 0,0,0};
    int i;
    struct palettetype pal;
    getpalette(&pal);
    for (i=0; i<pal.size; i++)
        setrgbpalette(pal.colors[i], (i+ctes[i][0])*Colors[i][0], (i+ctes[i][1])*Colors[i][1],
(i+ctes[i][2])*Colors[i][2]);
}

*/

// BLACK    DARKGRAY          BLACK    DARKGRAY
// BLUE     LIGHTBLUE         BLUE     LIGHTBLUE
// GREEN    LIGHTGREEN        GREEN    LIGHTGREEN
// CYAN     LIGHTCYAN         CYAN     LIGHTCYAN
// RED      LIGHTRED          RED      LIGHTRED
// MAGENTA  LIGHTMAGENTA      MAGENTA  LIGHTMAGENTA
// BROWN    YELLOW           BROWN    YELLOW
// LIGHTGRAY WHITE          LIGHTGRAY WHITE
void chpalette2 (void)
{
    int Colors[16][3]={0,0,0, 5, 5, 10, 1,12,3, 3, 8, 3, 7,2,1, 7,2,3, 7,3,2, 5,4,1, 5,4,1, 1,4,5, 1,5,1, 1,3,1,
3,1,1, 4,1,6, 1,1,0, 4,4,4};
    int ctes[16][3]= {0,0,0, 2,2,2, 6,6,6, 2, 2, 2, 3,3,3, 3,3,3, 3,3,3, 1,1,1, 1,1,1, 1,1,1, 1,1,1, 1,1,1,
1,1,1, 0,0,0, 1,1,1, 0,0,0};
    int i;
    struct palettetype pal;
    getpalette(&pal);
    for (i=0; i<pal.size; i++)
        setrgbpalette(pal.colors[i], (i+ctes[i][0])*Colors[i][0], (i+ctes[i][1])*Colors[i][1],
(i+ctes[i][2])*Colors[i][2]);
}

```

```

void boton1 (int i , char *msg)
{
    int colorbk=colorboton[i];
    vp.top=bot[i][1];
    vp.bottom=bot[i][3];
    vp.left=bot[i][0];
    vp.right=bot[i][2];

    setfillstyle (SOLID_FILL,0);
    bar (vp.left , vp.top , vp.right , vp.bottom);

    setfillstyle (SOLID_FILL,colorbk);
    bar (vp.left+1 , vp.top+1 , vp.right-1 , vp.bottom-1);

    sombras (3,1,1,2);

    settextstyle(0,HORIZ_DIR,1);
    if (colorbk != BLACK) setcolor (BLACK) ; else setcolor(WHITE);
    settextjustify (CENTER_TEXT,CENTER_TEXT);
    outtextxy ((vp.right+vp.left)/2,(vp.bottom+vp.top)/2,msg);
    settextjustify (LEFT_TEXT,CENTER_TEXT);
    if (colorbk == BLACK) setcolor(BLACK);
}

void sombras (int upcolor,int downcolor,int start,int ancho)
{
    int i;

    setfillstyle (SOLID_FILL,downcolor);

    for (i=0;i<ancho;i++)
        bar (i+start+vp.left , vp.bottom-i-start , vp.right-start, vp.bottom-i-start);

    for (i=0;i<ancho;i++)
        bar (vp.right-i-start , i+start+vp.top , vp.right-i-start , vp.bottom-start);
    setfillstyle (SOLID_FILL,upcolor);

    for (i=0;i<ancho;i++)
        bar (start+vp.left , i+start+vp.top , vp.right-i-start , i+start+vp.top);

    for (i=0;i<ancho;i++)
        bar (i+start+vp.left , start+vp.top , i+start+vp.left , vp.bottom-i-start);
}

void aplastar (int i,int start,int ancho)
{
    vp.top=bot[i][1];
    vp.bottom=bot[i][3];
    vp.left=bot[i][0];
    vp.right=bot[i][2];
    sombras (1,2,start,ancho);
}

```

```

}

void desaplstar (int i,int start,int ancho)
{
    vp.top=bot[i][1];
    vp.bottom=bot[i][3];
    vp.left=bot[i][0];
    vp.right=bot[i][2];
    sombras (3,1,start,ancho);
}

void saveimag(int x1,int y1,int x2, int y2)
{
    unsigned size;
    long int x11=(long int)x1; x1=(int)( x11*maxx)/640 );
    long int y11=(long int)y1; y1=(int)( y11*maxy)/640 );
    long int x22=(long int)x2; x2=(int)( x22*maxx)/640 );
    long int y22=(long int)y2; y2=(int)( y22*maxy)/640 );
    ratonesc();
    size = imagesize(x1, y1, x2, y2);
    if ( (buf = farmalloc(size)) == NULL)
    {
        closegraph();
        printf("Error: not enough heap space in save_screen().\n");
        exit(1);
    }
    getimage(x1, y1, x2, y2, buf);
    ratonpres();
}

void restoreimag(int x1,int y1,int x2,int y2)
{
    long int x11=(long int)x1; x1=(int)( x11*maxx)/640 );
    long int y11=(long int)y1; y1=(int)( y11*maxy)/640 );
    long int x22=(long int)x2; x2=(int)( x22*maxx)/640 );
    long int y22=(long int)y2; y2=(int)( y22*maxy)/640 );
    ratonesc();
    putimage(x1, y1, buf, COPY_PUT);
    farfree(buf);
    ratonpres();
}

void ir(int x1,int y1){
    long int x11=(long int)x1; x1=(int)( x11*maxx)/640 );
    long int y11=(long int)y1; y1=(int)( y11*maxy)/640 );
    moveto(x1,y1);
}

void printstr(char *str2, int size) {
    char buffer[100];
    ratonesc();
    sprintf(buffer,str2);
}

```

```

        if (reg2.bandera!='A'){
            paciente[b].indice[0]=reg2.indice[0];
            paciente[b].indice[1]=reg2.indice[1];
            paciente[b].indice[2]=reg2.indice[2];
            paciente[b].indice[3]=0;
            strcpy(paciente[b].nombre , reg2.nombre);
            strcpy(paciente[b].fecha , reg2.fecha);
            b++;
        }
    }
    a++;
}
fclose(FCOD1);
numpacientes=b;
if (b==0) { // no hay registro valido.
    ventanamsg("NO HAY ARCHIVO VALIDO");
    return(0);
}

ratonesc();

rectang(40,30,600,560,BLUE,BLUE);
rectang(50,40,590,550,WHITE,BLACK);

linea(60,68,60,472);
linea(580,68,580,472);
linea(60,68,580,68);
linea(60,472,580,472);

ir(70,50);printstr("REGISTROS ACTUALES",1);
boton1 (9,msg[9]); //BOTON ACEPTAR.

cont1=numpacientes;
cont3=0;
cont2=cont3;
lin=0;
while ( cont2<cont1 && lin<20 ) {
    ir( 70 , 80+20*lin);
    printstr1(paciente[cont2].indice ,1);
    ir( 110 , 80+20*lin);
    printstr1(paciente[cont2].nombre ,1);
    ir( 480 , 80+20*lin);
    printstr1(paciente[cont2].fecha ,1);
    cont2++;
    lin++;
}
pantselected=0;
cambiarectang1(pantselected+2,pantselected+2);
ratonpres();

do {
    ventana=5;
    opt=getkey1();

```

```

        settextstyle(0,HORIZ_DIR,size);
        outtext(buffer);
        settextstyle(0,HORIZ_DIR,1);
        ratonpres();
    }

void printstr1(char *str2, int size) {
    char buffer[100];
    sprintf(buffer,str2);
    settextstyle(0,HORIZ_DIR,size);
    outtext(buffer);
    settextstyle(0,HORIZ_DIR,1);
}

void printchar(char str2, int size) {
    char buffer[100];
    //    ratonesc();
    sprintf(buffer,"%c",str2);
    settextstyle(0,HORIZ_DIR,size);
    outtext(buffer);
    settextstyle(0,HORIZ_DIR,1);
    //    ratonpres();
}

void printnum(int str2, int size) {
    char buffer[100];
    ratonesc();
    sprintf(buffer,"%d",str2);
    settextstyle(0,HORIZ_DIR,size);
    outtext(buffer);
    settextstyle(0,HORIZ_DIR,1);
    ratonpres();
}

void printlong(long str2, int size) {
    char buffer[100];
    //ratonesc();
    sprintf(buffer,"%ld",str2);
    settextstyle(0,HORIZ_DIR,size);
    outtext(buffer);
    settextstyle(0,HORIZ_DIR,1);
    //ratonpres();
}

void printlong1(long str2, int size) {
    char buffer[100];
    //ratonesc();
    sprintf(buffer,"%ld",str2);
    settextstyle(0,VERT_DIR,size);
    outtext(buffer);
    settextstyle(0,HORIZ_DIR,1);
    //ratonpres();
}

```

```

void printdouble(double str2, int size) {
    char buffer[100];
    //ratonesc();
    sprintf(buffer, "%0.11f", str2);
    settextstyle(0, HORIZ_DIR, size);
    outtext(buffer);
    settextstyle(0, HORIZ_DIR, 1);
    //ratonpres();
}

void printdouble2(double str2, int size) {
    char buffer[100];
    //ratonesc();
    sprintf(buffer, "%0.21f", str2);
    settextstyle(0, HORIZ_DIR, size);
    outtext(buffer);
    settextstyle(0, HORIZ_DIR, 1);
    //ratonpres();
}

void printdouble1(double str2, int size) {
    char buffer[100];
    //ratonesc();
    sprintf(buffer, "%0.21f", str2);
    settextstyle(0, VERT_DIR, size);
    outtext(buffer);
    settextstyle(0, HORIZ_DIR, 1);
    //ratonpres();
}

void ventanamsg( char *mgs) {
    char c;
    int i,j,aux;
    saveimag(70,360,550,600);
    rectang(70,360,550,600,BLUE,BLUE);
    rectang(77,367,543,593,CYAN,BLACK);
    ir( 310-8*strlen(mgs) , 415 );
    printstr(mgs,2);
    setcolor(BLACK);
    boton1(10,msg[10]);
    delay(500);
    aux=ventana;
    ventana=2;
    do
        c=getkey1();
    while( c!=CR);
    restoreimag(70,360,550,600);
    ventana=aux;
    setcolor(WHITE);
}

int ventanamsg5( int indice ) {

```

```

char c;
int i,j,aux;
    saveimag(70,360,550,600);
    ratonesc();
    rectang(70,360,550,600,BLUE,BLUE);
    rectang(77,367,543,593,CYAN,BLACK);
    ir( 160 , 410 );
    printstr1("ESTA SEGURO QUE",2);
    ir( 160 , 440 );
    printstr1("DESEA ELIMINAR",2);
    ir( 160 , 470 );
    printstr1("EL REGISTRO NUM.",2);
    printnum(indice,2);
    printstr1("?",2);
    boton1(17,msg[17]);
    boton1(18,msg[18]);
    ratonpres();
    delay(500);
    aux=ventana;
    ventana=6;
    do
        c=getkey1();
        while( c!='S' && c!='N' && c!=27 );
        restoreimag(70,360,550,600);
        ventana=aux;
        setcolor(WHITE);
        if (c=='S') return(1);
        return(0);
}

int ventanamsg2( int a, int b, char *mgs) {
    int numero;
    rectang(a , b , a+510 , b+120 , BLACK , BLACK);
    rectang(a+10 , b+10 , a+500 , b+110 , LIGHTMAGENTA , BLACK );
    ir( a +30, b +20 );
    printstr(mgs,1);
    do {
        poneblancos( a+ 160 , b+55 , 15, LIGHTMAGENTA );
        poneblancos( a+ 160 , b+65 , 15, LIGHTMAGENTA );
        ir(a +160,b +60 );
        numero=ingresa_numero(4,2);
    } while ( (numero>1000)||((numero<10) ));
    return(numero);
}

char *ventanamsg3( int a, int b, char *mgs) {
    char numero[45];
    rectang(a , b , a+510 , b+200 , BLUE , BLUE );
    rectang(a+10 , b+10 , a+500 , b+190 , CYAN , BLACK );
    ir( a +30, b +50 );
    printstr(mgs,1);
    do {
        poneblancos( a+ 160 , b+140 , 30, CYAN );

```



```

        poneblancos( a+ 160 , b+150 , 30, CYAN ) ;
        ir(a +25 , b +140 );
        strcpy(numero,ingresa_cadena1(30,2));
    } while ( numero[0]!=' '||(numero[0]!='.') );
    return(numero);
}

double ventanamsg4( int a, int b, char *mgs) {
    float numero;
    char numero1[45];
    rectang(a , b , a+510 , b+120 , BLACK , BLACK);
    rectang(a+10 , b+10 , a+500 , b+110 , LIGHTMAGENTA , BLACK );
    ir( a +30, b +20 );
    printstr(mgs,1);
    do {
        poneblancos( a+ 160 , b+55 , 15, LIGHTMAGENTA ) ;
        poneblancos( a+ 160 , b+65 , 15, LIGHTMAGENTA ) ;
        ir(a +160,b +60 );
        strcpy(numero1,ingresa_numero2(8,2));
        numero=atoff(numero1);
    } while ( (numero>9999.0)|| (numero<=0.0) );
    return((double)numero);
}

int getkey() {
    int key=0;
    key=getch();
    if (key==0) {key=getch();key=key+256;}
    return(key);
}

int dentrorectang( int a , int b , int c , int d ){
    if ( ( a<=posxx )*( c>=posxx )*( b<=posyy )*( d>=posyy ) ) return(1);
    else return(0);
}

int dentrorectang1( int a , int b , int c , int d ){
    if ( ( a<=rcx )*( c>=rcx )*( b<=rdx )*( d>=rdx ) ) return(1);
    else return(0);
}

void rectang(int x1,int y1,int x2,int y2,int str1,int str2){
    long int x11=(long int)x1; x1=(int)( x11*maxx)/640 );
    long int y11=(long int)y1; y1=(int)( y11*maxy)/640 );
    long int x22=(long int)x2; x2=(int)( x22*maxx)/640 );
    long int y22=(long int)y2; y2=(int)( y22*maxy)/640 );
    //ratonesc();
    setfillstyle(SOLID_FILL,str1);
    bar(x1,y1,x2,y2);
    setcolor(str2);
    rectangle(x1,y1,x2,y2);
    //ratonpres();
}

```

