Universidad Tecnológica de la Mixteca

Electrocardiógrafo basado en la tarjeta TMS320C3x Starter Kit

Tesis que para obtener el título de Ingeniero en Electrónica, presenta:

Floriberto Ortiz Rodríguez

Asesor:

M.C.: José Antonio Moreno Espinosa

Acatlima, Huajuapan de León Oaxaca, Marzo de 2002.


UNIVERSIDAD TECNOLOGICA DE LA MIXTECA
Labor et
Sapientia
Libertas
UTM
Chun quiaa Tiaha sihi Nillao
OAXACA


UNIVERSIDAD TECNOLOGICA DE LA MIXTECA
Labor et
Sapientia
Libertas
UTM
Chun quiaa Tiaha sihi Nillao
OAXACA

Dedicatoria.

A mis padres, hermanos y abuelos.

Agradecimientos.

Agradezco profundamente a las siguientes personas por haberme apoyado plenamente en la culminación de la tesis sin esperar nada a cambio, salvo el verme convertido en un hombre de bien, pero sobre todo, en una mejor persona.

A mi asesor, M.C. José Antonio Moreno Espinosa, no solo por la asesoría que me brindó para la culminación de la tesis, si no por sus consejos para afrontar la vida y para ser cada día mejor.

Al Dr. Juan Carlos Ramos A., por la revisión, sugerencias y por el interés mostrado para mejorar la tesis, pero por encima de todo, por su grata amistad.

A mis profesores que compartieron conmigo sus conocimientos y experiencias y además, perdonaron mis faltas: Enrique Guzmán Ramírez, Hugo Leyva, Ramón Maldonado Basilio, Hugo Suárez Onofre, Esteban Guerrero, Heriberto Hernández, José Javier Báez Rojas, Arnulfo Pérez, Maribel Tello, Victor Manuel.

A mis familiares: tíos, primos, muy en especial a L. M. R..

A mis amigos que estuvieron conmigo en los momentos buenos y malos, cuyas virtudes siempre admiré: a José Manuel Ávila Vasquez, por su férrea disciplina en el estudio y el trabajo; a Lancelot García Leyva, por su actitud ante la vida de creer que cualquier obstáculo es casi siempre superable; a Hector Martínez Martínez, por su terquedad y consistencia, y que a pesar de todo, si se puede; a Jorge Luis Vidals Cruz, por su manera de disfrutar la vida. Y a todos mis compañeros de clase.

CONTENIDO.

III. Descripción del sistema desarrollado.

IV. Perspectivas y conclusiones.

Apéndices.

FIGURAS.

Prólogo

La tesis esta dividida en 4 capítulos. El primer capítulo trata acerca de la fisiología del corazón, como este órgano marcapasos funciona como una bomba, haciendo circular la sangre por el cuerpo humano y a través de ella es posible medir la corriente extracorporalmente que genera al polarizarse y despolarizarse las células y los tejidos, presenta también que tipo de onda se tiene que visualizar y en que partes del cuerpo humano es posible tomar estas mediciones.

El segundo capítulo trata sobre el concepto básico de las señales, y sus dos grandes divisiones: señales analógicas y digitales, así como de sus respectivas características que presenta cada una de ellos. También menciona como se puede pasar de una señal analógica a una digital y de los medios electrónicos que disponemos para poder hacerlo sin perder mucha información. El procesamiento digital es una parte muy importante pues a través de él, dada una señal, podemos tratarla de tal manera que sea poco afectada por el ruido o que sean poco perceptibles implementando algún tipo de filtrado.

El tercer capítulo resume todo lo que se refiere al desarrollo del sistema, hardware y software respectivamente, los circuitos que se utilizaron para tomar las señales eléctricas presentes en el ser humano, como fueron convertidas en señales digitales y también como fueron procesadas para que la señal final estuviera libre de algún tipo de ruido y fuera lo más fielmente posible a lo que un médico pudiera esperar y lograr una valoración más precisa. Se describen también, los programas realizados en la computadora que permitieron la adquisición de datos, el procesamiento digital por parte del procesador TMS320C31 y su posterior visualización en el monitor de una computadora.

En el cuarto capítulo se presenta las conclusiones del trabajo de tesis acerca del funcionamiento del sistema en general que alcances actuales tiene y cuáles son los que pudiera llegar a tener, añadiendo posteriormente determinados módulos de hardware o software para hacer el dispositivo mas eficiente y funcional.

Introducción.

En una época de constantes cambios, donde la tecnología ha avanzado de manera vertiginosa y ha pasado de mero desarrollo tecnológico, propio de países del primer mundo, a una nueva forma de vida del incipiente siglo XXI, en la que el trabajo cotidiano es cada vez más arduo e intenso, es común que las personas de ambos sexos se sientan agobiadas y estresadas por diversos factores, como la presión en el trabajo, el incremento en el ritmo del mismo o una dieta poco balanceada. Estos han llegado a tal punto que las enfermedades, propias del corazón se han vuelto cada día más frecuentes.

Una de las herramientas más útiles y auxiliar en la detección de enfermedades de este órgano, es sin duda el electrocardiógrafo. Este instrumento nos permite obtener un electrocardiograma de las pulsaciones del corazón con información necesaria para poder comprender, analizar e interpretar estos datos por especialistas en el área, y poder dar un diagnóstico confiable, de determinado padecimiento que permita controlar y en su caso curar a pacientes con estos problemas de salud.

La electrónica juega un papel importante en este sentido, pues es el medio o herramienta que permite desarrollar (construir) este tipo de dispositivos que ayudan a los especialistas a hacer una evaluación más acertada. En particular la bioelectrónica es el área que se encarga del desarrollo de aparatos médicos que permiten de manera más completa obtener toda la información necesaria del cuerpo humano y ofrecer resultados fácilmente interpretados por médicos, haciendo una evaluación y diagnóstico más eficientemente (rápido y seguro).

Objetivo.

Desarrollar un Electrocardiógrafo basado en la tarjeta TMS320C3x Starter Kit de Texas Instruments con componentes comerciales de fácil adquisición, con el objetivo de obtener una señal de electrocardiograma con 3 derivaciones bipolares (pudiéndose expandir hasta con 5 derivaciones más), que serán visualizadas en el monitor de una computadora. Estas derivaciones deben mostrar una respuesta muy similar a la presentada por el equipo médico comercial que se encuentra instalado en los hospitales de la región mixteca.

Justificación.

El equipo médico actual de electrocardiógrafos instalados en los hospitales de la región mixteca son de buena calidad, sin embargo, el precio al adquirir este equipo por parte de los hospitales y médicos, es relativamente caro, pues su precio oscila alrededor de los $ 1,200 y $ 4,500 dólares respectivamente, siendo un gasto que pocas instituciones de salud de esta región son capaces de solventar, e inaccesible al médico para un ejercicio privado de su profesión. Otro inconveniente se presenta cuando alguno de estos aparatos llega a fallar pues el gasto se incrementa al pagar los servicios de un técnico especializado y en la demora de tiempo que esto pueda originar.

CAPÍTULO I.

Fisiología del corazón.

1.1 El corazón.

El corazón del hombre es un músculo hueco localizado en la cavidad torácica entre los dos pulmones por arriba del diafragma, por delante de la columna vertebral, entre la 4ta. y 8va. vertebra dorsal, atrás del esternón y de los cartílagos costales, su forma se asemeja a una pirámide triangular invertida, espacialmente dentro de esta cavidad apunta de arriba hacia abajo, de derecha a izquierda y de atrás hacia adelante [1]. Su función es la de mantener con vida a todos los órganos presentes en el ser humano mediante la irrigación de la sangre a través de ellos por su efecto de bomba. (Fig. 1.1) y (Fig. 1.3)

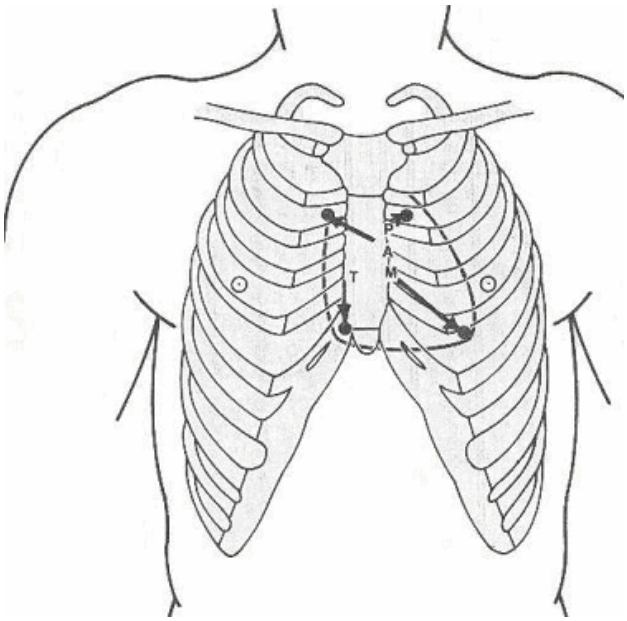


Figura 1.1 Ubicación del corazón dentro de la cavidad torácica

El corazón late debido a la presencia de un tejido especializado que funciona como un marcapaso capaz de iniciar potenciales de acción iterativos. El tejido marcapaso forma el sistema de conducción que normalmente propaga los impulsos por todo el corazón, a causa de que los líquidos corporales son buenos conductores eléctricos [2]. (Fig. 1.2)

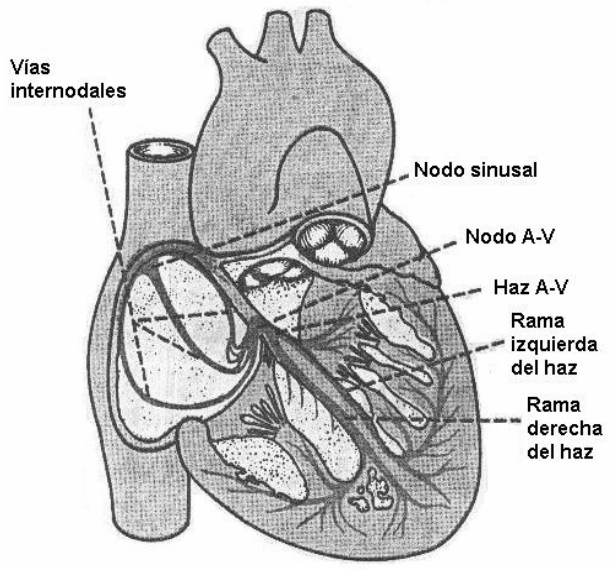


Figura 1.2 El corazón como una bomba de irrigación

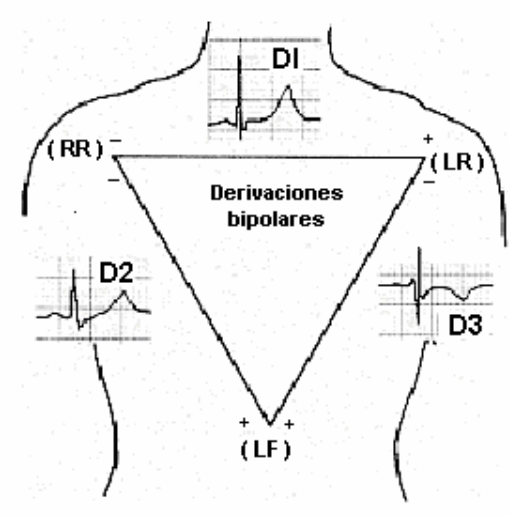


Figura 1.3 Registro extracorporal de los fenómenos celulares eléctricos

El registro de estas fluctuaciones de los potenciales durante el ciclo cardiaco es el electrocardiograma (ECG). (Fig. 1.4)