```

}

void linea(int x1,int y1,int x2,int y2){
    long int  x11=(long int)x1; x1=(int)( x11*maxx)/640 );
    long int  y11=(long int)y1; y1=(int)( y11*maxy)/640 );
    long int  x22=(long int)x2; x2=(int)( x22*maxx)/640 );
    long int  y22=(long int)y2; y2=(int)( y22*maxy)/640 );
    line(x1,y1,x2,y2);
}

void inimodgraf(){
    int gdriver = DETECT, gmode, errorcode;
    initgraph(&gdriver, &gmode, "");
    errorcode = graphresult();
    if (errorcode != grOk)
    {
        printf("Graphics error: %s\n", grapherrormsg(errorcode));
        printf("Press any key to halt.");
        getch();
        exit(1);
    }
}

void poneblancos(int x,int y,int lon,int color) {
    int xx;
    setcolor(color);
    ir(x,y);
    for(xx=1 ; xx<=lon ; xx++) printchar(219,1);
    setcolor(BLACK);
}

void poneblancos2(int x,int y,int lon,int color) {
    int xx;
    setcolor(color);
    ir(x,y);
    for(xx=1 ; xx<=lon ; xx++) printchar(219,2);
    setcolor(BLACK);
}

char *ingresa_cadena1(int lon,int size) {
    int car;
    char alf[45];
    int i=0,flag=0;
    alf[0]='\0';
    do {
        do {
            car=SPACE;
            car=getkey();
            if ( (car==BSPACE && i==0)|| (car==CR && i==0)||
                (!(car==BSPACE)&& i>=lon)&&!( (car==CR)&& i>=lon) ) flag=1;
            else flag=0;
        } while (car>200 || car==ESC || car==TAB || flag );
        if (car!=CR && car !=BSPACE) {

```

```

        car= toupper(car);
        printchar(car,size);
        alf[i]=car; i++;
    }
    if (car==BSPACE) {
        settextstyle(0,HORIZ_DIR,size);
        moveto( getx()-textwidth("H") , gety() );
        setcolor(CYAN);
        printchar(219,size);
        setcolor(BLACK);
        settextstyle(0,HORIZ_DIR,size);
        moveto( getx()-textwidth("H") , gety() );
        alf[i-1]='\0';i--;
    }
} while (car!=CR && i<=lon );
alf[i]='\0';
return(alf);
}

char *ingresa_cadena(int opt , int lon,int size) {
int car,aux;
char alf[80];
int i=1,flag=0;
alf[0]=opt ;
printchar(opt ,size);
do {
    do {
        car=SPACE;
        aux=ventana;
        ventana=9;
        car=getkey1();
        ventana=aux;
        if ( (car==BSPACE && i==0)||(car==CR && i==0)||
            (!(car==BSPACE)&& i>=lon)&&!(car==CR)&& i>=lon) ) flag=1;
        else flag=0;
    } while (car<5 || car>200 || car==ESC || car==TAB || flag );
    if (car!=CR && car !=BSPACE) {
        car= toupper(car);
        printchar(car,size);
        alf[i]=car; i++;
    }
    if (car==BSPACE) {
        settextstyle(0,HORIZ_DIR,size);
        moveto( getx()-textwidth("H") , gety() );
        setcolor(WHITE);
        printchar(219,size);
        setcolor(BLACK);
        settextstyle(0,HORIZ_DIR,size);
        moveto( getx()-textwidth("H") , gety() );
        alf[i-1]='\0';i--;
    }
} while (car!=CR && i<=lon );
alf[i]='\0';

```

```

    return(alf);
}

char *ingresa_numero1(int opt, int lon ,int size) {
int car,aux;
char alf[80];
int i=1,flag=0;
alf[0]=opt ;
printchar(opt ,size);
do {
    do {
        car=SPACE;
        aux=ventana;
        ventana=9;
        car=getkey1();
        ventana=aux;
        if ( (car==BSPACE && i==0)|| (car==CR && i==0)||
            (!(car==BSPACE)&& i>=lon)&&!(car==CR)&& i>=lon) ) flag=1;
        else flag=0;
    } while (
        !((car>=48)*(car<=57)) &&
        !(car==BSPACE) && !(car==CR) || flag );
    if (car!=CR && car !=BSPACE) {
        car= toupper(car);
        printchar(car,size);
        alf[i]=car; i++;
    }
    if (car==BSPACE) {
        settextstyle(0,HORIZ_DIR,size);
        moveto( getx()-textwidth("H") , gety() );
        setcolor(WHITE);
        printchar(219,size);
        setcolor(BLACK);
        settextstyle(0,HORIZ_DIR,size);
        moveto( getx()-textwidth("H") , gety() );
        alf[i-1]='\0';i--;
    }
} while (car!=CR && i<=lon );
alf[i]='\0';
return(alf);
}

char *ingresa_numero2(int lon ,int size) { //INGRESA UN DECIMAL EN STRING.
int car,aux;
int flagpunto=0;
char alf[80];
int i=0,flag=0;
alf[0]='\0';
do {
    do {
        car=SPACE;
        aux=ventana;
        ventana=9;

```

```

    car=getkey1();
    ventana=aux;
    if ( (car==BSPACE && i==0)|| (car==CR && i==0)||
        (car=='.')&&(flagpunto==1) ||
        (!(car==BSPACE)&& i>=lon)&&(!(car==CR)&& i>=lon) ) flag=1;
    else flag=0;
} while (
    !((car>=48)*(car<=57)) &&
    !(car==BSPACE) && !(car==CR) && !(car=='.' )
    || flag );
if (car=='.' ) flagpunto=1;
if (car!=CR && car !=BSPACE) {
    car= toupper(car);
    printchar(car,size);
    alf[i]=car; i++;
}
if (car==BSPACE) {
    settextstyle(0,HORIZ_DIR,size);
    moveto( getx()-textwidth("H") , gety() );
    setcolor(LIGHTMAGENTA);
    printchar(219,size);
    setcolor(BLACK);
    settextstyle(0,HORIZ_DIR,size);
    moveto( getx()-textwidth("H") , gety() );
    if ( alf[i-1]=='.' ) flagpunto=0;
    alf[i-1]='\0';i--;
}
} while (car!=CR && i<=lon );
alf[i]='\0';
return(alf);
}

```

```

int ingresa_numero(int lon,int size) {
    int car;
    int numero;
    int i=0,flag=0;
    numero=0 ;
    do {
        do {
            car=SPACE;
            car=getkey();
            if ( (car==BSPACE && i==0)|| (car==CR && i==0)||
                (!(car==BSPACE)&& i>=lon)&&(!(car==CR)&& i>=lon) ) flag=1;
            else flag=0;
        } while (
            !((car>=48)*(car<=57)) &&
            !(car==BSPACE) && !(car==CR) || flag );

        if (car!=CR && car !=BSPACE) {
            car= toupper(car);
            printchar(car,size);
            if ( (car>=48)*(car<=57) ) numero=10*numero+car-48;
            i++;
        }
    }
}

```

```

    }
    if (car==BSPACE) {
        settxtstyle(0,HORIZ_DIR,size);
        moveto( getx()-textwidth("H") , gety() );
        setcolor(LIGHTMAGENTA);
        printchar(219,size);
        setcolor(BLACK);
        settxtstyle(0,HORIZ_DIR,size);
        moveto( getx()-textwidth("H") , gety() );
        numero=numero/10 ; i--;
    }
} while (car!=CR && i<=lon);
return(numero);
}

int getkey1() {
int opcion=0,i;
char caracter;
long int x11;
long int y11;
union REGS regs;
regs.h.ah=6;
regs.h.dl=255;
int86(0x21, &regs, &regs);
if ( regs.x.flags & 64 ) opcion=0;
else if ( regs.h.al== 0 ) {
    regs.h.ah=6;
    regs.h.dl=255;
    int86(0x21, &regs, &regs);
    opcion= regs.h.al + 256 ;
}
else { opcion= regs.h.al ; }
if ( opcion!=0 ){
    if (opcion<256) return(toupper(opcion));
    else return(opcion);
}

if ( ratonclick() ) {
x11=(long int)rcx; posxx=(int)( (x11*640)/maxx );
y11=(long int)rdx; posyy=(int)( (y11*640)/maxy );

if (ventana==0){
if( dentrorectang1(bot[0][0],bot[0][1],bot[0][2],bot[0][3]) )opcion=700;
if( dentrorectang1(bot[1][0],bot[1][1],bot[1][2],bot[1][3]) )opcion=701;
// if( dentrorectang1(bot[2][0],bot[2][1],bot[2][2],bot[2][3]) )opcion=702;
if( dentrorectang1(bot[3][0],bot[3][1],bot[3][2],bot[3][3]) )opcion=703;
if (opcion==700) muevebot(0);
if (opcion==701) muevebot(1);
// if (opcion==702) muevebot(2);
if (opcion==703) muevebot(3);
return(opcion);
}
}

```

```

if (ventana==1){
  if( dentrorectang1(bot[4][0],bot[4][1],bot[4][2],bot[4][3]) )opcion=700;
  //if( dentrorectang1(bot[5][0], bot[5][1],bot[5][2],bot[5][3]) )opcion=701;
  //if( dentrorectang1(bot[6][0],bot[6][1],bot[6][2],bot[6][3]) )opcion=702;
  //if( dentrorectang1(bot[7][0],bot[7][1],bot[7][2],bot[7][3]) )opcion=703;
  //if( dentrorectang1(bot[8][0],bot[8][1],bot[8][2],bot[8][3]) )opcion=704;
  if (opcion==700) muevebot(4);
  //if (opcion==701) muevebot(5);
  //if (opcion==702) muevebot(6);
  //if (opcion==703) muevebot(7);
  //if (opcion==704) muevebot(8);
  return(opcion);
}

if ( ventana==2) {
  if( dentrorectang1(bot[10][0],bot[10][1],bot[10][2],bot[10][3]) )opcion=13;
  if (opcion==13) muevebot(10);
  return(opcion);
}

if (ventana==3){
  if( dentrorectang1(bot[11][0],bot[11][1],bot[11][2],bot[11][3]) )opcion=700;
  if( dentrorectang1(bot[12][0],bot[12][1],bot[12][2],bot[12][3]) )opcion=701;
  if( dentrorectang1(bot[13][0],bot[13][1],bot[13][2],bot[13][3]) )opcion=702;
  if( dentrorectang1(bot[14][0],bot[14][1],bot[14][2],bot[14][3]) )opcion=703;
  if( dentrorectang1(bot[15][0],bot[15][1],bot[15][2],bot[15][3]) )opcion=704;
  if( dentrorectang1(bot[16][0],bot[16][1],bot[16][2],bot[16][3]) )opcion=705;
  if (opcion==700) muevebot(11);
  if (opcion==701) muevebot(12);
  if (opcion==702) muevebot(13);
  if (opcion==703) muevebot(14);
  if (opcion==704) muevebot(15);
  if (opcion==705) muevebot(16);
  return(opcion);
}

if ( ventana==4) {
  if( dentrorectang( datafield[0][0], datafield[0][1]-7,
  datafield[0][0]+8*longitud_campo(0), datafield[0][1]+13) )opcion=600;
  if( dentrorectang( datafield[1][0], datafield[1][1]-7,
  datafield[1][0]+8*longitud_campo(1), datafield[1][1]+13) )opcion=601;
  if( dentrorectang1(bot[9][0],bot[9][1],bot[9][2],bot[9][3]) )opcion=700;
  if (opcion==700) muevebot(9);
  return(opcion);
}

if ( ventana==5) {
  for( i=2 ; i<= 21 ; i++ ) {
    if( dentrorectang( datafield[i][0], datafield[i][1]-7,
    datafield[i][0]+8*longitud_campo(i), datafield[i][1]+13) )opcion=600+i;
  }
  if( dentrorectang1(bot[9][0],bot[9][1],bot[9][2],bot[9][3]) )opcion=13;
  if (opcion==13) muevebot(9);
}

```

```

return(opcion);
}

if ( ventana==6) {
if( dentrorectang1(bot[17][0],bot[17][1],bot[17][2],bot[17][3]) )opcion='S';
if( dentrorectang1(bot[18][0],bot[18][1],bot[18][2],bot[18][3]) )opcion='N';
if (opcion=='S') muevebot(17);
if (opcion=='N') muevebot(18);
return(opcion);
}

}
else return(0);
return(0);
}

int tabla1[561];
int tabla2[561];

// tiempo1= (long)(FACTOR)*periodo - DIFERENCIA;
// fijar FACTOR y DIFERENCIA para ajustar el delay "tiempo1".

int FACTOR=4000;
long DIFERENCIA=40000;
int periodo=15;

int PUERTO=0x378;
int punterotab = 1;
int punterotab1 = 21;
int nummuestra=0;

struct registro {
char muestra1[16];
char muestra2[16];
};

struct registro reg[50];

int variableinutil[10];

int muestr3(){
long a;
int c,d;

outportb( PUERTO , 0x20 ); // WR del ADC0804 a bajo. Lee 4 bits MSB.
a=0;
while ( a<=10 ) a++;

outportb( PUERTO , 0x0a0 ); // WR del ADC0804 a alto. Lee 4 bits MSB.
a=0;
while ( a<=500 ) a++;

c=inportb( PUERTO+1 ); // los bits con informacion son 6,5,4,3.

```



```

c= c & 0x78 ;

outportb( PUERTO , 0x0c0 ); // 4 bits LSB.
a=0;
while ( a<=10 ) a++;
d=inportb( PUERTO+1 ); // los bits con informacion son 6,5,4,3.
d= d & 0x78 ;

c= 2*c + d/8 ;
return(c);
}

int muestr33(){ // retorna 0 para 0V. y 255 para 5V.
                // return(128*sin((double)punterotab/(double)100) +128);

long a;
int b,c,d;
outportb( PUERTO+2 , 254 ); // WR del ADC0804 a alto. PIN 3.
a=0;
while ( a<=10 ) a++;
outportb( PUERTO+2 , 255 ); // WR del ADC0804 a bajo.
a=0;
while ( a<=10 ) a++;
outportb( PUERTO+2 , 254 ); // WR del ADC0804 a alto.
a=0;
while ( a<=500 ) a++;

c=inportb( PUERTO );
return(c);