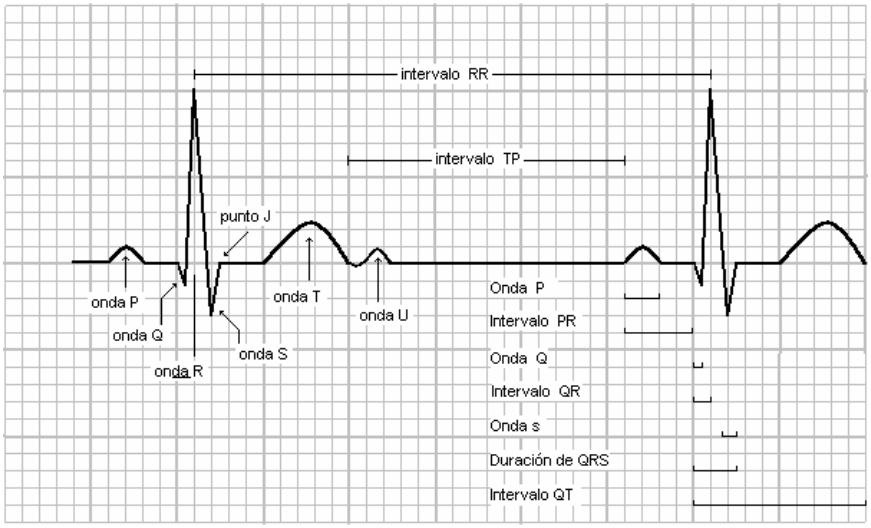


Figura 1.4 Componentes del electrocardiograma

1.2 Propiedades eléctricas.

El potencial de membrana en reposo de las células cardiacas del ser humano es aproximadamente de –90 mV (el interior es negativo con respecto al exterior) (Fig. 1.6). La estimulación produce un potencial de acción propagado que inicia la contracción, la despolarización es muy rápida (cambia de carga a +35 mV) e inclusive sobrepasa al potencial de membrana. En el corazón del ser humano los registros extracorporales de los fenómenos celulares eléctricos aparecen en el siguiente orden, como se observa en la Figura 1.5: la onda P que es positiva, el complejo QRS que presenta una porción positiva y otra negativa, la onda T que es positiva y en ocasiones la onda U que también es positiva. Todas las ondas con respecto al eje x, que es el eje isoeléctrico [5].

Los cambios en la concentración externa de K^+ afectan el potencial de membrana en reposo del músculo cardiaco, en tanto que los cambios en la concentración externa de Na^+ afectan la magnitud del potencial de acción [3].

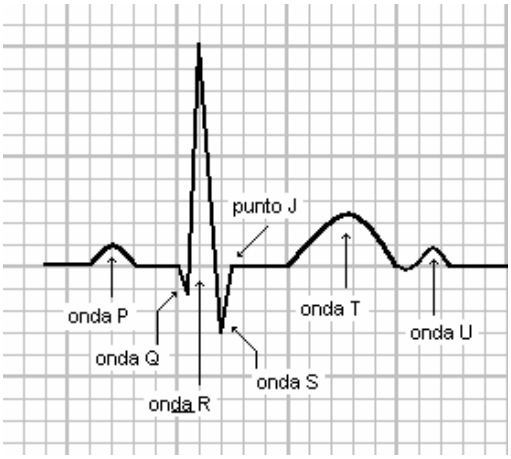


Figura 1.5 Electrocardiograma normal

En el músculo cardiaco, el tiempo de repolarización decrece cuando aumenta la frecuencia cardiaca. A una frecuencia de 75 latidos por minuto, la duración del potencial de acción (0.25 seg.) es casi 70 % mayor que a una frecuencia de 200 latidos por minuto (0.15 seg.).

En el músculo cardiaco en reposo, los iones positivos están situados en la cara externa de la pared celular, y los iones negativos en la pared interna. Una célula que se encuentra en este estado, se dice que está polarizada. La diferencia de concentraciones iónicas tiene como resultado un potencial eléctrico de menos sesenta milivolts. En forma espontánea o a consecuencia de un estímulo eléctrico externo puede invertirse la distribución de iones, en estas condiciones, la superficie externa de la célula adquiere una carga negativa, mientras que la superficie interna de la membrana toma una carga positiva, este fenómeno es llamado despolarización [3]. La despolarización iniciada en una zona de la célula desencadena un fenómeno similar en las regiones vecinas, de modo que el potencial de acción va propagándose a lo largo de la membrana de la fibra muscular. (Fig. 1.6)

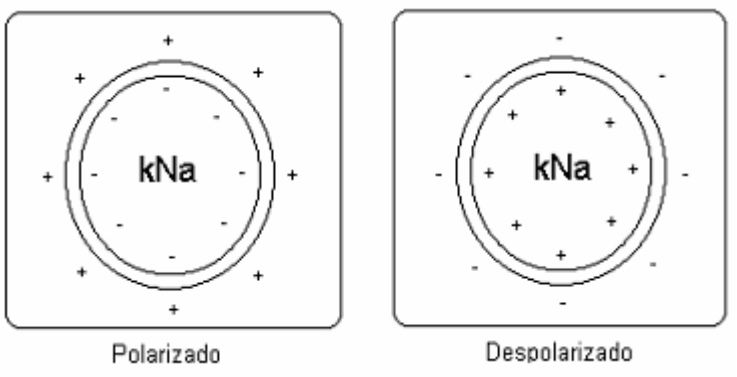


Figura 1.6 Condiciones eléctricas de polarización y despolarización

La despolarización del corazón se inicia en forma espontánea, a intervalos de un segundo aproximadamente, en el nodo sinoauricular (SA), y va extendiéndose sobre la aurícula. El paso de la despolarización de las aurículas a los ventrículos se retrasa aproximadamente 0.05 segundos a nivel del nodo auriculoventricular (AV). (Fig. 1.2) y (Fig. 1.7)

El miocardio sigue despolarizándose durante 0.12 segundos, luego recobra progresivamente su estado polarizado. Cuando una región del miocardio se vuelve negativa, siendo positiva el resto, el corazón se comporta como un dipolo, muestra propiedades de un par de cargas concentradas de polaridad opuesta. El dipolo cardiaco inicia un campo eléctrico a través de los líquidos corporales y en este campo las corrientes se dirigen del polo positivo al negativo a lo largo de infinidad de líneas, colocando electrodos en dos puntos cualesquiera, de este campo eléctrico, netamente separados, se puede medir la diferencia de potencial (el voltaje) entre los dos puntos. El voltaje entre dos electrodos dependerá de la magnitud del dipolo, de la resistencia de los tejidos corporales interpuestos, y de la orientación del dipolo en relación con los electrodos de registro. (Fig. 1.7)

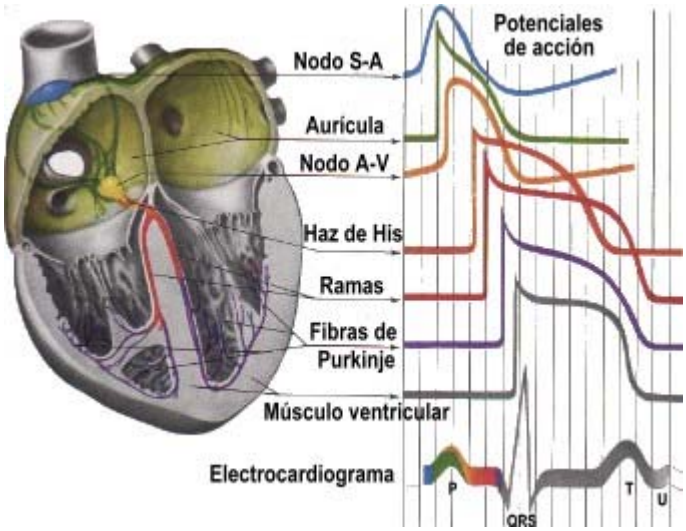


Figura 1.7 Actividad eléctrica registrada extra corporalmente

1.3 Aparato y técnicas de registro.

Para obtener un electrocardiograma, se usa un electrocardiógrafo, formado por electrodos, cables de conexión, selector de derivación, amplificador y registrador o inscriptor. Las corrientes eléctricas captadas por los electrodos son amplificadas [3]. Para registrar el electrocardiograma, el paciente debe estar acostado sobre la espalda en un lugar plano. El cuarto donde se efectúa el registro debe ser tibio (para no registrar escalofríos); la cama debe ser cómoda (para evitar tensión muscular y agitación) y el paciente debe estar convencido de que no sentirá dolor ni sensación molesta durante la prueba (para conseguir una buena relajación muscular), en los lugares donde se colocan los electrodos se debe limpiar la piel y debe colocarse pasta o jalea de electrodos, para que el contacto eléctrico sea completo. Se le pide al paciente que descanse, que no se mueva y que respire tranquilamente, evitando los suspiros durante el registro. Para tomar el registro, se conecta al amplificador la combinación apropiada de electrodos, para tomar sus respectivas derivaciones [3].

1.4 Preparar al paciente.

1. Retirar del paciente reloj, anillo, pulsera, cordón, etc..

2. Pedir al paciente relajarse y retirar lo que le “aprieta” o incomoda.

3. Con el paciente acostado, limpiar las áreas del cuerpo donde serán colocados los electrodos, con alcohol o éter y rasurando el área sólo en casos específicos.

4. Colocar gel en las áreas previamente limpias y poner los electrodos de manera adecuada.

5. Conectar a los electrodos ya posicionados, el cable del paciente.

1.5 Cuidados especiales con el cable del paciente.

El cable del paciente es uno de los accesorios más críticos del monitor del ECG, siendo así el responsable por el mayor número de fallas. Cuidar bien del cable es garantizar que aproximadamente el 70% de los defectos no ocurrirán. Aquí mencionamos como cuidar del cable de ECG:

1. Mantenerlo siempre que sea posible conectado al aparato.

2. No “plegarlo” de manera de provocar “fractura” del cable.

3. No utilizar soluciones abrasivas para limpiarlo.

4. Evitar el contacto con soluciones salinas.

1.6 Artefactos en el Electrocardiograma.

El ECG puede quedar alterado por diversos artefactos, debidos a movimientos del paciente, defectos técnicos diversos. Los artefactos más frecuentes son: (Fig. 1.8)

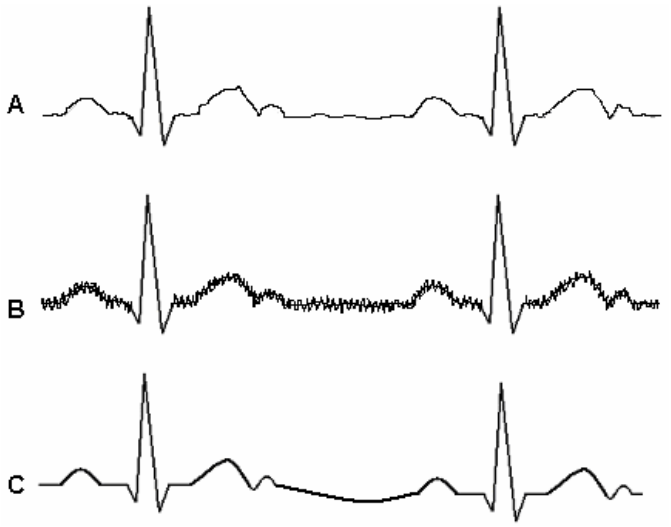


Figura 1.8 Posibles artefactos en un electrocardiograma

A. El temblor muscular, se caracteriza por pequeñas oscilaciones irregulares, de frecuencia variable, superpuestas al trazo; se deben a tensión muscular, movimientos bruscos, inspiraciones profundas, y también se observan en casos de tirotoxicosis, parkinsonismo, o simplemente contacto defectuoso entre los electrodos y la piel [3].
B. Corriente alterna, que se presenta como ondas regulares en dientes de sierra, a razón de 60 ciclos por segundo; se deben a una interferencia originada por algún tipo de aparato eléctrico conectado en el mismo circuito.
C. Inestabilidad de la línea basal, que se caracteriza por desplazamientos del electrocardiograma, lentos o rápidos, hacia arriba o hacia abajo; se deben a contactos sucios o mal apretados, también a movimientos respiratorios irregulares.

1.7 Fuentes de ruido e interferencia en el ECG.

La presencia del ruido en el registro de biopotenciales es un hecho prácticamente inevitable. En el registro del ECG la existencia de ruido se debe a muchas causas. Algunas de ellas son controlables, pero otras han sido poco investigadas. Un mejor conocimiento del ruido y sus causas puede ayudar en el paso posterior de la cadena de procesado, que es su eliminación.

Como paso previo a citar las fuentes de ruido cabe definir que es ruido. Una definición común al hablar de sistemas electrónicos considera como aquella señal ajena a la señal de interés y que es susceptible de provocar un error en nuestro sistema de medida. Así el ruido podremos clasificarlo según sea una señal determinista o aleatoria, o bien según su origen: externo o interno al sistema de medida. Normalmente se utiliza el término ruido cuando el origen es interno al propio sistema de medida y la naturaleza de la señal suele ser aleatoria.

Por contra, el término interferencia se aplica a aquellas señales externas al sistema de medida, cuya evolución temporal suele seguir una ley preestablecida que puede ser conocida de antemano, aunque su valor en un instante determinado pueda venir caracterizado por una variable aleatoria. Por ejemplo las interferencias debidas a la red de distribución eléctrica, y las producidas por equipos eléctricos o electrónicos próximos al entorno de medida. Así las fuentes de ruido e interferencia que se encuentran habitualmente en un registro electrocardiográfico son:

- Electromiograma (EMG)
- Interfaz electrodo-paciente
- Sistema de medida
- Fuentes de interferencia
- Red de distribución eléctrica
- Otras fuentes: ordenadores, monitores, equipos electrónicos.

1.8 Ruido o interferencia.

Para evitar el ruido en el trazado de ECG es necesario:

1. Utilización de un cable tierra de manera adecuada.
2. Tener cuidado con la calidad del electrodo.
3. Certificarse que el cable de paciente se encuentra en buen estado.

Todavía existen ruidos de otras procedencias no eléctricas, como por ejemplo: temblor muscular, movimientos respiratorios, etc., que deberán ser evitados con una buena preparación del paciente.

1.9 Electrocardiógrafo.

Uno de los aspectos mas importantes en la obtención de señales eléctricas tomadas del cuerpo humano, es sin duda, la eliminación del ruido presente en dicha señal que llega a ser hasta 10 veces más grande que la señal eléctrica cardiaca de interés, esto implica la realización de un circuito lo suficientemente eficiente con el fin de amplificar la señal de interés y eliminar el ruido.

Se tiene que hacer una correcta selección del circuito a utilizar; uno de los más eficientes y que mejor respuesta proporciona, es sin duda un amplificador de instrumentación. Este amplificador de instrumentación consta de un amplificador diferencial mejorado, el cual presenta una ganancia ajustable y una alta resistencia de entrada acoplado con un tercer operacional que proporciona una ganancia unitaria, con sus cuatro resistencias iguales. (Fig. 1.9)

La señal de voltaje en la piel debido a los latidos del corazón oscila de 1 a 1.5 mV (Vint), en tanto que el voltaje de ruido del cuerpo puede ser de varios decimos de volts más (Vruido). De modo que sería imposible hacer una medición de ECG (Electrocardiograma) con un amplificador de entrada única. Lo que se necesita es un amplificador que pueda distinguir entre Vint y Vruido y amplifique

Vint, en este caso el amplificador de instrumentación es el indicado. La señal que se obtiene a la salida del circuito puede mejorar si le aplicamos un filtro que elimine considerablemente el nivel de ruido que pueda presentar. La señal de interés oscila de 1 a 1.4 [14] Hz (palpitaciones en reposo del corazón 60-72 por segundo).

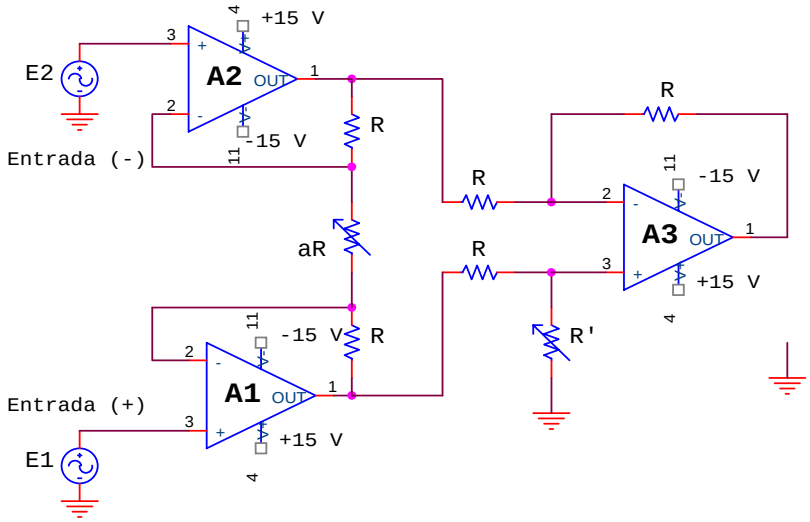


Figura 1.9 Amplificador de instrumentación

Las señales eléctricas que se toman del cuerpo, se toman por medio de electrodos conectados a un amplificador, estos electrodos contienen un gel que hace contacto con la piel y aumenta la conductividad eléctrica, sin embargo dicha señal hay que procesarla para poder observar algo. Existen regiones especiales del cuerpo humano donde se colocan estos electrodos, estos lugares han sido estudiados por médicos y han coincidido en doce lugares que presentan respuestas adecuadas, cada una de ellas es llamada derivación.

1.10 Derivaciones electrocardiográficas.

Los potenciales eléctricos pueden ser recogidos de la superficie corporal mediante dos electrodos, uno conectado al polo positivo, el otro al polo negativo. La disposición específica que guardan los electrodos recibe el nombre de derivación. Se han empleado mas de cuarenta derivaciones distintas en los

registros electrocardiográficos; pero habitualmente son 12 las que más se usan [4]:

Tres derivaciones bipolares de miembros, llamadas D1, D2 y D3.
Tres derivaciones unipolares de miembros, llamadas aVR, aVL y aVF.
Seis derivaciones precordiales, llamadas V1, V2, V3, V4, V5, V6.

Las tres derivaciones bipolares de miembros se consiguen disponiendo los electrodos de la siguiente manera:

D1: brazo izquierdo (+) y brazo derecho (-).
D2: pierna izquierda (+) y brazo derecho (-).
D3: pierna izquierda (+) y brazo izquierdo (-).

1.11 Colocación de electrodos.

Es importante observar la posición correcta de los electrodos, pues solamente de esta forma podemos garantizar que la señal de ECG será fiel a la realidad del paciente. Además, la preparación del paciente es muy importante, limpiar bien el lugar donde se colocará el electrodo, usar siempre gel adecuado, si es necesario limpiar o rasurar la región de la piel donde será posicionado el electrodo. (Fig. 1.3) y (Fig. 1.11)

Se colocan los electrodos sobre las muñecas izquierda y derecha, y un poco arriba del tobillo izquierdo, para las derivaciones bipolares. Cada una de estas derivaciones registra la diferencia de potencial entre los segmentos a los cuáles se conecta. (Fig. 1.10) Las tres derivaciones unipolares de miembros se consiguen con la siguiente disposición de los electrodos:

aVR: brazo derecho (+) y la CTg (-).
aVL: brazo izquierdo (+) y la CTg (-).
aVF: pie izquierdo (+) y la CTg (-).

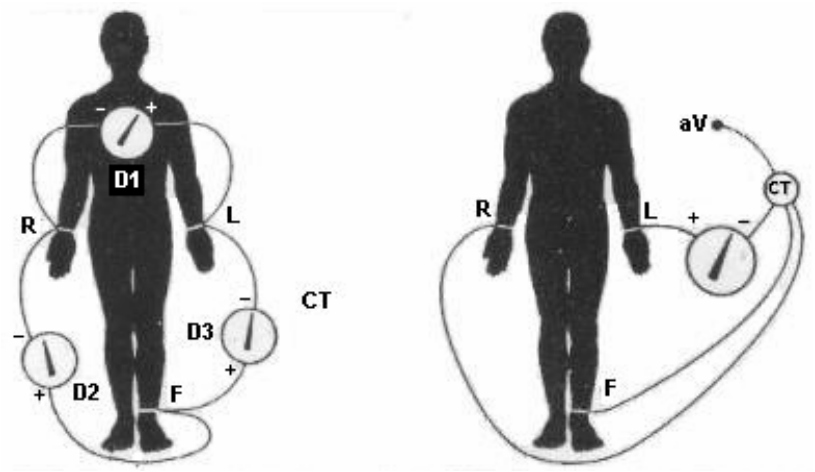


Figura 1.10 Derivaciones electrocardiográficas bipolares y unipolares

Las seis derivaciones precordiales se consiguen disponiendo como sigue los electrodos [4]: (Fig. 1.11)

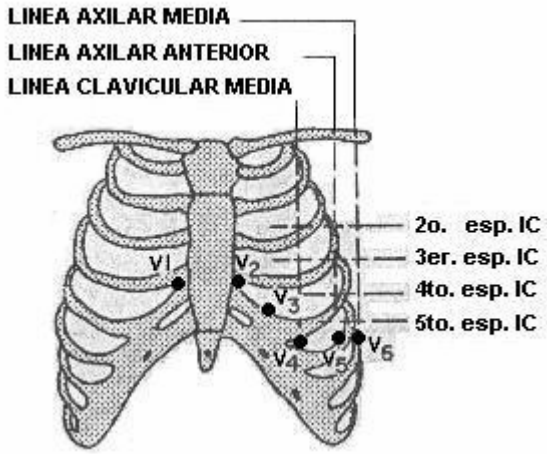


Figura 1.11 Derivaciones precordiales

V1: 4º. Espacio intercostal, en el borde derecho del esternón (+), y la CTw (-).