}

#define NUMMUESMAX 100
#define NUMMUESACT 10
int muestras[NUMMUESMAX];
int funcion=4;

int muestr1(){
long a;
int c,d,variaciones,pendiente,pendienteant;
a=0;
for( c=0 ; c < periodo*2 ; c++ ){
    a=a+ muestr3();
    //a=a+ muestr33();
}
a=a/(long)(periodo*2 );

for( c=0 ; c < NUMMUESACT-1 ; c++ ){
    muestras[c]=muestras[c+1];
}
muestras[NUMMUESACT-1] = a;

a=muestras[0];

```

```

// a=muestras[NUMMUESACT-1];

if ( muestras[0] == muestras[1] ) pendienteant = 0;
if ( muestras[0] < muestras[1] ) pendienteant = 1;
if ( muestras[0] > muestras[1] ) pendienteant = -1;
pendiente=pendienteant;
variaciones=0;

for( c=1 ; c < NUMMUESACT-1 ; c++){
  if ( muestras[c] < muestras[c+1] ) pendiente = 1;
  if ( muestras[c] > muestras[c+1] ) pendiente = -1;
  if ( pendiente== 1 && pendienteant== -1 ) variaciones++ ;
  if ( pendiente== -1 && pendienteant== 1 ) variaciones++ ;
  pendienteant=pendiente;
}
if(1) {

//if( variaciones > 4 ) {
  a=0;
  for( c=0 ; c < NUMMUESACT ; c++){
    a=a+muestras[c];
  }
  a=a/(long)(NUMMUESACT);
}

/*

if (funcion==0) return(128*sin((double)nummuestra/(double)43) +128);
if (funcion==1) return(128*sin((double)nummuestra/(double)43)*
                      sin((double)nummuestra/(double)43) +128);
if (funcion==2) return(128*sin((double)nummuestra/(double)30)*
                      sin((double)nummuestra/(double)60) +128);
if (funcion==3) return(128*sin((double)nummuestra/(double)30)*
                      cos((double)nummuestra/(double)60) +128);
if (funcion==4) return(128*sin((double)nummuestra/(double)20)*
                      sin((double)nummuestra/(double)60) +128);
if (funcion==5) return(128*sin((double)nummuestra/(double)20)*
                      cos((double)nummuestra/(double)60) +128);
if (funcion==6) return(128*sin((double)nummuestra/(double)30)*
                      sin((double)nummuestra/(double)45) +128);
if (funcion==7) return(128*sin((double)nummuestra/(double)30)*
                      cos((double)nummuestra/(double)45) +128);

*/
return((int)a);
}

int muestr2(){ // retorna 0 para 15V. y 255 para -15V.
               // return(128*sin((double)punterotab/(double)100) +128);
  long a;
  int b,c,d;
  outportb( PUERTO+2 , 255 ); // START del ADC0808 a bajo.

```

```

// ADDRESS A del ADC0808 a alto.
a=0;
while ( a<=100 ) a++;
outportb( PUERTO+2 , 254 ); // START del ADC0808 a alto.
// ADDRESS A del ADC0808 a alto.
a=0;
while ( a<=5000 ) a++;
outportb( PUERTO+2 , 255 ); // START del ADC0808 a bajo.
// ADDRESS A del ADC0808 a alto.
a=0;
while ( a<=20000 ) a++;

c=0;
for(d=7; d>=0; d--){
  outportb( PUERTO , d );
  a=0;while ( a<=200 ) a++;
  b=inportb( PUERTO+1);
  b=b & 8;
  c=2*c+b;
}
return( c/8 );
}
char *mtoa(long muestra) {
  char cad1[16];
  char cad2[16];
  long num1;
  int car1;
  int i=0;
  // 0 ---> +1500.
  // 255 ---> -1500.
  num1=-((muestra)*3000/255 + 1500);
  ltoa(num1,cad1,10);
  if (num1>=0){
    while( strlen(cad1)<3 ) {
      strcpy(cad2,"0");strcat(cad2,cad1);strcpy(cad1,cad2);
    }
  } else {
    while( strlen(cad1)<4 ) {
      strcpy(cad2,"-");strcat(cad2,cad1);strcpy(cad1,cad2);cad1[1]='0';
    }
  }
  i=strlen(cad1);
  cad1[i+1]=cad1[i ];
  cad1[i ]=cad1[i-1];
  cad1[i-1]=cad1[i-2];
  cad1[i-2]='.';
  /*
  if ( num1<0 ) { cad1[i]='-';i++;
                 num1=-num1; }
  car1= num1/1000+48;
  if (car1 != 48) { cad1[i]=car1; i++; }
  car1= num1/100-(num1/1000)*10+48;
  cad1[i]=car1; i++;

```

```

    cad1[i]='.'; i++;
    car1= num1/10 -(num1/100)*10+48;
    cad1[i]=car1; i++;
    car1= num1 -(num1/10)*10+48;
    cad1[i]='\0';
    */
    return(cad1);
}

double FACTOR1=1.0; //factores de conversion V---> Unidades de Variable.
double FACTOR2=1.0;
char NOMBRE1[45]; //nombre de variables
char NOMBRE2[45];
double MINIMO=-15.0; //rango a graficar.
double MAXIMO=15.0;

char *mtoa1(long muestra) {
    char cad1[16];
    char cad2[16];
    long num1;
    int car1;
    int i=0;
    double aa;
    // 0 ---> +15000.
    // 255 ---> -15000.
    num1=-(muestra)*30000/255 + 15000;
    aa=(double)num1*FACTOR1;
    ltoa((long)aa,cad1,10);
    if (num1>=0){
        while( strlen(cad1)<4 ) {
            strcpy(cad2,"0");strcat(cad2,cad1);strcpy(cad1,cad2);
        }
    } else {
        while( strlen(cad1)<5 ) {
            strcpy(cad2,"-");strcat(cad2,cad1);strcpy(cad1,cad2);cad1[1]='\0';
        }
    }
    i=strlen(cad1);
    cad1[i+1]=cad1[i ];
    cad1[i ]=cad1[i-1];
    cad1[i-1]=cad1[i-2];
    cad1[i-2]=cad1[i-3];
    cad1[i-3]='.';
    return(cad1);
}

char *mtoa2(long muestra) {
    char cad1[16];
    char cad2[16];
    long num1;
    int car1;
    int i=0;
    double aa;

```

```

// 0 ---> +15000.
// 255 ---> -15000.
num1=- (muestra)*30000/255 + 15000;
aa=(double)num1*FACTOR2;
ltoa((long)aa,cad1,10);
if (num1>=0){
    while( strlen(cad1)<4 ) {
        strcpy(cad2,"0");strcat(cad2,cad1);strcpy(cad1,cad2);
    }
} else {
    while( strlen(cad1)<5 ) {
        strcpy(cad2,"-");strcat(cad2,cad1);strcpy(cad1,cad2);cad1[1]='0';
    }
}
i=strlen(cad1);
cad1[i+1]=cad1[i ];
cad1[i ]=cad1[i-1];
cad1[i-1]=cad1[i-2];
cad1[i-2]=cad1[i-3];
cad1[i-3]='.';
return(cad1);
}

struct formato1 {
    char indice[4];
    char nombre[46];
    char fecha[12];
    char bandera;
    char nuevalinea[2];
};

struct formato1 paciente[1000];
struct formato1 reg1;
struct formato1 reg2;
int muestrasgrab[16384];
int numpacientes;

void ir1(int i){
    ir( datafield[i][0] , datafield[i][1] );
}

void ir2(int i){
    ir( datafield[i][0] , datafield[i][1]-20 );
}
int longitud_campo(int i){
    int k;
    if (i==0) k=45; // NOMBRE
    if (i==1) k=11; // FECHA
    if (i>1) k=63; // LINEAS DEL CUADRO ARCHIVO.
    return(k);
}
void rectanging(int i,int color){
    rectang(datafield[i][0] , datafield[i][1]-7 ,

```

```

        datafield[i][0]+8*longitud_campo(i) , datafield[i][1]+13 ,color,BLACK);
        setcolor(BLACK);
    }
    void cambiarectang( int i, int j ) {
        rectanging(i,WHITE);
        ir1(i);
        if (i==0) printstr( reg1.nombre , 1 );
        if (i==1) printstr( reg1.fecha , 1 );
        rectanging(j, GREEN );
        ir1(j);
        if (j==0) printstr( reg1.nombre , 1 );
        if (j==1) printstr( reg1.fecha , 1 );
    }
    int indselected;
    int totalmuestras;
    int cont1,cont2,cont3;

    void rectanging1(int i,int color){
        rectang(datafield[i][0] , datafield[i][1]-7 ,
            datafield[i][0]+8*longitud_campo(i) , datafield[i][1]+13 ,color,WHITE);
        setcolor(BLACK);
    }
    void cambiarectang1( int i, int j ) {
        rectanging1(i,WHITE);
        ir( 70 , 80+20*(i-2) );
        printstr1(paciente[cont3 +i-2].indice ,1);
        ir( 110 , 80+20*(i-2) );
        printstr1(paciente[cont3 +i-2].nombre ,1);
        ir( 480 , 80+20*(i-2) );
        printstr1(paciente[cont3 +i-2].fecha ,1);

        rectanging1(j,BLACK );
        setcolor(WHITE);
        ir( 70 , 80+20*(j-2) );
        printstr1(paciente[cont3 +j-2].indice ,1);
        ir( 110 , 80+20*(j-2) );
        printstr1(paciente[cont3 +j-2].nombre ,1);
        ir( 480 , 80+20*(j-2) );
        printstr1(paciente[cont3 +j-2].fecha ,1);
    }
    void status( char *mgs) {
        // ratonesc();
        poneblancos2(35,576,30, GREEN);
        ir(35,576);
        setcolor(BLACK);
        printstr1(mgs,2);
        // ratonpres();
    }
    void displayar_campos(){

        rectang(40,150,600,560,BLUE,BLUE);
        rectang(50,160,590,550,WHITE,BLACK);
        rectanging(0,WHITE);ir2(0);
    }

```

```

printstr("NOMBRE: ",1);
rectang(1,WHITE);ir2(1);
printstr("FECHA: ",1);
} //displayar_campos

void registracion(int b) {
int a,j,ji,k,i,opt,dig2,dig1,dig0;
double ii,aa,aa1;
int car,existente;
int flag0;

flag0=0;
strcpy(reg1.nombre,""); //45
strcpy(reg1.fecha,""); //11
reg1.bandera='V';
reg1.nuevalinea[0]=13;
reg1.nuevalinea[1]=10;

dig2=b/100 ;
dig1=b/10- (b/100)*10 ;
dig0=b - (b/10)*10 ;
reg1.indice[0]=dig2+48;
reg1.indice[1]=dig1+48;
reg1.indice[2]=dig0+48;
reg1.indice[3]=32;

ratonesc();

displayar_campos();
ir(70,170);printstr1("REGISTRACION",1);
ir(480,300);printstr1("REGISTRO#",1);
rectang(480,320,530,340,WHITE,BLACK);
ir(490,327);printnum(b,1);
boton1 (9,msg[9]); //BOTON ACEPTAR.
rectang(0,GREEN);

ratonpres();

i=0;
do {
ventana=4;
opt=getkey1();
if ( opt==600 ) { // NOMBRE
cambirectang(i,0);i=0;
}
if ( opt==601 ) { // FECHA
cambirectang(i,1);i=1;
}
if ( (i==0)*(isalnum(opt))*(opt<256) ) {
rectang(i,WHITE);
ir1(i);
strcpy(reg1.nombre,ingresa_cadena(opt,longitud_campo(i),1) );
}
}

```

```

        rectanging(i, GREEN);
        ir1(i);
        printstr(reg1.nombre , 1 );
        flag0=1;
    };
    if ( (i==1)*(isalnum(opt))*(opt<256) ) {
        rectanging(i, WHITE);
        ir1(i);
        strcpy(reg1.fecha, ingresa_cadena(opt,longitud_campo(i),1) );
        rectanging(i, GREEN);
        ir1(i);
        printstr( reg1.fecha , 1);
    };
    if (opt==TAB || opt==DOWN || opt==UP ) {
        rectanging(i, WHITE);
        ir1(i);
        if (i==0) printstr( reg1.nombre , 1 );
        if (i==1) printstr( reg1.fecha , 1 );
        if (i==1) i=0 ; else i++;
        rectanging(i, GREEN );
        ir1(i);
        if (i==0) printstr( reg1.nombre , 1 );
        if (i==1) printstr( reg1.fecha , 1 );
    };
    if ( ( opt==700 || opt==13 ) ) {
        if (flag0==0) {
            ventanamsg("INGRESE EL NOMBRE");opt=0;
        }
    }
} while ( opt!=700 && opt !=13 ); //do

} //funcion registracion

int cargar(){

int a,b,j,i,k,i,opt,lin,dig2,dig1,dig0;
int car,existente;
int indice;
int pantsselected;

FCOD1= fopen("ecmidx27.txt","rb+");
if (FCOD1==NULL) { // archivo no existe.
    ventanamsg("NO EXISTE ARCHIVO");
    return(0);
}
clearerr(FCOD1);
rewind(FCOD1);
a=fseek(FCOD1,0 ,SEEK_SET);
a=0;b=0;
while( !feof(FCOD1) && a<1000 ) {
    fread(&reg2, sizeof(reg2), 1, FCOD1);
    if ( !feof(FCOD1) ) {

```



```

if ( ( opt==PGDOWN ) ) {
    if (cont3+20 < cont1){
        cont3=cont3+20;
        cont2=cont3;
        lin=0;
        ratonesc();
        rectang(62,70,578,470,WHITE,WHITE);
        setcolor(BLACK);
        while ( cont2<cont1 && lin<20 ) {
            ir( 70 , 80+20*lin);
            printstr1(paciente[cont2].indice ,1);
            ir( 110 , 80+20*lin);
            printstr1(paciente[cont2].nombre ,1);
            ir( 480 , 80+20*lin);
            printstr1(paciente[cont2].fecha ,1);
            cont2++;
            lin++;
        }
        pantselected=0;
        cambiarectang1(pantselected+2,pantselected+2);
        ratonpres();
    }
}
if ( ( opt==PGUP ) ) {
    if (cont3>0){
        cont3=cont3-20;
        cont2=cont3;
        lin=0;
        ratonesc();
        rectang(62,70,578,470,WHITE,WHITE);
        setcolor(BLACK);
        while ( cont2<cont1 && lin<20 ) {
            ir( 70 , 80+20*lin);
            printstr1(paciente[cont2].indice ,1);
            ir( 110 , 80+20*lin);
            printstr1(paciente[cont2].nombre ,1);
            ir( 480 , 80+20*lin);
            printstr1(paciente[cont2].fecha ,1);
            cont2++;
            lin++;
        }
        pantselected=0;
        cambiarectang1(pantselected+2,pantselected+2);
        ratonpres();
    }
}
if ( ( opt>=602 && opt<=621 ) ) {
    if ( cont3+opt-602 < numpacientes ) {
        cambiarectang1(pantselected+2,opt-600);
        pantselected=opt-602;
    }
}
if ( ( opt==UP ) ) {

```