V2: 4º. Espacio intercostal, en el borde izquierdo del esternón (+), y la CTw (-).

V3: punto medio entre v2 y v4 (+), y la CTw (-).

V4: 5º. Espacio intercostal, a nivel de la línea medioclavicular (+), y la CTw (-).

V5: línea axilar anterior, a la misma altura que V4 (+), y la CTw (-).

V6: línea axilar media, al mismo nivel que V4 (+), y la CTw (-).

Recapitulando, la función del corazón es la de mantener con vida a todos los órganos presentes en el ser humano mediante la irrigación de la sangre. El corazón funciona por medio de un tejido marcapaso que es el sistema de conducción que normalmente propaga los impulsos por todo el corazón, debido a que los líquidos corporales son buenos conductores. Esto origina cambios de polaridad en las paredes celulares y en los tejidos que pueden ser captados extracorporalmente en determinadas regiones del cuerpo humano, donde estudiosos en el área han coincidido en que son los lugares más apropiados para obtenerlos. Para poder obtener estos impulsos es necesario implementar un circuito electrónico a fin de amplificar la señal de interés y atenuar el nivel de ruido, pues esta llega a ser de varios decimos de milivoltios más grande y no se obtendría la señal de interés. En la realización de la tesis se utilizaron únicamente las derivaciones bipolares llamadas D1, D2 y D3 mencionadas en la sección 1.10 y 1.11.

CAPITULO II.

Procesamiento digital de señales.

2.1 Señales.

Una señal se define como una cantidad física que varía con el tiempo , el espacio o cualquier otra variable o variables independientes, esta señal lleva consigo determinado tipo de información, describimos una señal como una función de una o más variables independientes [6]. Por ejemplo las funciones:

$$s_1(t) = 7\,t$$
$$s_2(t) = 14\,t^2 \qquad (2.1)$$

Describen dos señales, una que varía linealmente con la variable independiente *t* (tiempo) y una segunda que varía cuadráticamente con respecto a *t*, sin embargo pueden definirse con dos o más variables. Existen gran variedad de señales en la naturaleza y por ende en el ser humano. Si una señal es función de una única variable independiente, la señal se denomina unidimensional y si es función de *M* variables independientes se denomina M-dimensional. Asociados a estas señales naturales se encuentran los medios con los que son generados, por lo tanto, la forma en que se generan las señales se encuentra asociada con un sistema que responde ante un estímulo o fuerza. El estímulo en combinación con el sistema se llama fuente de señal [6].

Un sistema se puede definir como un conjunto de elementos que realiza una operación sobre una señal. Cuando pasamos una señal a través de un sistema, como en el caso del filtrado, decimos que hemos procesado la señal, este procesamiento implica la separación de la señal deseada del ruido y la

interferencia. Un sistema no solo puede incluir dispositivos físicos, sino también realizaciones de operaciones mediante software. En el procesamiento digital de señales, se utiliza frecuentemente una PC (o un sistema digital distinto), las operaciones realizadas sobre una señal constan de varias operaciones matemáticas especificadas por un programa (software), en este caso el programa representa una implementación del sistema en software.

El método o conjunto de reglas para implementar el sistema mediante un programa que ejecuta las operaciones matemáticas correspondientes se denomina algoritmo. Generalmente existen muchos algoritmos para implementar un sistema, tanto en software como en hardware, en la práctica se prefieren diseñar algoritmos que sean computacionalmente eficientes, rápidos y de implementación simple [8].

2.2 Señales en tiempo continuo y en tiempo discreto.

Señales en tiempo continuo o señales analógicas están definidas para todos los valores del tiempo y pueden tomar cualquier valor en el intervalo continuo (*a, b*), donde a puede ser $-\infty$ y *b* puede ser ∞. (Fig. 2.1)

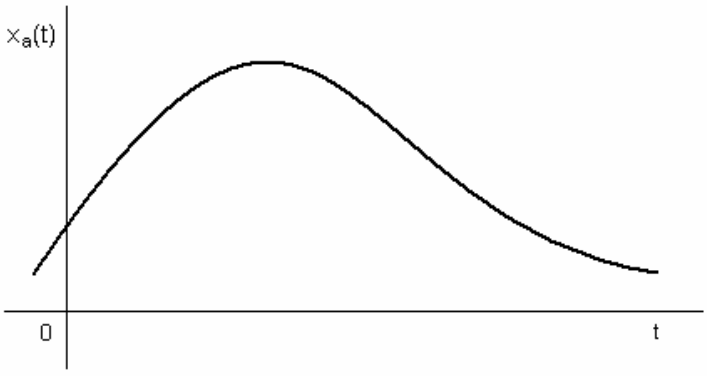


Figura 2.1 Representación de un segmento de una señal analógica

Las señales en tiempo discreto están definidas solo para ciertos valores de tiempo, estos instantes del tiempo no necesitan estar equidistantes. En la práctica las señales en tiempo discreto pueden originarse de dos formas:

a) Eligiendo valores de una señal analógica en determinados instantes de tiempos. Este proceso se denomina muestreo. Todos los aparatos de medición que proporcionan medidas en instantes de tiempos regulares generan señales en tiempo discreto.

b) Acumulando una variable a lo largo de un determinado periodo de tiempo. Por ejemplo, cuantas personas pasan determinada calle en una hora, [6] etc. (Fig. 2.2)

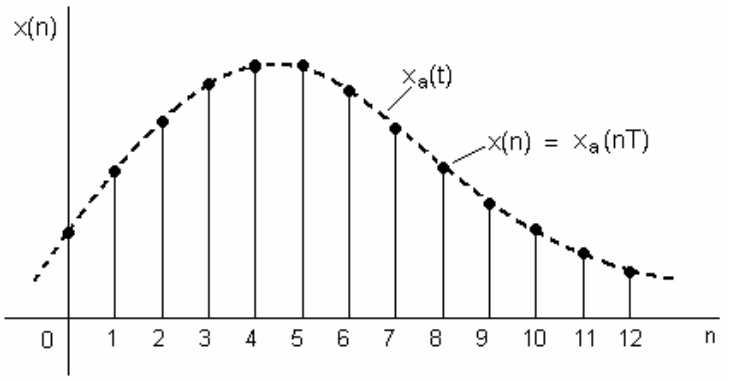


Figura 2.2 Representación de un segmento de una señal en tiempo discreto

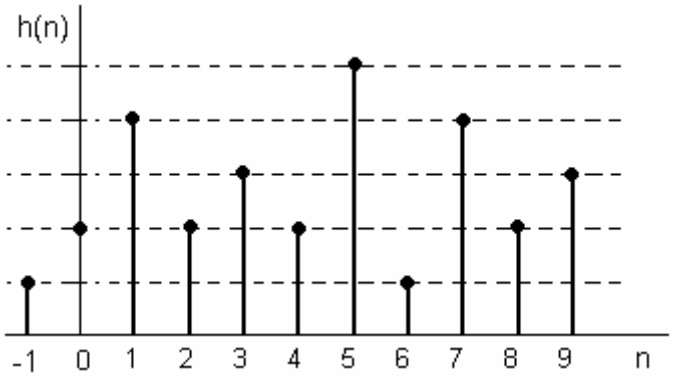


Figura 2.3 Señal digital con cinco valores posibles de amplitud

La figura 2.3 muestra una señal digital con 5 valores posibles [6]. Para que una señal pueda ser procesada digitalmente ha de ser en tiempo discreto y debe tomar valores discretos (debe de ser una señal digital). Si la señal a procesar es analógica, se convierte a digital muestreándola en el tiempo y posteriormente cuantificando sus valores en un conjunto discreto [7].

2.3 Conversión analógico–digital y digital–analógico.

La mayoría de las señales de interés práctico, señales de voz, biológicas, sísmicas, radar, sonar, etc., y de distintos tipos de comunicación, como las señales de audio, AM, FM y video, son analógicas. Para procesar señales analógicas por medios digitales es necesario convertirlas a formato digital, esto es, transformarlas en una secuencia de números de precisión finita. Este procesamiento se denomina conversión analógico-digital por sus siglas (A/D) y sus dispositivos correspondientes que lo hacen posible ADC's. De manera conceptual esta conversión puede darse de forma general en tres pasos, tal como se indica. (Fig. 2.4)

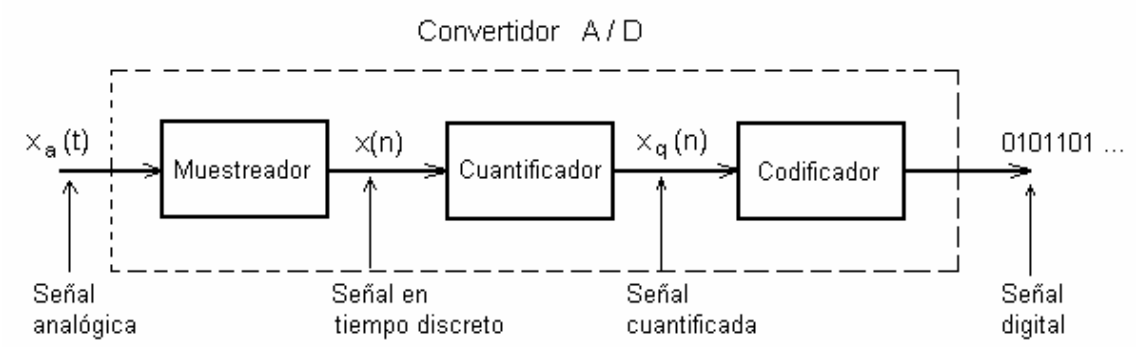


Figura 2.4 Partes básicas de un convertidor analógico-digital (A / D)

1) Muestreo: Es la conversión de una señal de tiempo continuo a una señal en tiempo discreto, obtenida tomando muestras de la señal analógica en instantes de tiempo discreto. Si $x_a(t)$ es la entrada al muestreador, la salida es $x_a(nT) = x(n)$ donde T es el intervalo de muestreo.

2) Cuantificación: Es la conversión de una señal en tiempo discreto con valores continuos a una señal en tiempo discreto con valores discretos (señal digital). El valor de cada muestra de la señal se representa mediante un valor seleccionado de un conjunto finito de valores posibles. La diferencia entre la muestra sin cuantificar *x(n)* y la salida cuantificada $x_q(n)$ se denomina error de cuantificación.
3) Codificación: En el proceso de codificación, cada valor discreto $x_q(n)$ se representa mediante una secuencia binaria de *b* bits [8].