```

if(pantselected>0){
  cambiarectang1(pantselected+2,pantselected+1);
  pantselected--;
}
else if (cont3>0){
  cont3=cont3-20;
  cont2=cont3;
  lin=0;
  ratonesc();
  rectang(62,70,578,470,WHITE,WHITE);
  setcolor(BLACK);
  while ( cont2<cont1 && lin<20 ) {
    ir( 70 , 80+20*lin);
    printstr1(paciente[cont2].indice ,1);
    ir( 110 , 80+20*lin);
    printstr1(paciente[cont2].nombre ,1);
    ir( 480 , 80+20*lin);
    printstr1(paciente[cont2].fecha ,1);
    cont2++;
    lin++;
  }
  pantselected=19;
  cambiarectang1(pantselected+2,pantselected+2);
  ratonpres();
}
}
if ( ( opt==DOWN ) ) {
  if(cont3+pantselected+1<numpacientes && pantselected<19){
    cambiarectang1(pantselected+2,pantselected+3);
    pantselected++;
  }
  else if (cont3+20 < cont1){
    cont3=cont3+20;
    cont2=cont3;
    lin=0;
    ratonesc();
    rectang(62,70,578,470,WHITE,WHITE);
    setcolor(BLACK);
    while ( cont2<cont1 && lin<20 ) {
      ir( 70 , 80+20*lin);
      printstr1(paciente[cont2].indice ,1);
      ir( 110 , 80+20*lin);
      printstr1(paciente[cont2].nombre ,1);
      ir( 480 , 80+20*lin);
      printstr1(paciente[cont2].fecha ,1);
      cont2++;
      lin++;
    }
    pantselected=0;
    cambiarectang1(pantselected+2,pantselected+2);
    ratonpres();
  }
}
}

```

```

if ( ( opt==DELETE ) ) {
    indice=pantselected+cont3;
    dig2=paciente[indice].indice[0];
    dig1=paciente[indice].indice[1];
    dig0=paciente[indice].indice[2];
    b=100*(dig2-48) + 10*(dig1-48) + dig0-48;

    setcolor(BLACK);
    FCOD1= fopen("ecmidx27.txt","rb+");
    if (FCOD1==NULL) { // archivo no existe.
        ventanamsg("FALLA ECMIDX27.TXT");
    }
    else if (ventanamsg5(b)){
        clearerr(FCOD1);
        rewind(FCOD1);
        a=fseek(FCOD1, (long)b*(long)(sizeof(reg2)), SEEK_SET);
        fread(&reg2, sizeof(reg2), 1, FCOD1);
        reg2.bandera='A';
        a=fseek(FCOD1, (long)b*(long)(sizeof(reg2)), SEEK_SET);
        fwrite(&reg2, sizeof(reg2), 1, FCOD1);
        fclose(FCOD1);
        for(i=indice; i< numpacientes-1 ; i++){
            strcpy(paciente[i].indice , paciente[i+1].indice);
            strcpy(paciente[i].nombre , paciente[i+1].nombre);
            strcpy(paciente[i].fecha , paciente[i+1].fecha);
        }
        numpacientes--;
        cont1--;
        if (numpacientes==0)return(0);
        delay(500);
        if (indice==numpacientes && pantselected==0 && cont3>0){
            cont3=cont3-20;
            cont2=cont3;
            lin=0;
            ratonesc();
            rectang(62,70,578,470,WHITE,WHITE);
            setcolor(BLACK);
            while ( cont2<cont1 && lin<20 ) {
                ir( 70 , 80+20*lin);
                printstr1(paciente[cont2].indice ,1);
                ir( 110 , 80+20*lin);
                printstr1(paciente[cont2].nombre ,1);
                ir( 480 , 80+20*lin);
                printstr1(paciente[cont2].fecha ,1);
                cont2++;
                lin++;
            }
            pantselected=19;
            cambiarectang1(pantselected+2,pantselected+2);
            ratonpres();
        } else {
            cont2=cont3;
            lin=0;

```

```

    ratonesc();
    rectang(62,70,578,470,WHITE,WHITE);
    setcolor(BLACK);
    while ( cont2<cont1 && lin<20 ) {
        ir( 70 , 80+20*lin);
        printstr1(paciente[cont2].indice ,1);
        ir( 110 , 80+20*lin);
        printstr1(paciente[cont2].nombre ,1);
        ir( 480 , 80+20*lin);
        printstr1(paciente[cont2].fecha ,1);
        cont2++;
        lin++;
    }
    if (indice==numpacientes ) pantsselected--;
    cambiarectang1(pantsselected+2,pantsselected+2);
    ratonpres();
}
}
else{ fclose(FCOD1); }
}
if ( ( opt==700 || opt==13 ) ) {
    indice=pantsselected+cont3;
    setcolor(BLACK);
    ir(120,500);printnum(indice,1);
    delay(1000);
    indselected=indice;
}
} while ( opt!=27 && opt!=13 ); //do

if(opt==27 ) return(0);

b=indselected;
dig2=paciente[b].indice[0];
dig1=paciente[b].indice[1];
dig0=paciente[b].indice[2];

strcpy(reg1.nombre , paciente[b].nombre);
strcpy(reg1.fecha , paciente[b].fecha );
strcpy(reg1.indice , paciente[b].indice);

nombre[0]='E';
nombre[1]='C';
nombre[2]='M';
nombre[3]='T';
nombre[4]='B';
nombre[5]=dig2;
nombre[6]=dig1;
nombre[7]=dig0;
nombre[8]='.';
nombre[9]='T';
nombre[10]='X';
nombre[11]='T';
nombre[12]=0;

```

```

FCOD2= fopen(nombre,"rb+");
if (FCOD2==NULL) { //Archivo no existe.
    ventanamsg("NO EXISTE ARCHIVO");
    return(0);
}
clearerr(FCOD2);
rewind(FCOD2);
a=fseek(FCOD2,0 ,SEEK_SET);
a=0;
while ( a< 16384 && !feof(FCOD2) ) {
    dig2=fgetc(FCOD2);
    dig1=fgetc(FCOD2);
    dig0=fgetc(FCOD2);
    b=fgetc(FCOD2);
    if (!feof(FCOD2)) {
        muestrasgrab[a]=100*(dig2-48) + 10*(dig1-48) + dig0-48;
        a++;
    }
}
totalmuestras=a;
fclose(FCOD2);
return(1);
} //cargar

void arraytofile(){
    int a,b,dig2,dig1,dig0;
    char nombre[20];

    FCOD1= fopen("ecmidx27.txt","rb+");
    if (FCOD1==NULL) { // archivo no existe.
        FCOD1= fopen("ecmidx27.txt","wb");
        if (FCOD1==NULL){ perror("fopen"); getkey(); exit(0); }
        fclose(FCOD1);
        delay(30);
        FCOD1= fopen("ecmidx27.txt","rb+");
    }
    clearerr(FCOD1);
    rewind(FCOD1);
    a=fseek(FCOD1,0 ,SEEK_SET);
    a=0;b=0;
    while( !feof(FCOD1) && b==0 ) {
        fread(&reg1, sizeof(reg1), 1, FCOD1);
        if ( !feof(FCOD1) ) {
            if (reg1.bandera=='A'){
                b=a;
            }
        }
        else b=a;
        a++;
    }
    registracion(b);

    a=fseek(FCOD1, (long)b*(long)(sizeof(reg1)) ,SEEK_SET);

```

```

fwrite(&reg1, sizeof(reg1), 1, FCOD1);
fclose(FCOD1);
reg1.indice[3]=0;

dig2=b/100 ;
dig1=b/10- (b/100)*10 ;
dig0=b - (b/10 )*10 ;
nombre[0]='E';
nombre[1]='C';
nombre[2]='M';
nombre[3]='T';
nombre[4]='B';
nombre[5]=dig2+48;
nombre[6]=dig1+48;
nombre[7]=dig0+48;
nombre[8]='.';
nombre[9]='T';
nombre[10]='X';
nombre[11]='T';
nombre[12]=0;
FCOD2= fopen(nombre,"wb");
if (FCOD2==NULL){ perror("fopen"); getkey(); exit(0); }
a=0;
while ( a< nummuestra ) {
    b=muestrasgrab[a];
    dig2=b/100 ;
    dig1=b/10- (b/100)*10 ;
    dig0=b - (b/10 )*10 ;
    fputc( dig2+48 ,FCOD2 ) ;
    fputc( dig1+48 ,FCOD2 ) ;
    fputc( dig0+48 ,FCOD2 ) ;
    fputc( 13 ,FCOD2 ) ;
    //fputc( 10 ,FCOD2 ) ;
    a++;
}
totalmuestras=nummuestra;
fclose(FCOD2);
}

void pantallagraficar(){
    int i,j,a;
    double ii;
    long k;

    ratonesc();
    rectang(1,1,639,556,LIGHTBLUE,LIGHTBLUE);
    rectang(1,556,639,588,GREEN,GREEN);
    setcolor(BLACK);
    //for (i=11;i<=16;i++)
    //{
    // boton1 (i,msg[i]);
    //}
    //printstr("PERIODO DE MUESTRO = ",2);

```

```

//prntnum(periodo,2);
//printstr(" MSEG",2);
ir(1,15);
printstr1("ESCUELA SUPERIOR POLITECNICA DEL LITORAL",2);
ir(1,40);
printstr1("TOPICOS ESPECIALES DE ELECTRONICA MEDICA",2);
ir(110,65);
printstr1("CAPTURADOR DE SEÑAL CARDIACA",2);
ir(158,85);
printstr1("CREADO POR: VICENTE PLAZA, MANUEL SANTIN.",1);
linea(1,95,639,95);
linea(50,135,50,538);
linea(49,135,49,538);
linea(49,538,610,538);
linea(49,539,610,539);

if (totalmuestras!==-1) {
    ir( 15 , 110);
    printstr1("NOMBRE: ",1);
    printstr1(reg1.nombre ,1);
    ir( 15 , 125);
    printstr1("FECHA: ",1);
    printstr1(reg1.fecha ,1);
    ir( 180, 125);
    printstr1("REGISTRO#: ",1);
    printstr1(reg1.indice ,1);
}

//for( i=1 ; i<=4 ; i++ ) {
//    j=140*i;
//    k=(long)j * periodo ;
//    ii=(double)k /1000;
//    ii=ii;
//    linea( j+50 ,554, j+50, 570 );
//    ir(j+10 , 540 );
//    printdouble( ii , 2 );
//}
//setcolor(RED);
//ir(15,20);
//printstr("VOLT.",2);
//a=-15;
//for( i=0 ; i<=10 ; i++ ) {
//    j=450-40*i;
//    linea( 40 ,j, 50, j );
//    if (a<0) ir(5 , j); else ir(15,j);
//    prntnum( a , 2 );
//    a=a+3;
//}
//ir(565,520);
//printstr("SEG" , 2 );
ratonpres();
} // pantallagraficar.

```

```

void main (void) {
    int a,b,c,d,i,j;
    int aux,medicion,pausa,grabando,reproduciendo;
    int error=0;
    long tiempo ,tiempo1;
    long medicion1,medicion2;
    long muestra1,muestra2;
    double ii,aa,aa1;
    long k;
    strcpy(NOMBRE1,"VAR1[volt]");
    strcpy(NOMBRE2,"VAR2[volt]");
    FACTOR1=1.0; //factores de conversion VOLT x FACTOR = VARIABLE.
    FACTOR2=1.0;

    outportb( PUERTO , 0x0a0 ); // WR del ADC0804 a alto. Lee 4 bits MSB.
    // Habilita la parte mas significativa del 74244 que esta conectada
    // a los 4 bits mas signif. del ADC0804.

    outportb( PUERTO+2 , 0 );
    // configura el puerto 378H como de salida si es bidireccional.

    for( a=0 ; a<1000 ; a++){
        itoa( a , cad1 , 10 );
        strcpy(paciente[a].nombre , cad1);
        itoa( a , cad1 , 16 );
        strcpy(paciente[a].fecha , cad1);
    }

    datafield[0][0]= 100;
    datafield[0][1]= 240;
    datafield[1][0]= 100;
    datafield[1][1]= 310;

    for( i=2 ; i<= 21 ; i++ ) {
        datafield[i][1]= 76 + 20*(i-2);
        datafield[i][0]= 66;
    }

    for( i=0 ; i< NUMMUESMAX ; i++ ) {
        muestras[i]=0;
    }

    for( i=0 ; i<=560 ; i++ ) {
        tabla1[i]=350;
    }

    for( i=0 ; i<=560 ; i++ ) {
        tabla2[i]=350;
    }

    inimodgraf();
    maxx=getmaxx();
}

```



```

maxy=getmaxy();

// portada();

cleardevice();
chpalette2();
setfillstyle(SOLID_FILL,0);

ratonini();
ratonpres();

bot[0][0]=220;bot[0][1]=130 ;bot[0][2]=420;bot[0][3]=170 ;
bot[1][0]=220;bot[1][1]=220 ;bot[1][2]=420;bot[1][3]=260 ;
bot[2][0]=220;bot[2][1]=220 ;bot[2][2]=420;bot[2][3]=260 ;
bot[3][0]=220;bot[3][1]=310 ;bot[3][2]=420;bot[3][3]=350 ;

bot[4][0]=210;bot[4][1]=220 ;bot[4][2]=430;bot[4][3]=280 ;

bot[5][0]=0;bot[5][1]=41 ;bot[5][2]=280;bot[5][3]=80 ;
bot[6][0]=0;bot[6][1]=81 ;bot[6][2]=280;bot[6][3]=120;
bot[7][0]=0;bot[7][1]=121 ;bot[7][2]=280;bot[7][3]=160;
bot[8][0]=0;bot[8][1]=161 ;bot[8][2]=280;bot[8][3]=200;

bot[9][0]=451;bot[9][1]=370;bot[9][2]=551;bot[9][3]=400;
bot[10][0]=250;bot[10][1]=370;bot[10][2]=350;bot[10][3]=410;

bot[11][0]=000;bot[11][1]=441;bot[11][2]=110;bot[11][3]=479;
bot[12][0]=111;bot[12][1]=441;bot[12][2]=220;bot[12][3]=479;
bot[13][0]=221;bot[13][1]=441;bot[13][2]=330;bot[13][3]=479;
bot[14][0]=331;bot[14][1]=441;bot[14][2]=440;bot[14][3]=479;
bot[15][0]=441;bot[15][1]=441;bot[15][2]=540;bot[15][3]=479;
bot[16][0]=541;bot[16][1]=441;bot[16][2]=639;bot[16][3]=479;

bot[17][0]=200;bot[17][1]=370;bot[17][2]=270;bot[17][3]=410;
bot[18][0]=330;bot[18][1]=370;bot[18][2]=400;bot[18][3]=410;

bot[19][0]=201;bot[19][1]=241;bot[19][2]=400;bot[19][3]=280;
colorboton[0 ]=8 ;
colorboton[1 ]=9 ;
colorboton[2 ]=11;
colorboton[3 ]=6 ;
colorboton[4 ]=4 ;
colorboton[5 ]=5 ;
colorboton[6 ]=6 ;
colorboton[7 ]=7 ;
colorboton[8 ]=8 ;
colorboton[9 ]=9 ;
colorboton[10]=10;
colorboton[11]=11;
colorboton[12]=12;
colorboton[13]=13;
colorboton[14]=14;
colorboton[15]=15;

```

```

colorboton[16]=0;
colorboton[17]=10;
colorboton[18]=0;

do{

    ratonesc();
    rectang(1,1,639,639,RED,BLACK);
    rectang(180,130,460,500,LIGHTRED,BLACK);

    boton1 (0,msg[0]);
    boton1 (1,msg[1]);
    boton1 (3,msg[3]);

    ratonpres();

    do {

        ventana=0;
        opc=getkey1();

    } while (opc!=700 && opc!=701 && opc!=703 &&
            opc!='s' && opc!='S' && opc!='G' && opc!='g' &&
            opc!='F' && opc!='f' );

    switch (opc ) {

    case 701:
    case 'G':
    case 'g':
    {
        grabando=0;
        reproduciendo=0;
        nummuestra=0;
        totalmuestras=-1;
        punterotab=1;
        tiempo1= (long)(FACTOR)*periodo - DIFERENCIA;
        pantallagraficar();
        status("MUESTREANDO...");
        ratonesc();
        for (i=11;i<=16;i++)
        {
            boton1 (i,msg[i]);
        }
        ratonpres();
        pausa=0;

        do {

            if(pausa==0){

                medicion=muest1();
                if (reproduciendo==1){

```

```

    medicion=muestrasgrab[nummuestra];
}
muestra1=medicion;
// 0 ----> 535
// 255 ----> 135
medicion1=535-(long)(medicion)*80/51;

//medicion=muest2();
//muestra2=medicion;
//medicion2=50+(long)(medicion)*400/255;

//tiempo=0;
//while( tiempo<tiempo1 ) { tiempo++; }

if( punterotab<=540 ) punterotab1=punterotab+20;
else punterotab1=punterotab+20-560;
setcolor(LIGHTBLUE);
if (punterotab1>1) {
    linea(punterotab1+49,tabla1[punterotab1-1],punterotab1+50,tabla1[punterotab1]);
    // linea(punterotab1+49,tabla2[punterotab1-1],punterotab1+50,tabla2[punterotab1]);
}
tabla1[punterotab]=(int)medicion1;
// tabla2[punterotab]=(int)medicion2;
if (punterotab>1) {
    setcolor(BLACK);
    linea(punterotab+49,tabla1[punterotab-1],punterotab+50,tabla1[punterotab]);
    //setcolor(RED);
    //linea(punterotab+49,tabla2[punterotab-1],punterotab+50,tabla2[punterotab]);
}
if (punterotab==560) punterotab=1;
else punterotab++;
if (grabando==1){
    muestrasgrab[nummuestra]=(int)muestra1;
}
nummuestra++;
if (grabando==1 && nummuestra==16384){
    grabando=0;
    arraytofile();
    pausa=0;
    punterotab=1;
    pantallagraficar();
    status("MUESTREANDO...");
    nummuestra=0;
    funcion++; funcion=funcion & 7;
}
if (nummuestra==16385) nummuestra=0;
if (reproduciendo==1 && nummuestra==totalmuestras){
    pausa=1;
    status("FIN REPRODUCCION");
}

/*
if (nummuestra<50){

```

```

a=nummuestra;
strcpy(reg[a].muestra1 , mtoa(muestra1) );
strcpy(reg[a].muestra2 , mtoa(muestra2) );
for( i=0 ; i<=15 ; i++ ) {
    if (reg[a].muestra1[i]=='\0') reg[a].muestra1[i]=' ';
    if (reg[a].muestra2[i]=='\0') reg[a].muestra2[i]=' ';
}
reg[a].muestra2[14]=13;
reg[a].muestra2[15]=10;
nummuestra++;
}
*/
}
ventana=3;
aux=getkey1();
if ( aux=='G' || aux==700 ) {
    if (grabando==0 && reproduciendo==0) {
        grabando=1;
        nummuestra=0;
        status("GRABANDO...");
    }
}
if ( aux=='P' || aux==701 ) {
    if (grabando==1) {
        grabando=0;
        arraytofile();
        pausa=0;
        punterotab=1;
        pantallagraficar();
        status("MUESTREANDO...");
        nummuestra=0;
        funcion++; funcion=funcion & 7;
    }
    if (reproduciendo==1) {
        reproduciendo=0;
        status("PARAR REPRODUCCION");
        delay(1000);
        pausa=0;
        punterotab=1;
        pantallagraficar();
        status("MUESTREANDO...");
        nummuestra=0;
    }
}
if ( aux=='A' || aux==704 ) {
    if (grabando==0) {
        if ( cargar() ){
            pantallagraficar();
            status("REPRODUCIENDO...");
            reproduciendo=1;
            nummuestra=0;
            punterotab=1;
            pausa=0;
        }
    }
}

```

```

    }
    else {
        pantallagraficar();
        status("MUESTREANDO...");
        reproduciendo=0;
        pausa=0;
        punterotab=1;
        nummuestra=0;
    }
}
}
if ( aux=='R' || aux==702 ) {
    if (totalmuestras!=1 && grabando==0) {
        status("INICIA REPRODUCCION.");
        delay(1000);
        pantallagraficar();
        status("REPRODUCIENDO...");
        punterotab=1;
        pausa=0;nummuestra=0;reproduciendo=1;
    }
}
if ( aux=='C' || aux==703 ) {
    if (pausa==0) pausa=1;
    else pausa=0;
    if (reproduciendo==1 && nummuestra==totalmuestras && pausa==0){
        status("INICIA REPRODUCCION.");
        delay(1000);
        pantallagraficar();
        status("REPRODUCIENDO...");
        punterotab=1;
        pausa=0;nummuestra=0;reproduciendo=1;
    }
}
if (aux==27 || aux=='S' || aux==705) {
    aux=27;
    // for (a=0 ; a<=49 ; a++) {
    //     fwrite(&reg[a], sizeof(reg[a]), 1, FCOD);
    // }
    // fclose(FCOD);
}
}while ( (aux!=27) );
}break;

case 702:
case 'R':
case 'r':
{
    nummuestra=0;
    error=0;
    FCOD=fopen("mues512.txt","wb");
    if (FCOD==NULL){ ventanamsg("ERROR AL CREAR ARCHIVO"); error=1; }
    else {
        clearerr(FCOD);

```

```

        rewind(FCOD);
        a=fseek(FCOD,0 ,SEEK_SET);
    }
    if (error==0){
        for (a=0 ; a<=49 ; a++) {
            strcpy(reg[a].muestra1 ,"      ");
            strcpy(reg[a].muestra2 ,"      ");
        }
        punterotab=1;
        tiempo1=(long)(FACTOR)*periodo - DIFERENCIA ;
        rectang(1,1,639,545,LIGHTBLUE,BLACK);
        rectang(1,546,639,639,GREEN,BLACK);
        ir(10,570);
        printstr("PARA DETENER PRESIONE LA TECLA [D]",2);
        ir(10,615);
        printstr("PARA CERRAR PRESIONE LA TECLA [ESC]",2);
        ir(460,30);
        printstr("PERIODO DE MUESTRO=",1);
        ir(470,50);
        setcolor(RED);
        printnum(periodo,1);setcolor(BLACK);
        printstr(" MILISEGUNDOS",1);
        ir(460,120);
        printstr("FACTORES CONVERSION:",1);
        ir(470,150);
        printstr("VOLTIOS x ",1);
        setcolor(RED);
        printdouble(FACTOR1,1);setcolor(BLACK);
        printstr(" =",1);
        ir(470,168);
        printstr(NOMBRE1,1);
        ir(470,200);
        printstr("VOLTIOS x ",1);
        setcolor(RED);
        printdouble(FACTOR2,1);setcolor(BLACK);
        printstr(" =",1);
        ir(470,218);
        printstr(NOMBRE2,1);

        setcolor(RED);
        ir(15,20);
        printstr(NOMBRE2,2);
        ratonesc();
        aa=MINIMO;
        for( i=0 ; i<=10 ; i++ ) {
            j=450-40*i;
            linea( 50 ,j, 450, j );
            ir(3,j);
            aa1=aa*FACTOR2;
            if (aa1<100.0 && aa1>-100.0) printdouble2( aa1 , 1 );
            else printlong( (long)aa1 ,1);
            aa=aa+(MAXIMO-MINIMO)/10;
        }
    }

```

```

aa=MAXIMO;
for( i=0 ; i<=10 ; i++ ) {
    j=450-40*i;
    setcolor(RED);
    linea( j ,50, j ,450 );
    ir(j+4 , 500);
    aa1=aa*FACTOR1;
    setcolor(BLACK);
    if (aa1<100.0 && aa1>-100.0) printdouble1( aa1 , 1 );
    else printlong1( (long)aa1 ,1);
    aa=aa-(MAXIMO-MINIMO)/10,;
}
ratonpres();
ir(455,455);
printstr(NOMBRE1 , 2 );
linea(50,50,50,453);
linea(50,454,450,454);
linea(50,453,450,453);

pausa=0;
do {

    if(pausa==0){
        // 0 --> 15v
        // 255 --> -15v  Y-15=-30/255*X  Y=15-2/17*X  Y=(255-2*X)/17
        //MINIMO ---> 50  medicion1
        //MAXIMO ---> 450  Z=(450-50)/(MAX-MIN)*(Y-MAX)+450
        //MINIMO ---> 450  medicion2
        //MAXIMO ---> 50  Z=(450-50)/(MIN-MAX)*(Y-MAX)+50

        medicion=muest1();
        muestra1=medicion; //BLACK
        medicion1=400*(255-(long)medicion*2-MAXIMO*17)/((MAXIMO-MINIMO)*17)+450;
        medicion=muest2();
        muestra2=medicion; //RED
        medicion2=400*(255-(long)medicion*2-MAXIMO*17)/((MINIMO-MAXIMO)*17)+50;

        tiempo=0;
        while( tiempo<tiempo1 ) { tiempo++; }
//    if( punterotab<=540 ) punterotab1=punterotab+20;
//    else punterotab1=punterotab+20-560;
//    setcolor(LIGHTBLUE);
//    if (punterotab1>1) {
//        linea(punterotab1+49,tabla1[punterotab1-1],punterotab1+50,tabla1[punterotab1]);
//        linea(punterotab1+49,tabla2[punterotab1-1],punterotab1+50,tabla2[punterotab1]);
//    }
        tabla1[punterotab]=(int)medicion1;
        tabla2[punterotab]=(int)medicion2;
        if (punterotab>1) {
            setcolor(BLACK);

            linea(medicion1,medicion2,medicion1+2,medicion2);
            linea(medicion1,medicion2+1,medicion1+2,medicion2+1);

```

```

        linea(medicion1,medicion2+2,medicion1+2,medicion2+2);

//      linea(punterotab+49,tabla1[punterotab-1],punterotab+50,tabla1[punterotab]);
//      linea(punterotab+49,tabla2[punterotab-1],punterotab+50,tabla2[punterotab]);
    }
    if (punterotab==560) punterotab=1;
    else punterotab++;
    if (nummuestra<50){
        a=nummuestra;
        strcpy(reg[a].muestra1 , mtoa1(muestra1) );
        strcpy(reg[a].muestra2 , mtoa2(muestra2) );
        for( i=0 ; i<=15 ; i++) {
            if (reg[a].muestra1[i]=='\0') reg[a].muestra1[i]=' ';
            if (reg[a].muestra2[i]=='\0') reg[a].muestra2[i]=' ';
        }
        reg[a].muestra2[14]=13;
        reg[a].muestra2[15]=10;
        nummuestra++;
    }
}
aux=revisatec();
if ( aux==68 || aux==100 ) {
    if (pausa==0) pausa=1;
    else pausa=0;
}
if (aux==27) {
    for (a=0 ; a<=49 ; a++) {
        fwrite(&reg[a], sizeof(reg[a]), 1, FCOD);
    }
    fclose(FCOD);
}
}while ( (aux!=27) );
} // if error==0.
}break;

case 700:
case 'F':
case 'f':
{
do{
    ratonesc();
    rectang(1,1,639,639,BLUE,BLACK);
    ir(100,580);
    printstr1("PRESIONE [ESC] PARA RETORNAR",2);
    //for (i=4;i<=8;i++)
    //{
        boton1 (4,msg[4]);
    //}
    ratonpres();
    do {
        ventana=1;
        opc=getkey1();
    }
}
}

```



```

//while (opc!=700 && opc!=701 && opc!=702 && opc!=703 &&opc!=704 &&
//      opc!=49 && opc!=50 && opc!=51 && opc!=52 &&opc!=53 &&
//      opc!=27);
while ( opc!=700 && opc!=49 && opc!=27 );

if (opc==700 || opc==49)
{
    periodo=ventanams2(70,400,"INGRESE EL PERIODO MUESTREO EN MILISEGUNDOS (2-
40):");
};
if (opc==701 || opc==50)
{
    strcpy(NOMBRE1,ventanams3(70,300,"INGRESE NOMBRE 1ra VARIABLE EJEMPLO
W(r/s):"));
};
if (opc==702 || opc==51)
{
    strcpy(NOMBRE2,ventanams3(70,300,"INGRESE NOMBRE 2da VARIABLE EJEMPLO
P1(pa):"));
};
if (opc==703 || opc==52)
{
    FACTOR1=ventanams4(70,300,"INGRESE FACTOR CONVERSION 1ra VARIABLE
EJEMPLO 56.78:");
};
if (opc==704 || opc==53)
{
    FACTOR2=ventanams4(70,300,"INGRESE FACTOR CONVERSION 2da VARIABLE
EJEMPLO 6.85:");
};
} while (opc != 27);
} break;

case 703:
case 's':
case 'S':
{
    cleardevice();
    closegraph();
    ratonini();
    return ;
}