En principio la señal analógica puede reconstruirse a partir de sus muestras, siempre que la tasa de muestreo sea lo suficientemente alta como para evitar el problema del aliasing. Por otra parte, la cuantificación es un proceso no reversible que resulta en distorsión de la señal, esta distorsión depende de la precisión, medida en función del número de bits del proceso de conversión A/D. Los factores que afectan la elección de la precisión deseada del convertidor A /D son el costo y la tasa de muestreo, el costo aumenta con la precisión y/o la tasa de muestreo. (Fig. 2.5)

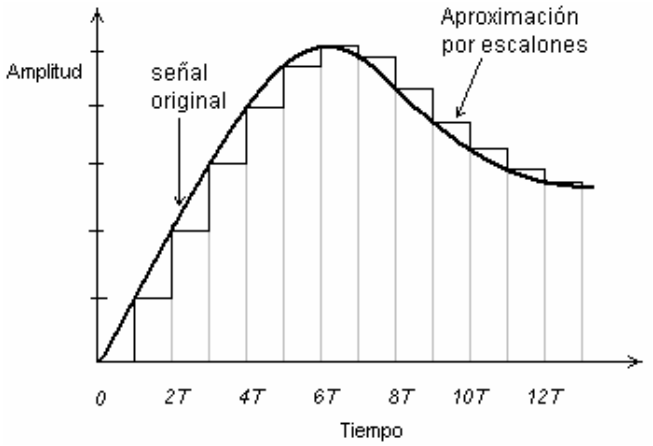


Figura 2.5 Conversión (A/D) por aproximación de escalones

2.4 Teorema del muestreo.

Una de las maneras más comunes de muestrear una señal es el muestreo periódico uniforme y se describe mediante la relación:

$$x(n) = x_a(nT) \qquad -\infty < n < \infty \tag{2.2}$$

Donde *x(n)* es la señal en tiempo discreto obtenida tomando muestras de la señal analógica $x_a(t)$ cada *T* segundos, como se muestra en la figura 2.6. El intervalo de tiempo *T* entre dos muestras sucesivas se denomina periodo de muestreo o intervalo de muestreo, y su inversa *Fs = 1/T* se llama velocidad de muestreo (muestras por segundo) o frecuencia de muestreo [7].

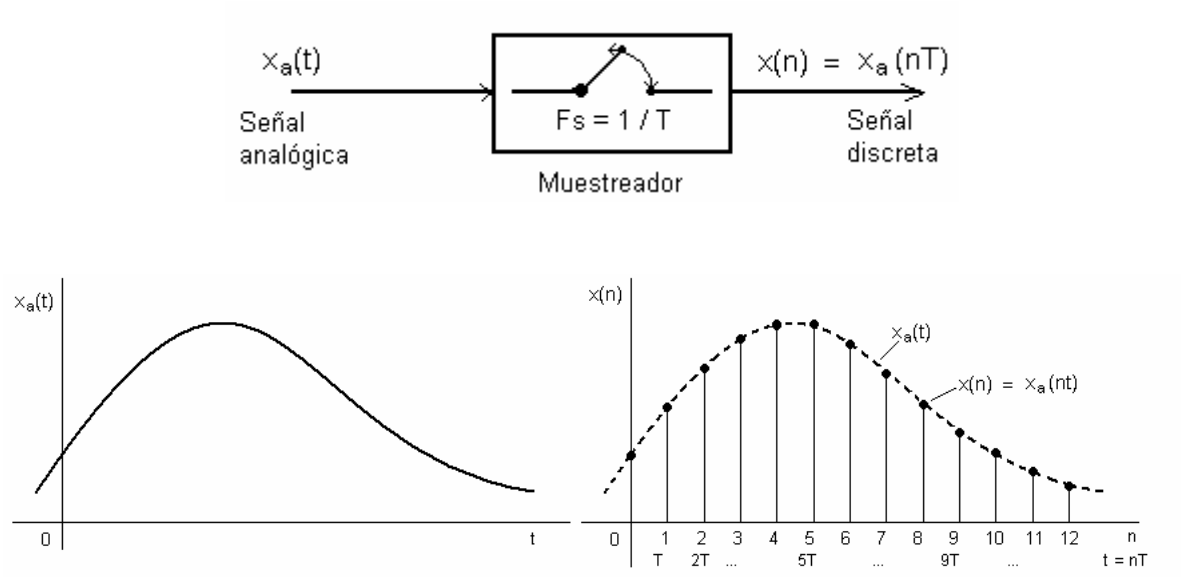


Figura 2.6 Muestreo periódico de una señal analógica

Si la frecuencia de muestreo es suficientemente alta, comparada con la frecuencia máxima F_{max} presente en una señal en tiempo continuo, la señal original puede reconstruirse a partir de estas muestras [8].

Para poder elegir el período de muestreo T o la velocidad de muestreo F_s correcta es necesario cierta información sobre las características de la señal que va a ser muestreada, en particular información sobre el contenido frecuencial de la señal. Si conocemos la máxima frecuencia de una determinada clase de señal, se puede especificar la velocidad de muestreo necesaria para convertir las señales analógicas en señales digitales. El conocimiento de $F_{máx}$ nos permite seleccionar la velocidad de muestreo apropiada. La frecuencia más alta de la señal analógica que puede reconstruirse sin ambigüedad cuando la señal se muestrea a una velocidad $F_s / 2$ o por debajo de $– F_s / 2$ produce muestras que son idénticas a las correspondientes frecuencias dentro del intervalo $- F_s / 2 \leq F \leq F_s / 2$. Para evitar las ambigüedades que resultan del aliasing, se debe seleccionar una velocidad de muestreo lo suficientemente alta, se debe escoger $F_s / 2$ mayor que $F_{máx}$ [6]. Esto es:

$$F_s > 2\, F_{máx} \tag{2.3}$$

2.5 Cuantificación y errores de cuantificación.

Las principales funciones involucradas en la conversión analógico–digital son el muestreo, la cuantificación de la amplitud y la codificación. Cuando el valor de cualquier muestra cae entre dos estados de salida adyacentes permitidos, se debe de leer como el estado de salida permitido más cercano al valor real de la señal. El proceso de representación de una señal continua o analógica mediante un número finito de estados discretos se denomina cuantificación de la amplitud.
Cuantificación: significa la transformación de una señal continua (analógica) en un conjunto de estados discretos. El estado de salida de cualquier muestra cuantificada se describe entonces mediante un código numérico. El proceso de representar el valor de una muestra mediante un código numérico (tal como el código binario) se denomina codificación.
Codificación: es el proceso de asignación de una palabra o código digital a cada uno de los estados discretos. El error cometido al representar la señal de valor

continuo por un conjunto finito de valores discretos se denomina error de cuantificación o ruido de cuantificación.

Debido a que el proceso de cuantificación es un proceso de aproximación en el que la cantidad analógica se aproxima mediante un número digital finito, el error de cuantificación es un error de redondeo. Es claro que, mientras más fino sea el nivel de cuantificación, más pequeño será el error de redondeo. En la figura 2.7 se muestra una entrada analógica *x(t)* y la salida discreta *y(t)*, la cual está en la forma de una función escalonada. El error de cuantificación *e(t)* es la diferencia entre la señal de entrada y la salida cuantificada [7].

$$e(t) = x(t) - y(t) \tag{2.4}$$

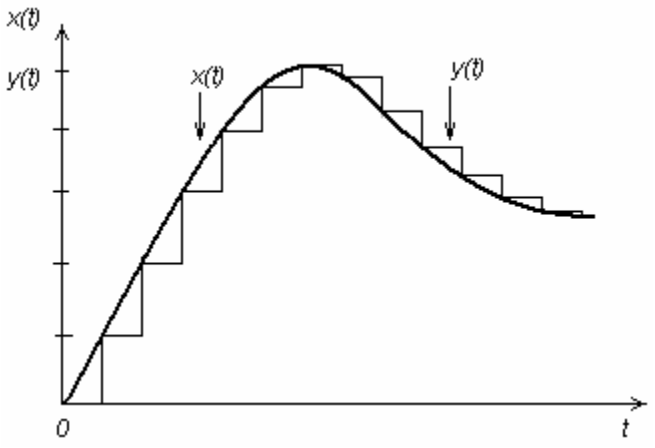


Figura 2.7 Entrada analógica x(t) y salida discreta y(t)

2.6 Procesamiento digital de señales.

El rápido desarrollo de la tecnología de circuitos integrados ha estimulado el desarrollo de computadoras digitales más potentes, pequeñas, rápidas y baratas. Estos circuitos digitales han hecho posible construir sistemas digitales altamente sofisticados, capaces de realizar funciones y tareas de procesamiento de señal digital, que normalmente eran difíciles y caros con circuitos de

procesamiento de señales analógicas. Muchas de las tareas del procesamiento de señal que convencionalmente se realizaban analógicamente se realizan hoy mediante hardware digital más barato y a menudo más fiable. En particular, el procesamiento de señal digital permite operaciones programables. Por medio de software se pueden modificar fácilmente las funciones de procesamiento de señal para que sean realizadas por el hardware. Por tanto, el hardware digital y el software asociado proporcionan un mayor grado de flexibilidad en el diseño de sistemas. Además, normalmente se consigue mayor precisión con el hardware y el software digital en comparación con los circuitos analógicos y los sistemas de procesamiento de señal analógicas [6].

El procesamiento digital de señal analógicas tiene algunos inconvenientes, el primero y más importante, la conversión de una señal analógica en digital obtenida muestreando la señal y cuantificando las muestras, produce una distorsión que nos impide la reconstrucción de la señal analógica original a partir de las muestras cuantificadas. El control de esta distorsión se logra con la elección apropiada de la velocidad de muestreo y la precisión del proceso de cuantificación. En segundo lugar, existen efectos debidos a la precisión finita que deben ser considerados en el procesado digital de las muestras cuantificadas. Para muchas señales de gran ancho de banda, se requiere procesamiento en tiempo real. Para tales señales, el procesado analógico u óptico, son las únicas soluciones válidas.

2.7 Elementos de un sistema de procesamiento digital de señales.

La mayor parte de las señales que aparecen en los ámbitos de la ciencia y la ingeniería son de naturaleza analógica, son funciones de una variable continua, como el tiempo o el espacio y normalmente toman valores en un rango continuo. Tales señales pueden ser procesadas directamente por sistemas analógicos adecuados con el propósito de cambiar sus características o extraer cualquier

información deseada. Se dice entonces que la señal ha sido procesada directamente en forma analógica. (Fig. 2.8)

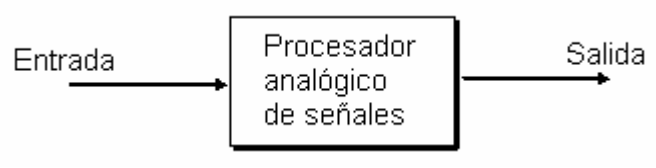


Figura 2.8 Diagrama a bloques del procesamiento analógico

Para realizar el procesamiento digitalmente se necesita una interfaz entre la señal analógica y el procesador digital. Esta interfaz se denomina convertidor analógico–digital (ADC). La señal de salida es una señal adecuada como entrada al procesador digital. El procesador digital de señales puede ser una PC digital o un pequeño microprocesador programado para realizar las operaciones deseadas sobre la señal de entrada. Los dispositivos programables proporcionan gran flexibilidad de cambiar las operaciones de procesamiento de señal mediante un cambio del software.