} //switch
ratonini();
ratonpres();

} while(1); //do

} //main

```

DIAGRAMA DE BLOQUES

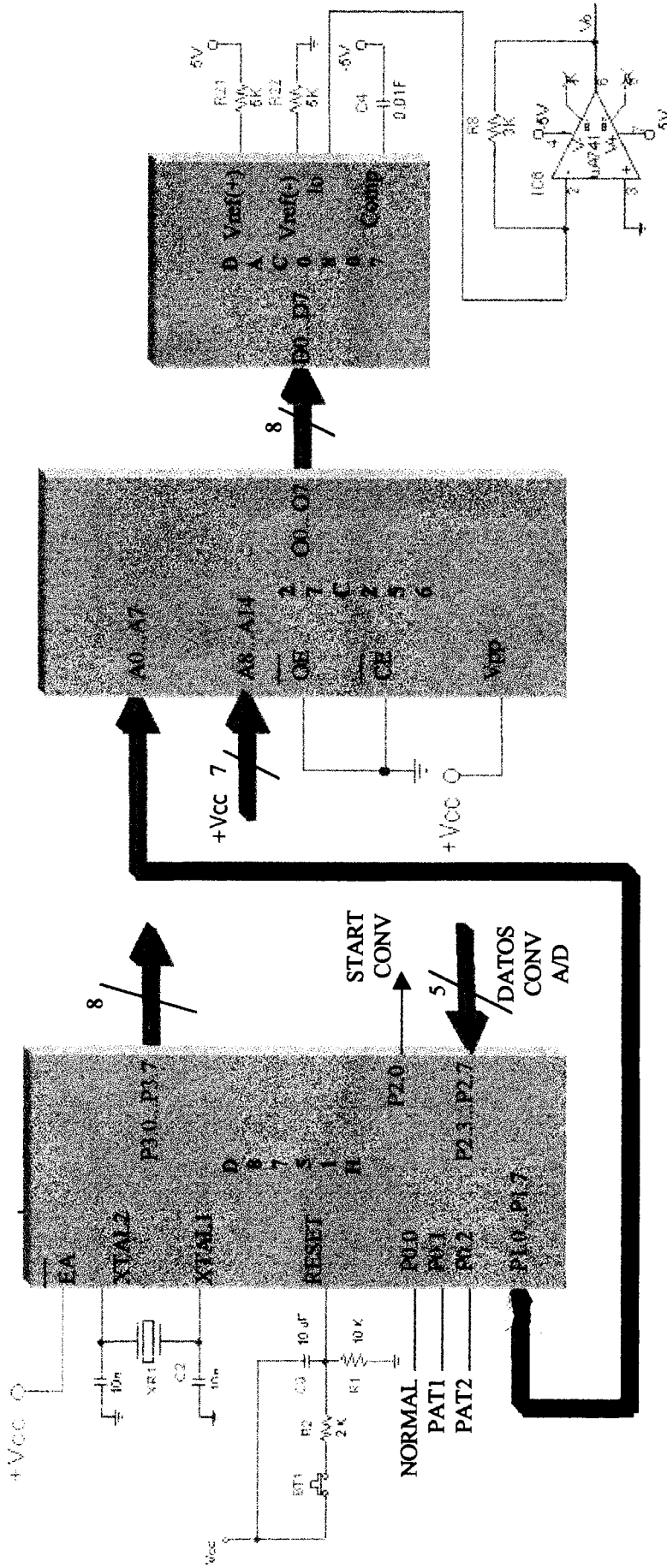
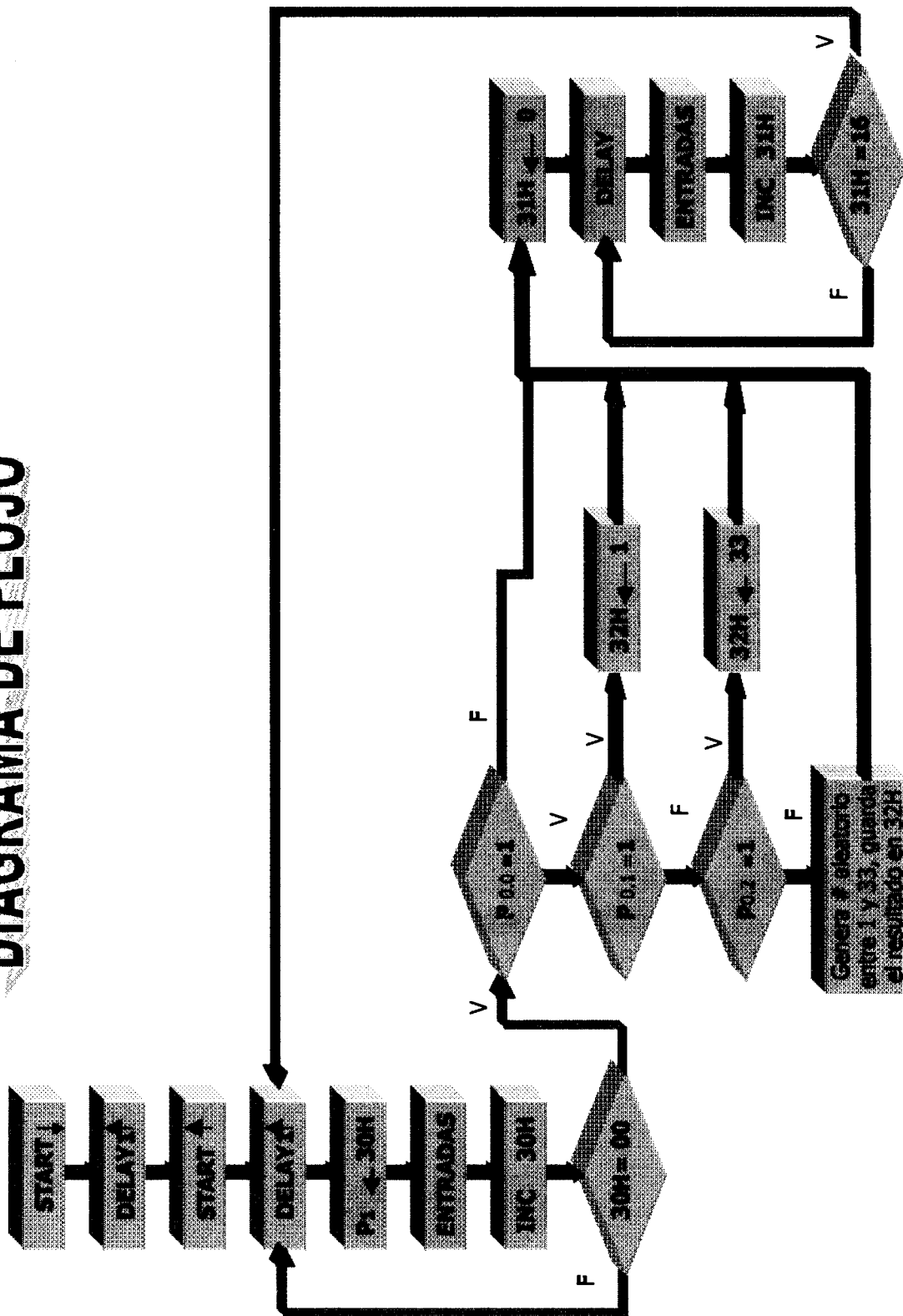
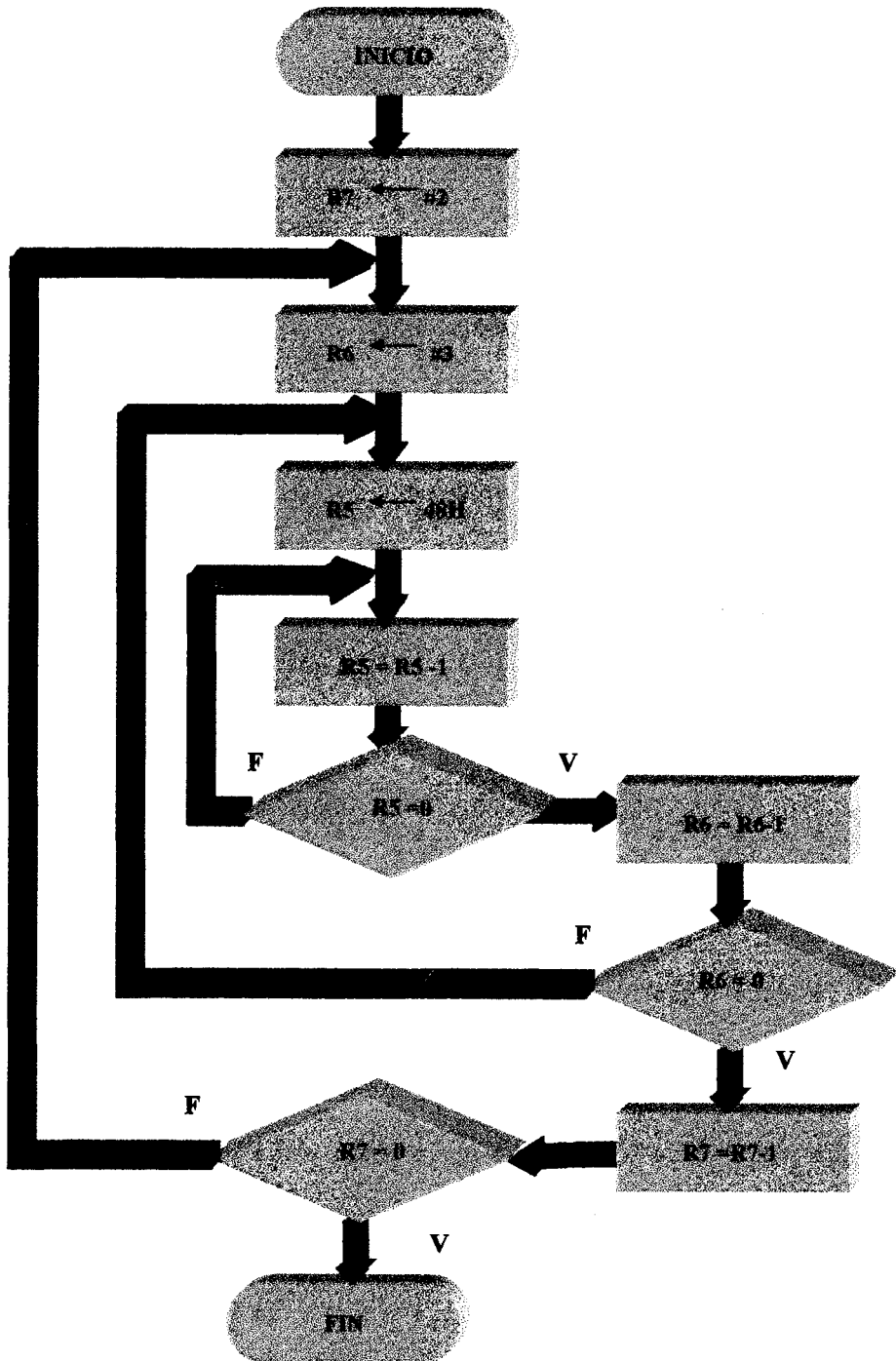


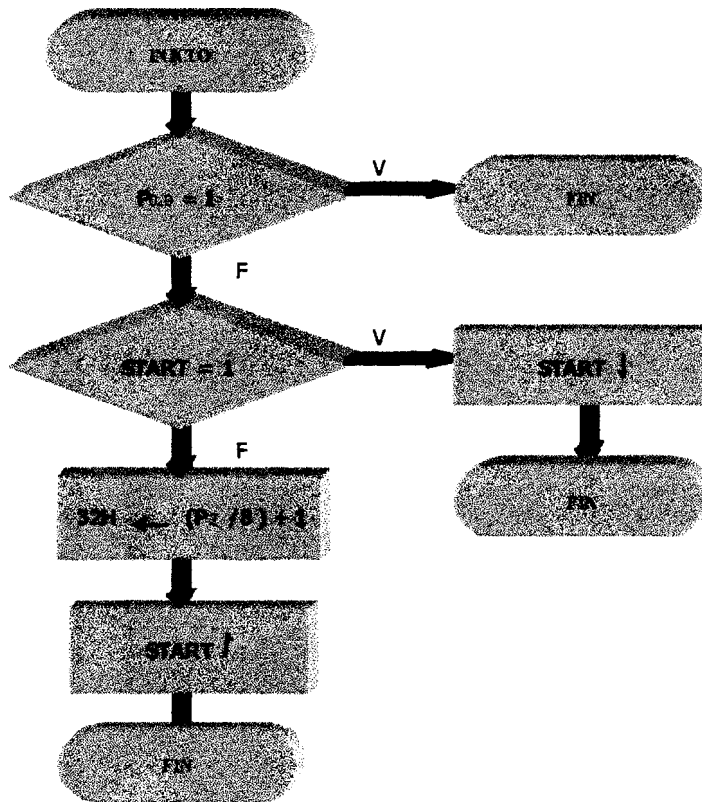
DIAGRAMA DE FLUJO



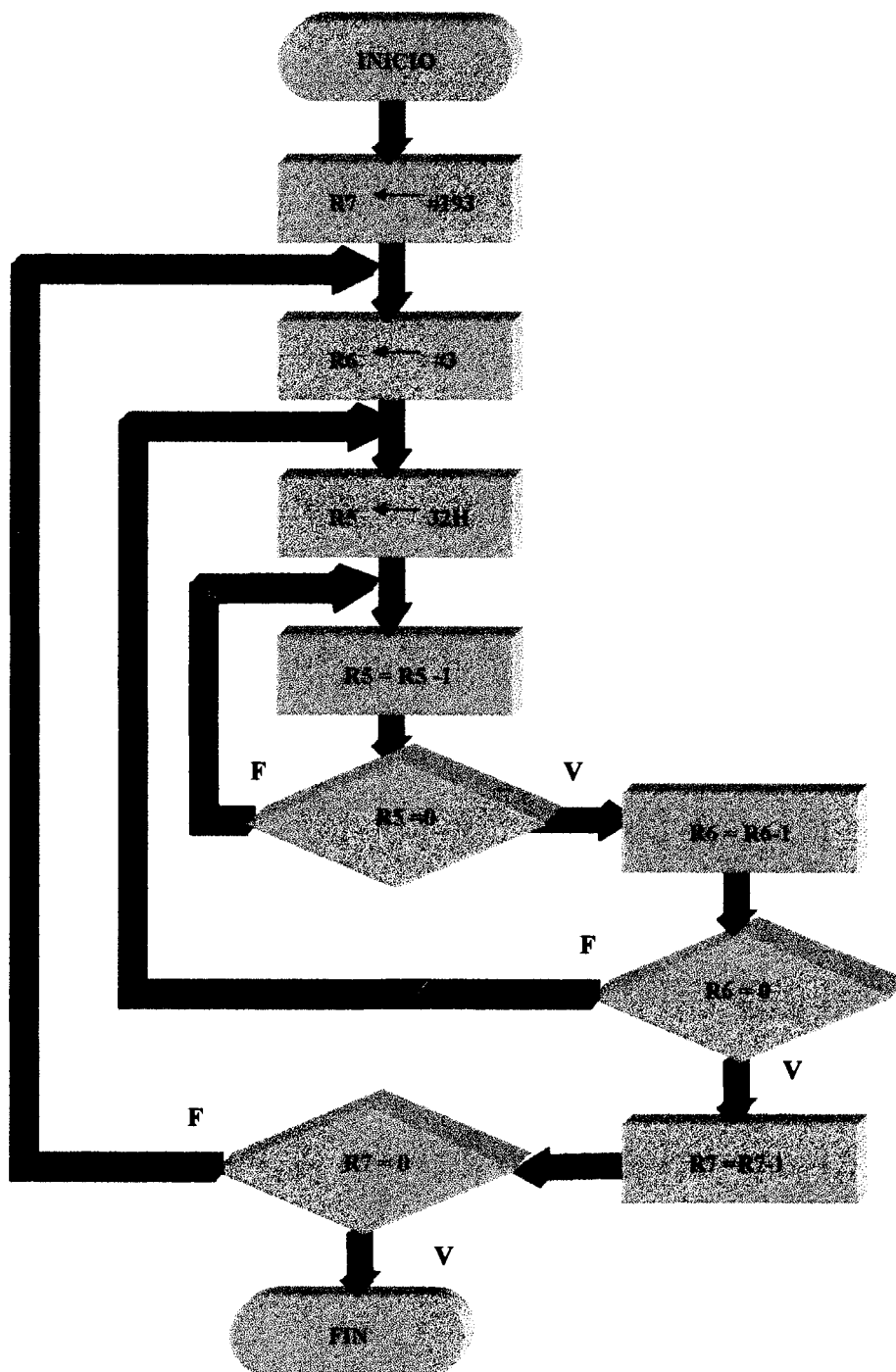
PROCEDIMIENTO DELAY1

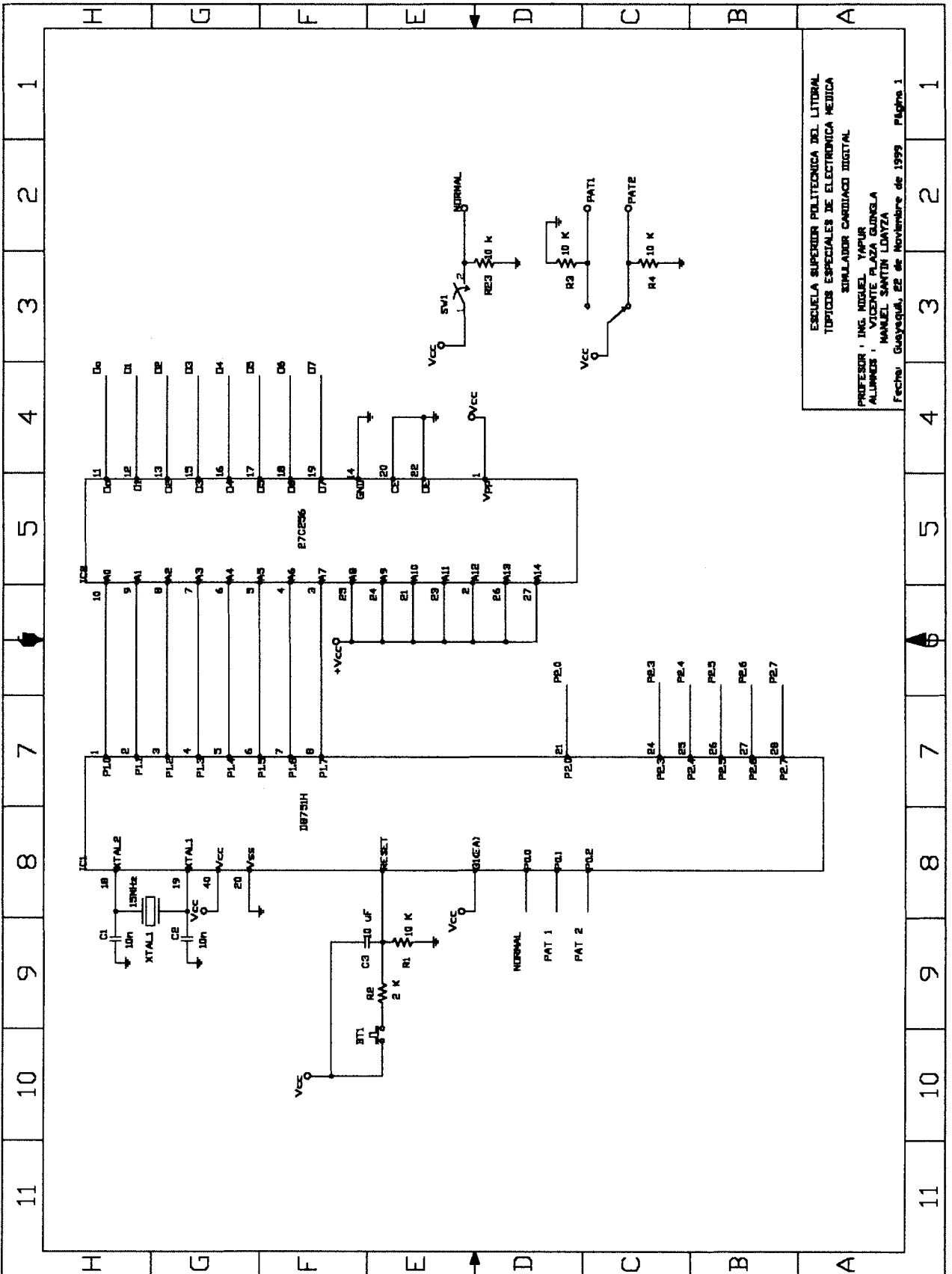


PROCEDIMIENTO ENTRADAS

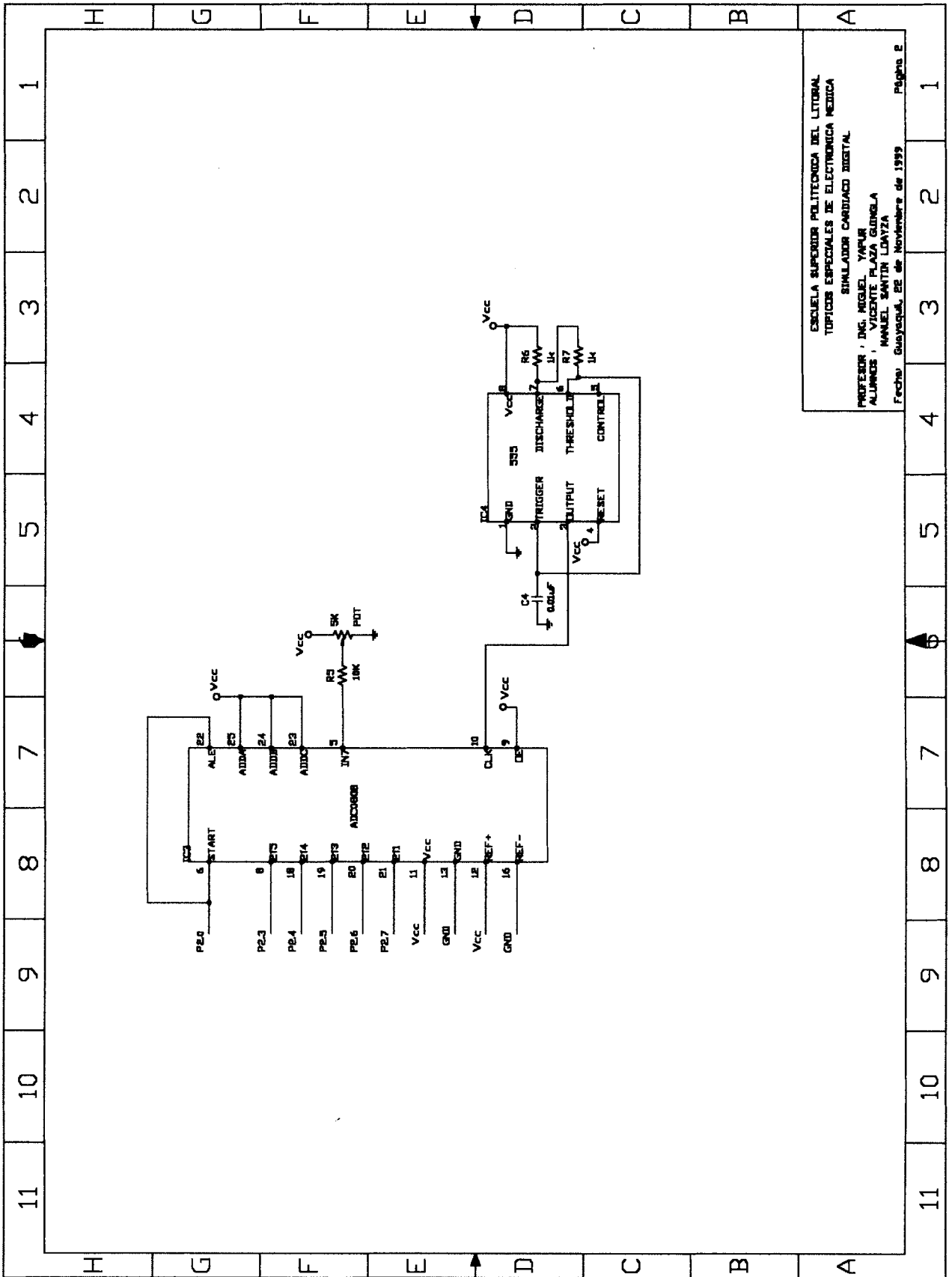


PROCEDIMIENTO DELAY

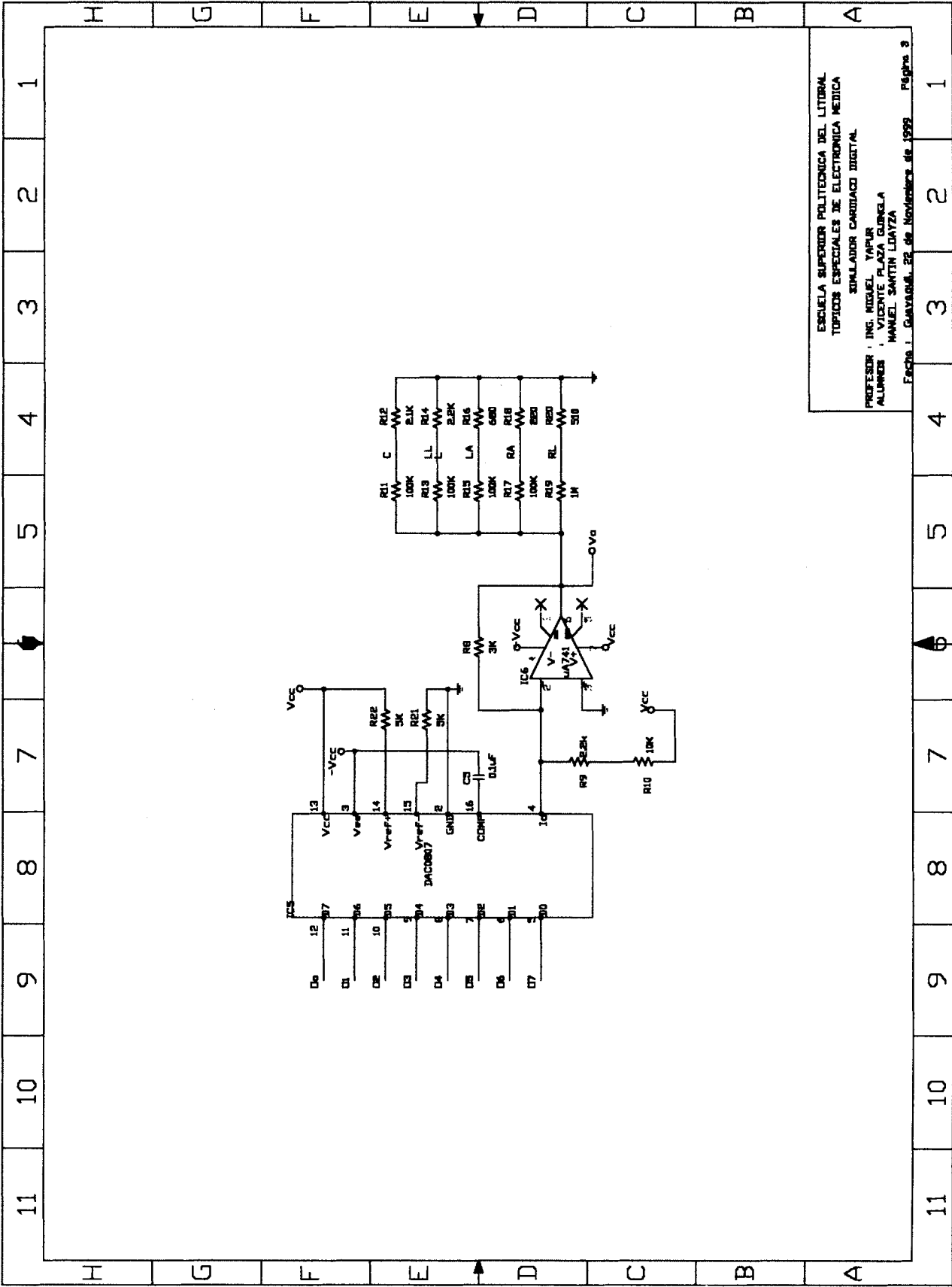




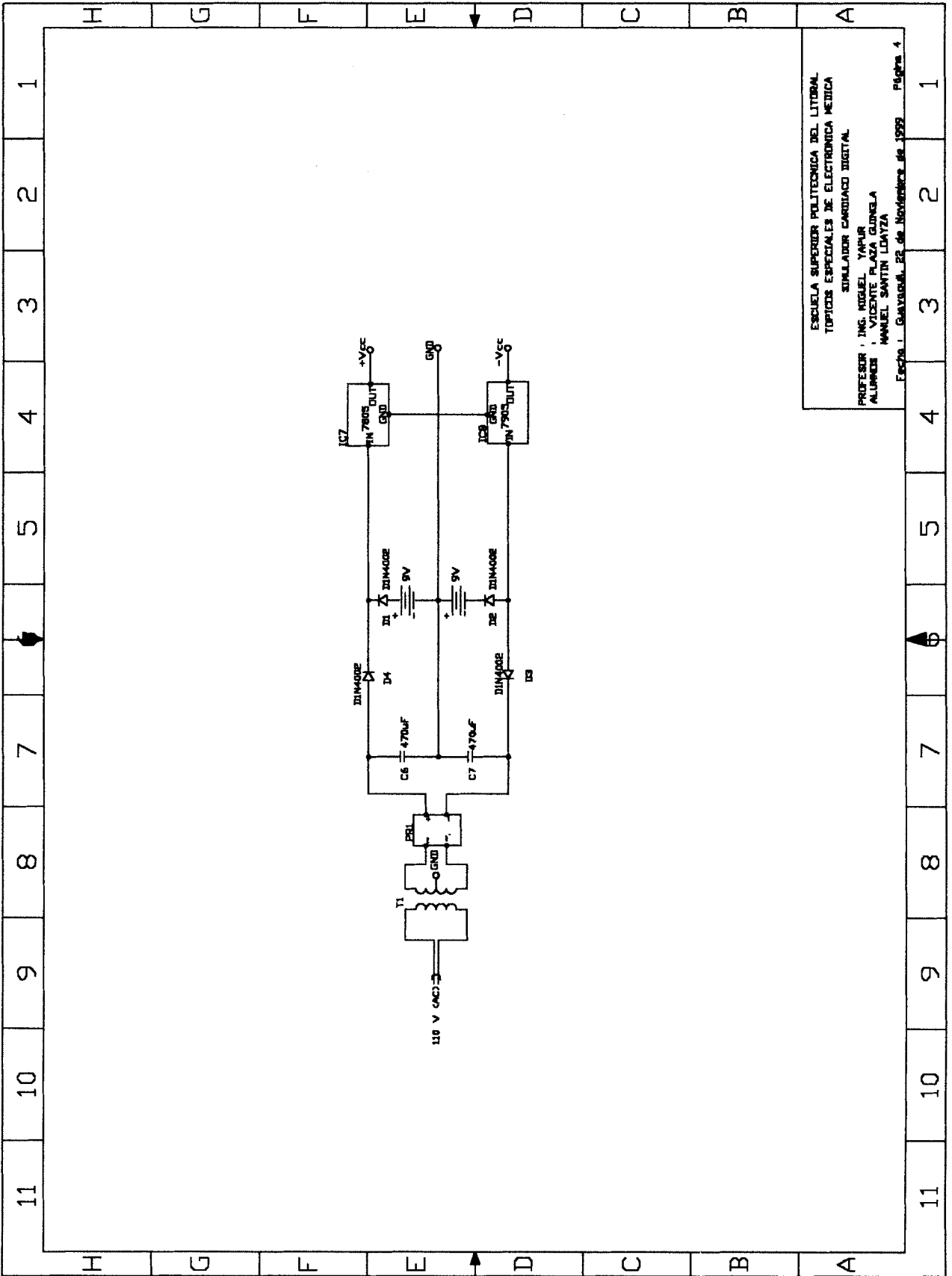
ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL
 TUBOS ESPECIALES DE ELECTRÓNICA MÉRICA
 SIMULADOR DIGITAL
 PROFESOR : ING. MIGUEL YAPIR
 ALUMNOS : VICENTE PLAZA GUINOLA
 MANUEL SANTIN LOAYZA
 Fecha: Guayaquil, 22 de Noviembre de 1999 Página 1



ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL
 TÓPICOS ESPECIALES DE ELECTRONICA MEDICA
 SIMULADOR CARDIACO DIGITAL
 PROFESOR : ING. MIGUEL YANUB
 ALUMNOS : VICENTE PLAZA GONZALE
 MANUEL SANTIN LOAYZA
 Fecha Guayaquil, EE de Noviembre de 1999 Pagina 2

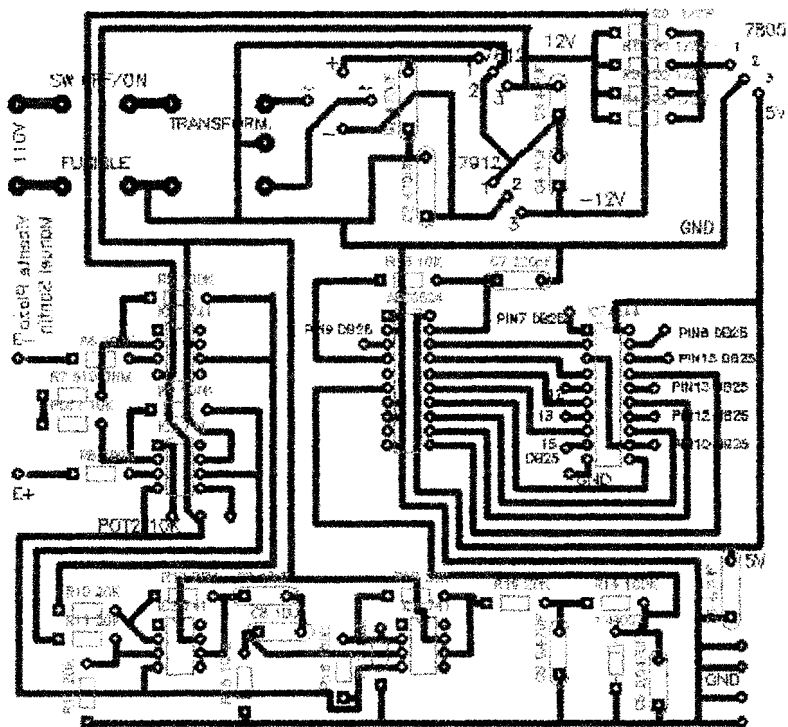


ESCUOLA SUPERIOR POLITECNICA DEL LITORAL
 TOPICOS ESPECIALES DE ELECTRONICA MEDICA
 SIMULADOR CARDIACO DIGITAL
 PROFESOR : ING. MIGUEL YAPUR
 ALUMNOS : VICENTE PLAZA GUMBELA
 MANUEL SAMTIN LOAYZA
 Fecha : GUAYASAL, 22 de Noviembre de 1999 Pagina 3

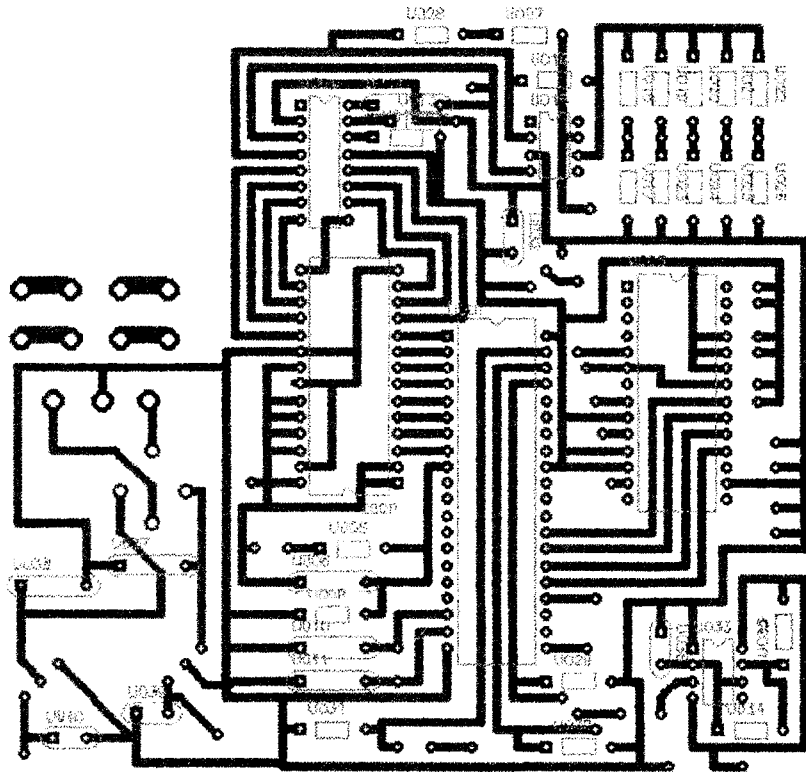


ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL
TOPICOS ESPECIALES DE ELECTRONICA MEDICA
SIMULADOR CARDIACO DIGITAL
PROFESOR : ING. ROQUEL YAPUR
ALUMNOS : VICENTE PLAZA GUINELA
MANUEL SANTIN LIMAYZA
Fecha : Guayaquil, 22 de Noviembre de 1999 Página 4

CIRCUITO IMPRESO DEL C.S.C.



CIRCUITO IMPRESO DEL SIMULADOR



CODIGO FUENTE PARA PROGRAMAR EL MICROCONTROLADOR