En aplicaciones donde la salida digital del procesador digital de señales se ha de entregar en forma analógica, como en comunicaciones digitales, se debe proporcionar otra interfaz, desde el dominio digital al analógico, tal interfaz se denomina convertidor digital-analógico (DAC), de este modo la señal se entrega al usuario en forma analógica. (Fig. 2.9)

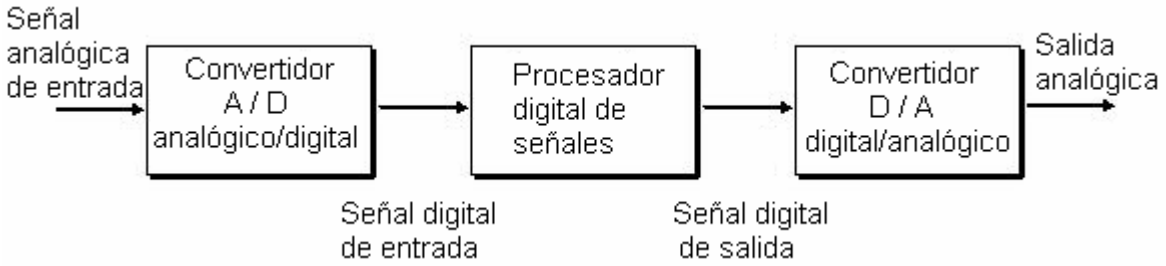


Figura 2.9 Diagrama a bloques del procesamiento digital

2.8 Ventajas del procesamiento digital de señales frente al analógico.

Un sistema digital programable permite flexibilidad a la hora de reconfigurar las operaciones de procesamiento digital de señales sin más que cambiar el programa. La reconfiguración de un sistema analógico implica habitualmente el rediseño del hardware, seguido de la comprobación y verificación para ver que opera correctamente y una elección adecuada de la precisión del formato de datos al elegir procesador de señales. Las tolerancias en los componentes de los circuitos analógicos hacen que para el diseñador del sistema sea extremadamente difícil controlar la precisión de un sistema de procesamiento analógico de señales. En cambio un sistema digital permite un mejor control de los requisitos de precisión, tales requisitos resultan a su vez en la especificación de la precisión de los convertidores A/D y del procesador digital de señales, en términos de la longitud de la palabra, aritmética de punto fijo o punto flotante, etc.

Las señales digitales se almacenan fácilmente en soportes magnéticos (discos), sin deterioro o pérdida de fidelidad de la señal, como consecuencia las señales se hacen transportables y pueden procesarse en tiempo no real en un laboratorio remoto. El método de procesamiento digital de señales también posibilita la implementación de algoritmos de procesamiento de señal más sofisticados, generalmente es muy difícil realizar operaciones matemáticas precisas sobre señales en formato analógico, pero esas mismas operaciones pueden efectuarse de modo rutinario sobre una PC u otro sistema digital utilizando software, en algunos casos esta es más barata que su equivalente analógico[6]. una limitación práctica es la velocidad de operación de los convertidores ADC y DAC de los procesadores digitales de señales [6].

2.9 Filtros Digitales.

En el procesamiento digital de señales un filtro se utiliza para dejar pasar determinadas frecuencias y atenuar otras. Los filtros se suelen clasificar según sus características en el dominio de la frecuencia como paso bajo, paso alto, paso banda y elimina banda. En el diseño de filtros selectivos, las características deseadas del mismo se especifican en el dominio de la frecuencia, en función de la respuesta del filtro en magnitud y en fase. Existen filtros de respuesta finita al impulso conocidas como FIR y filtros de respuesta infinita al impulso conocidas como IIR. En el proceso del diseño del filtro determinamos los coeficientes del filtro FIR o IIR. La elección del tipo de filtro depende de la naturaleza del problema y de las especificaciones de la respuesta en frecuencia deseada [10]. (Fig. 2.10)

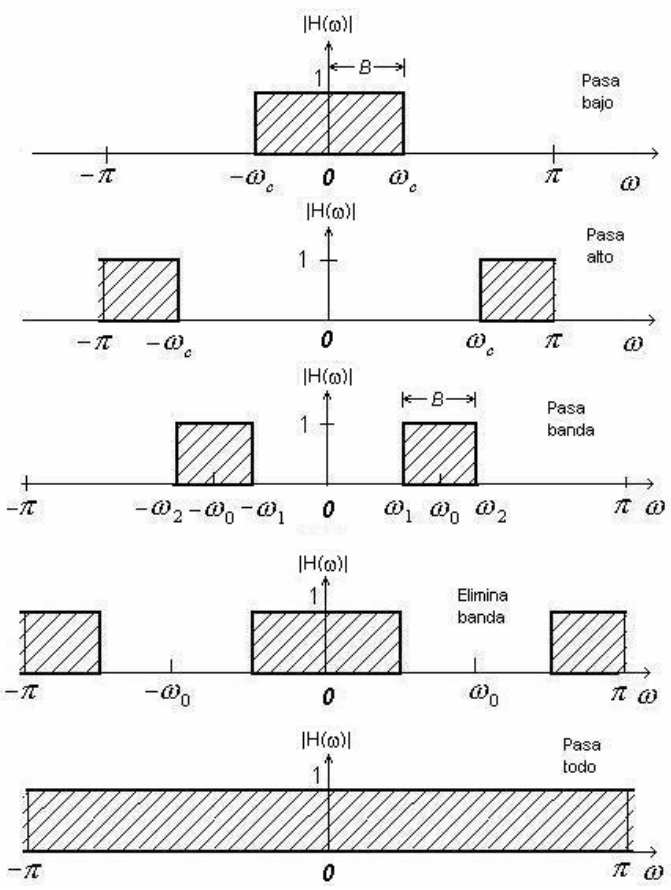


Figura 2.10 Tipos de Filtro

En la práctica, los filtros FIR se emplean en problemas de filtrado donde hay un requisito de fase lineal dentro de la banda de paso del filtro [9]. Un filtro IIR tiene lóbulos laterales menores en la banda de rechazo que un filtro FIR con el mismo número de parámetros, por esta razón si se puede tolerar alguna distorsión de fase o ésta no es importante, se prefiere un filtro IIR porque su implementación involucra menos parámetros, requiere menos memoria y tiene menor complejidad computacional. En la actualidad el diseño de filtros digitales se facilita por la disponibilidad de numerosos programas de software donde se puede especificar el tipo y orden del filtro y además se pueden calcular sus coeficientes. (Fig. 2.11)

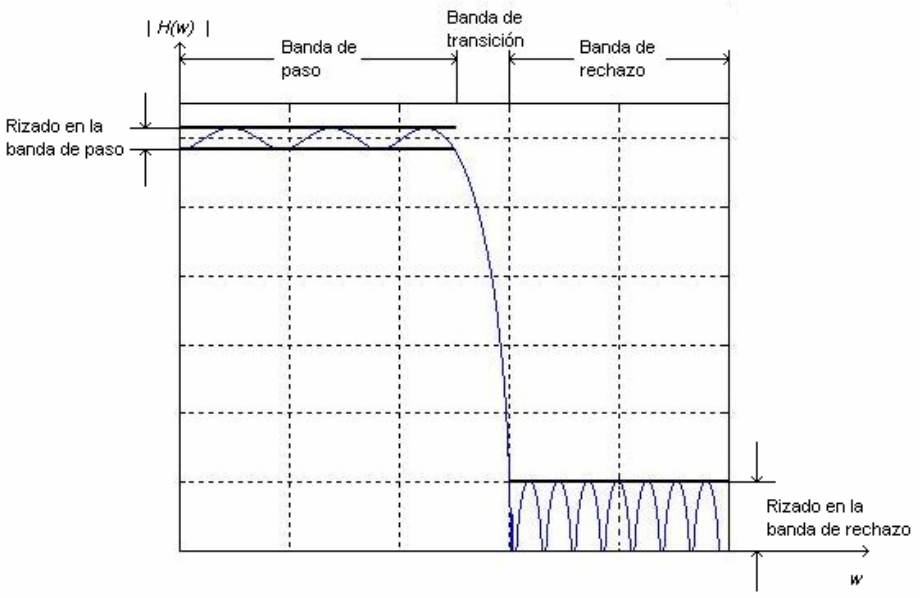


Figura 2.11 Característica de magnitud de filtros

Características de los filtros digitales FIR

1. Los filtros FIR pueden ser diseñados en fase lineal. (Fig. 2.12)
2. Los filtros FIR realizables son no recursivos e inherentemente estables, siendo de tamaño finito.
3. Los filtros FIR pueden ser implementados eficientemente en sistemas DSP [10].

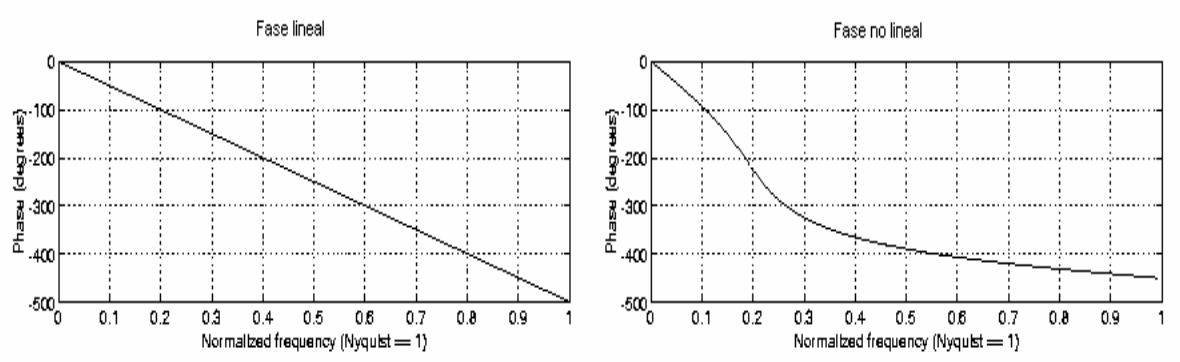


Figura 2.12 Respuesta en fase de filtros FIR e IIR

Los filtros analógicos están formados por resistores, capacitores, inductores, amplificadores operacionales los cuáles no poseen mucha estabilidad térmica. Mientras los filtros digitales presentan algunas ventajas.