```

; Version posterior a ecm2.asm que proporciona ancho de pulso constante
; a cualquier frecuencia.
; Generador de seales para electrocardiograma.
; p1 y p3 conectados a la EPROM 27C256.
; p1 es el menos significativo. p3 siempre vale #0ffh.
; p0.0 es el switch NORMAL/PATOLOGIA.
; p0.1-p0.2 es TAQUICARDIA/BRAQUICARDIA/ARRITMIA.
; p2 esta conectado con el ADC0808 que controlara la frecuencia en NORMAL.
; p2.0 conectado al pin START del ADC0808.
; el pulso ocupa 256 bytes en la eprom. fmin=40ppm , fmax=120ppm.

```

e0

```

    mov p3, #0ffh
    mov 30h, #0h
    mov 32h, #10
    mov 33h, #37
    clr p2.0
    lcall delay1
    setb p2.0

```

e13

```

    lcall delay1
    mov p1, 30h
    lcall entradas
    inc 30h
    mov a, 30h
    cjne a, #00h, e13

```

```

    jnb p0.0, e14

```

```

    jnb p0.1, e7
    mov 32h, #1
    ljmp e14

```

e7

```

    jnb p0.2, e8
    mov 32h, #33
    ljmp e14

```

e8

```

    mov a, 33h
    mov b, #37
    mul ab

```

```
    add a,#59
    mov 33h,a
    anl a,#0f8h
    rr a
    rr a
    rr a
    add a,#1
    mov 32h,a
e14
    mov 31h,#00h
e15
    lcall delay
    lcall entradas
    inc 31h
    mov a,31h
    cjne a,#16,e15
    ljmp e13
```

entradas

```
    jb p0.0, e5

    jb p2.0, e4

    mov a,p2
    anl a,#0f8h
    rr a
    rr a
    rr a
    add a,#1
    mov 32h,a
    setb p2.0
    sjmp e5
e4
    clr p2.0
e5
    ret
```

delay1

```
    mov r7,#2
e211
    mov r6,#3
e201
```

```
    mov r5,#48h
e191
    djnz r5,e191
    djnz r6,e201
    djnz r7,e211
    ret
```

```
delay
    mov r7,#193
e21
    mov r6,#3
e20
    mov r5,32h
e19
    djnz r5,e19
    djnz r6,e20
    djnz r7,e21
    ret
```


CODIGO HEXADECIMAL DEL PROGRAMA

:1000000075B0FF75300075320A753325C2A0120035
:1000100076D2A012007685309012005E0530E53071
:10002000B400F030802530810675320102004B307B
:10003000820675322102004BE53375F025A4243B7E
:10004000F53354F80303032401F53275310012002F
:100050008312005E0531E531B410F30200132080F5
:100060001420A00FE5A054F80303032401F532D2B5
:10007000A08002C2A0227F027E037D48DDFEDEFA60
:10008000DFF6227FC17E03AD32DDFEDEFADFF6222F
:00000001FF

DATOS DIGITALES ALMACENADOS EN LA EPROM

7F 7F 80 81 82 83 84 85 86 86 87 88 89 8A 8B 8B 8C 8D 8E 8E 8F 8F 8F 90 91 91 91
 91 91 91 90 90 90 8F 8F 8F 8E 8E 8E 8D 8D 8D 8C 8C 8C 8A 8A 88 88 86 86 84 84
 82 81 81 80 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7E 7D 7B 79 78 79
 8E A0 AC B5 C0 CB D5 DB E1 E6 EB F3 F4 F0 E8 E3 DF DC D4 C1 AE 98 8F 89 80
 6D 69 60 63 68 6A 6B 6C 70 71 77 78 79 7A 7A 7B 7C 7D 7D 7E 7F 7F 7F 7F 7F
 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 80 81 82 83 84 85 86 87 88 8A
 8B 8C 8E 8F 91 92 95 96 98 99 9B 9B 9C 9C 9C 9C 9D 9D 9D 9D 9E 9E 9E 9E 9E
 9F 9F 9F 9F 9F A0 A0 A0 A0 A0 A0 9F 9F 9F 9E 9E 9E 9E 9D 9D 9D 9C 9C 9B 9B
 9A 9A 99 98 97 96 96 95 95 95 94 94 93 91 90 8F 8E 8E 8D 8D 8B 8A 88 87 86 86 84
 83 81 80 80 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F 7F

Nota: Este grupo de datos se repiten 32 veces, y se almacenan en la EPROM

LISTA DE COMPONENTES

COMPONENTES	CANTIDAD	CANTIDAD POR COMPONENTE

LISTA DE COMPONENTES

COMPONENTES	A DE C	DETALLE Y DESCRIPCION

LISTA DE PRECIOS

COMPONENTES DE LOS PRECIOS

LISTA DE PRECIOS

COMPONENTES	UNID.	P. B. U.	P. P. U.

BIBLIOGRAFIA

1. José Adolfo Gonzáles Vázquez, Introducción a los microcontroladores Hardware, Software, Aplicaciones (Madrid, McGraw-Hill, 1992).
2. Robert F. Coughlin / Frederick F. Driscoll, Amplificadores operacionales y circuitos integrados lineales, (Cuarta Edición, Prentice Hall, 1989) pp 88-106.
3. Ronald J. Tocci, Sistemas Digitales, principios y aplicaciones (Sexta Edición, Prentice Hall) , pp 561-607.
4. Robert Boylestad / Louis Nashelsky, Electrónica teoría de circuitos (Quinta Edición, México, Prentice Hall, 1989) pp. 500-674.
5. Barry N. Feinberg, Applied Clinical Engineering, (New Jersey, Prentice Hall) pp. 50-103.