1. Estabilidad térmica: los cambios de temperatura afectan a los resistores, capacitores e inductores, mientras que con los filtros digitales estos son eliminados pues utilizan sumadores, multiplicadores, retardos unitario y registros de almacenamiento, registros que son bastante menos sensibles a los cambios de temperatura.
2. Precisión : al incrementarse el tamaño de bit en el registro del procesador aumenta el rango dinámico, estabilidad y tolerancia a la respuesta en frecuencia.
3. Adaptabilidad. La respuesta en frecuencia del filtro digital puede ser cambiado eficientemente al leer un nuevo set de coeficientes de filtro del registro de coeficientes de la memoria.

Desventajas

1. Ancho de banda limitado: como resultado del proceso de muestreo del convertidor A/D, el ancho de banda para una señal real discreta está limitado a la mitad de la frecuencia de muestreo.

2. La implementación de un sistema discreto puede resultar en una degradación de la calidad de la señal original. Este efecto resulta al usar datos y registro de coeficientes con un número finito de bits.

Como complemento a lo descrito anteriormente, se ejemplifica el diseño de un filtro pasa bajos FIR de orden 15, con una frecuencia de corte de 50 Hz. y una frecuencia de muestreo de 200 muestras por segundo (m/s) utilizando la función *remez* de Matlab [9]. Esta función permite calcular los coeficientes de un filtro FIR, y su algoritmo se basa en la teoría de aproximación de Eugeny Yakoulevich, esta función requiere de los siguientes parámetros: Vector de frecuencias normalizadas (f), cuyo valor máximo 1, es igual a 100 para este ejemplo, (200m/s / 2, según el teorema de muestreo) y el valor mínimo es 0. Vector de magnitudes (*a*), los valores de este vector tienen una correspondencia con los valores del vector de frecuencias normalizadas, los vectores (*f*) y (*a*) especifican las características en frecuencia y en magnitud del filtro. Por último el orden del filtro de la respuesta en la frecuencia indicada por la variable (*b*), es de 15, los coeficientes de este vector generados son de 16 valores, pues es el orden del filtro más 1.

La frecuencia a la que debe cortar el filtro es de 50 Hz, que corresponde al valor 0.5 del vector de frecuencias normalizadas, en este punto la respuesta en magnitud del filtro debe ser de 0.7 (-3 dB) menor a la magnitud de la señal de entrada, es el lugar donde debe cortar el filtro. (Fig. 2.1e)

El siguiente bloque es la representación de lo descrito anteriormente en código de Matlab.

```
» f=[0  0.4  0.5  0.6  0.7  0.8  0.9  1];    vector de frecuencias normalizadas
» a=[1  1  0.7 0.2 0.1 0.01 0.0001 0 ];  vector  de magnitudes
» b= remez(15,f,a);              obtención de los coeficientes del filtro remez
» [h,w]=freqz(b,1,200);         obtiene la respuesta en frecuencia del filtro
» plot(f,a,abs(h))               muestra la forma de la respuesta del filtro
```

Note que la magnitud se mantiene en 1 hasta llegar al valor 0.5 del vector de frecuencias donde cae a 0.7 (-3dB), los siguientes valores del vector de magnitud van decrementándose paulatinamente.

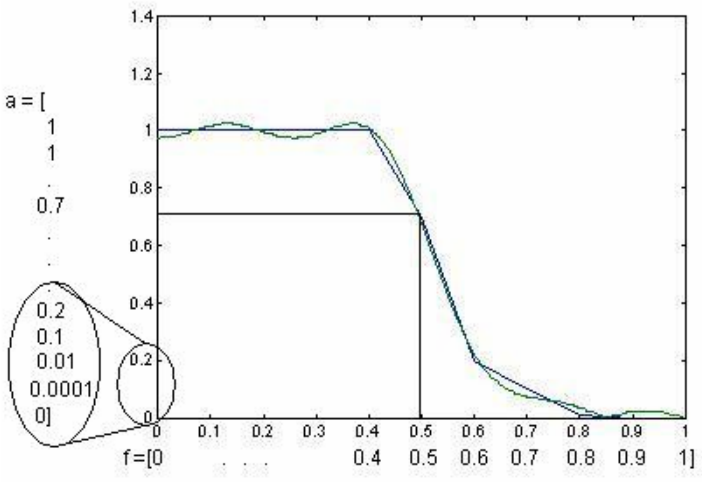


Figura 2.1e Respuesta en magnitud del filtro *remez*

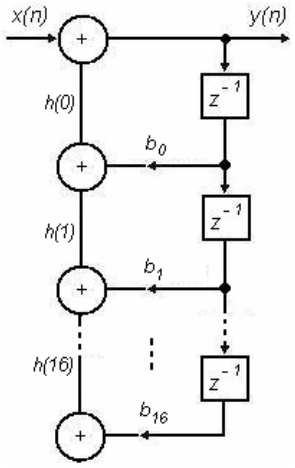


Figura 2.2e Estructura del filtro FIR de orden 15

La figura 2.2e muestra la estructura del filtro FIR para el ejemplo 2.1e, donde *x(n)* es la entrada, *y(n)* la salida con los datos filtrados, b_0 ... b_{16} son los coeficientes del filtro, *h(n)* son las muestras almacenadas y Z^{-1} los retardos que cada uno genera.

CAPITULO III.

Descripción del sistema desarrollado.

3.1 Descripción general de la familia TMS320C3x.

La familia del procesador digital de señales (DSP) TMS320C3x esta compuesta por procesadores de punto flotante de 32 bits. El bus interno de esta familia y las instrucciones especiales presentes en este procesador lo hacen muy veloz y flexible y puede ejecutar hasta 50 millones de instrucciones por segundo (MFLOPS). La familia de procesadores de señales TMS320 optimiza la velocidad al implementar funciones en hardware que otros procesadores implementan a través de software [11]. (Fig. 3.1)

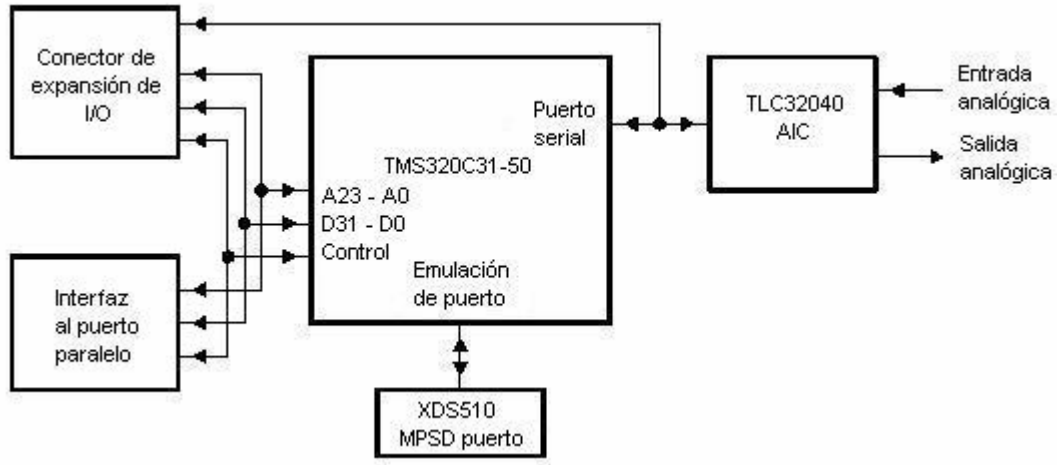


Figura 3.1 Diagrama a bloques de la familia DSP TMS320C3x

Algunas de sus principales características son:

- Bus de datos de 32 bits y Bus de direcciones de 24 bits.
- 8 registros de propósito general R0-R7 (32 bits c/u).
- 8 registros aritméticos auxiliares AR0-AR7 (32 bits c/u).
- 2 registros auxiliares aritméticos(32 bits c/u).
- 2 bloques de memoria RAM de 1kb cada uno por 32 bits.
- ALU de 40 a 32 bits, para representar números enteros o de punto flotante.
- Un chip de DMA.
- Operaciones lógico aritmético enteras o de punto flotante.
- Dos y tres operaciones por instrucción.
- ALU paralela entre el bus de datos y direcciones.
- Bloque con capacidad para repetir.
- Un puerto serie de 8/16/24/32 bits.
- Dos Timers de 32 bits.
- 50 MIPS.
- CLK de 50 Mhz.
- Un circuito analógico de interfaz (AIC), el TLC32040.
- Una interfaz al puerto paralelo.
- 2 conectores RCA, una entrada y una salida analógica en la tarjeta [10].

El CPU de la familia TMS320C3x consta de los siguientes componentes (Fig. 3.2):

- Multiplicador de entero / punto flotante.
- Unidad Lógica-Aritmética para operaciones de punto flotante, enteros y operaciones lógicas aritméticas.
- Registro de 32 bits.
- Buses internos de direcciones y datos.
- Archivo de Registro del CPU.
- Registro aritmético auxiliar [10].

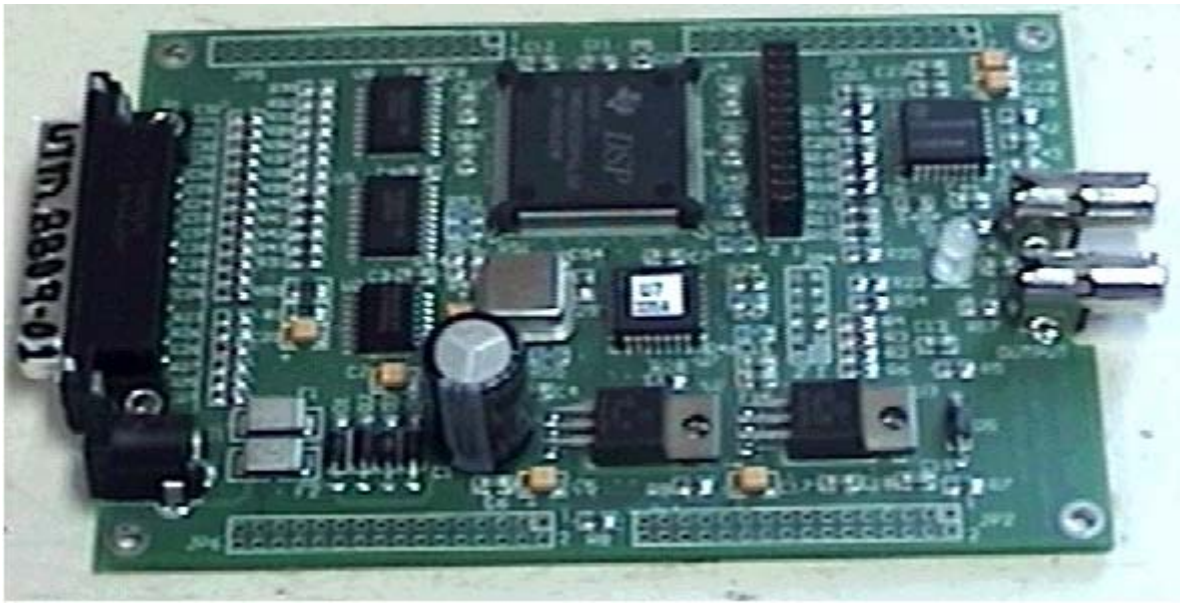


Figura 3.2 Tarjeta TMS320C31 Starter Kit de TI.

3.2 Mapa de memoria del DSP TMS320C31.

La configuración del pin MCBL / MP del TMS320C31 determina el modo en el cuál TMS320C31 puede funcionar:

- Modo de microprocesador (MCBL / MP = 0), o
- Modo de microcomputador / modo de cargador (MCBL / MP = 1)

La diferencia entre estos dos modos es su mapa de memoria. El espacio total de memoria de la familia TMS320C3x es de 16M (millón) con una palabra de 32 bits, programas, datos y espacios de entrada / salida están contenidos dentro de este espacio de direcciones de 16M [10]. (Fig. 3.3)

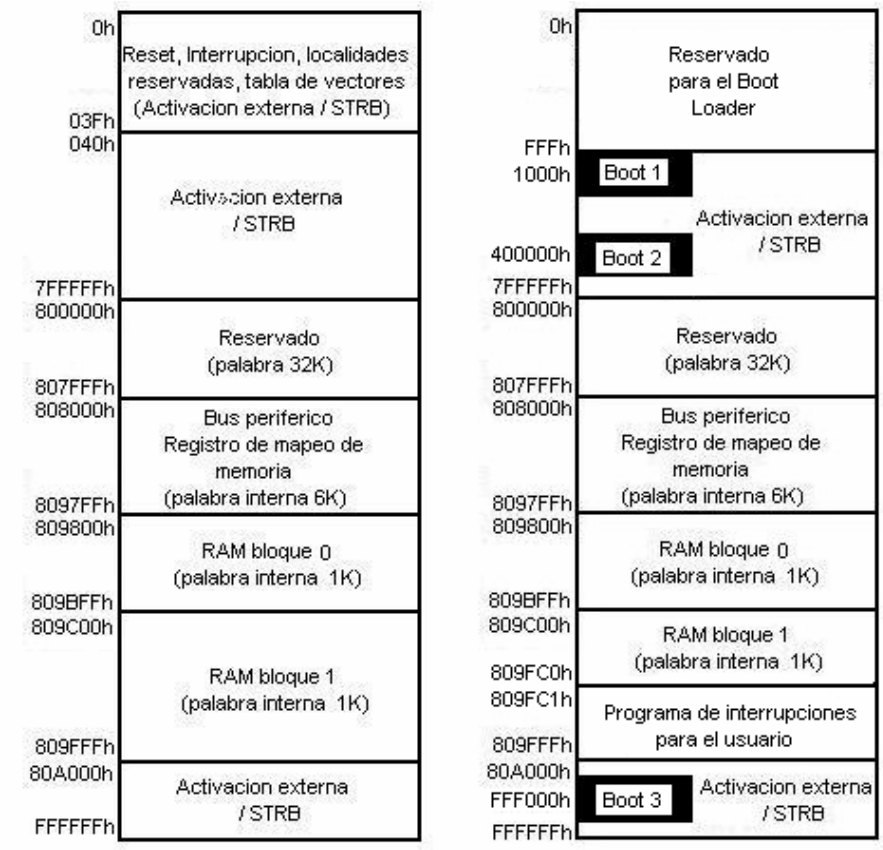


Figura 3.3 Mapa de memoria del procesador TMS320C31

Los componentes básicos de la tarjeta DSP TMS320C3X Starter Kit son: Un CODEC TLC32040, una PAL22V10, un sistema de reloj, una interfaz al puerto paralelo, un LED de tres colores, los conectores para el puerto paralelo de la DSK al host de la PC y el programa para la comunicación. Conectores de expansión que incluyen headers de 32 pines, y cuenta con 2 conectores RCA para conectar una entrada y una salida analógica en la tarjeta [12]. (Fig. 3.4)

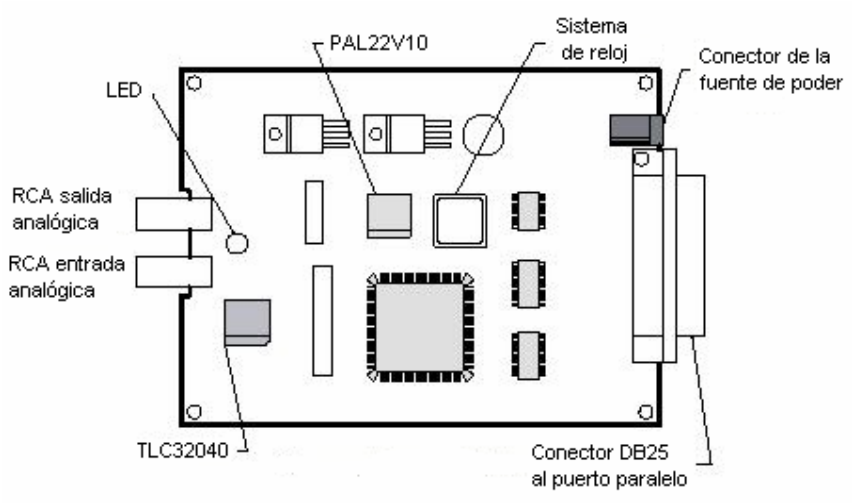


Figura 3.4 Componentes de la tarjeta TMS320C31 Starter Kit

3.3 Requerimientos de hardware.

Host: PC compatible IBM PC/AT, disco duro, drive de 3 ½ y un puerto de impresora, para conectar la tarjeta.
Memoria : espacio requerido por el programa 640 Kb como mínimo.
Display : monitor a color o monocromático, se recomienda un monitor de color.
Fuente: una fuente de alimentación en el rango de 12 a 7 Volts de entrada en DC, que soporte de 400 a 1500 mA.
Cable: un cable para el puerto paralelo DB25 [12].

3.4 Requerimientos de software.

Sistema Operativo: MS-DOS o PC-DOS (Ver. 5.0 o posteriores), Windows o OS/2
Archivos: dsk3a.exe es un archivo ejecutable para el ensamblador DSK. Al ejecutarse este archivo, genera todos los demás archivos necesarios que utiliza el DSK, dsk3d.exe: es un archivo ejecutable necesario para correr la interfaz del debuger del DSK [12].

3.5 Diagrama General a bloques del electrocardiógrafo.

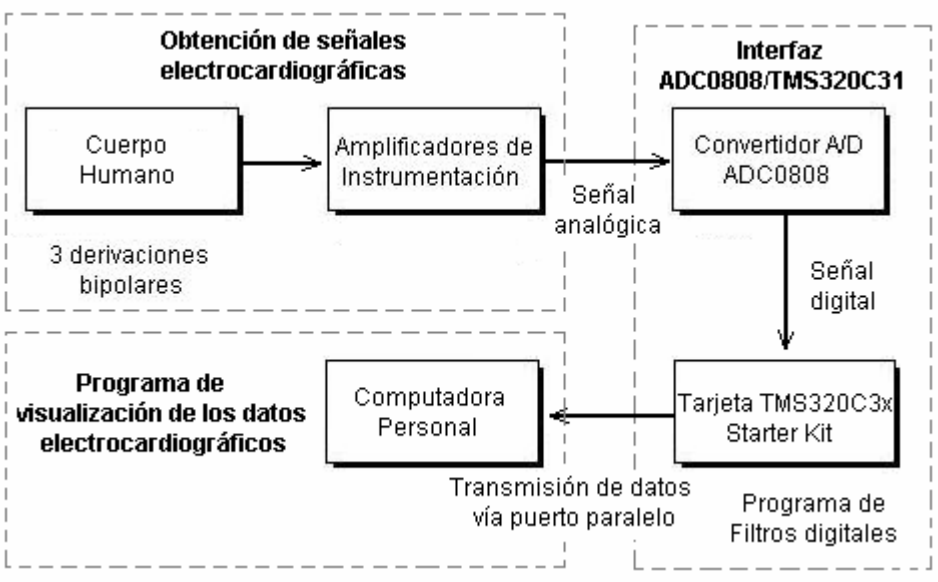


Figura 3.5 Diagrama general a bloques del electrocardiógrafo

La figura 3.5 muestra el diagrama general a bloques del electrocardiógrafo: la obtención de las 3 señales analógicas (3 derivaciones bipolares) por medio de los amplificadores de instrumentación; la conversión de señal analógica a señal digital por medio del convertidor ADC0808 y su respectivo procesamiento digital a través de los filtros digitales implementados en el procesador TMS320C31; la transmisión de los datos procesados digitalmente vía puerto paralelo hacia la PC y por último la visualización de los datos en el monitor de la PC por medio de un programa realizado en Borland C. Cada uno de estos bloques será descrito a detalle en las siguientes secciones.

3.6 Interfazando el convertidor ADC0808 al procesador TMS320C31.

La tarjeta TMS320C3x Starter Kit tiene un CODEC (convertidor A/D y D/A) interno, con una entrada analógica. Como se requiere de al menos 3 entradas

analógicas, al ser 3 derivaciones bipolares que serán sensadas, no podrá ser utilizado el CODEC, en su lugar se utilizará un dispositivo externo, el ADC0808, que es un convertidor analógico-digital de 8 bits de resolución para adquisición de datos de tecnología CMOS con un multiplexor de 8 entradas que pueden ser almacenadas. Los 8 canales multiplexados pueden ser accesados directamente por 8 entradas analógicas, siendo su rango de entrada de 0V a 5V, con una alimentación de 5 Volts y hasta con una entrada de 5V en DC. Se puede interfazar de manera sencilla a cualquier microprocesador, sus entradas son almacenadas y también sus direcciones, sus salidas tienen un tercer estado de alta impedancia. Tienen bajo consumo de potencia 15 mW y el tiempo de conversión del ADC0808 es de 100 μs., al tener la señal electrocardiográfica una frecuencia de operación de 1 a 3 Hz., el tiempo de conversión es suficiente y no habrá problemas de pérdida de información [13].

El ADC0808 es el encargado de digitalizar la señal analógica de entrada que será procesada en la DSP TMS320C31 y que proviene de los amplificadores de instrumentación, por lo que será necesario interconectarlo a la tarjeta TMS320C31 Starter Kit. Todo dispositivo externo que vaya a ser interfazado al DSP TMS320C31 será a través del bus primario y de sus respectivas líneas de control del procesador. A continuación hacemos una pequeña descripción de las líneas de control del DSP TMS320C31 que serán utilizadas en la interfaz ADC0808/DSP TMS320C31:

/STRB: Señal estroboscópica que indica al microprocesador cuando esta listo para poder enviar o recibir información a través de su bus de datos, ya sea de memoria o de algún dispositivo periférico. Su estado inicial es un nivel alto y cuando se activa se pone a un nivel bajo.

R/W: Señal que indica una escritura o una lectura de memoria o de algún otro dispositivo periférico que este interconectado al sistema. Cuando se hace una lectura, la señal en este pin se pone a un nivel alto, y al realizar una escritura se pone a un nivel bajo. Estas señales están en un solo pin del microcontrolador.

/RDY: Señal que controla el acceso del bus primario, se activa en nivel bajo.
D0..D31: Bus de datos, el procesador TMS320C31 puede leer 8, 16 o 32 bits del bus de datos.
A0..A32: Bus de direcciones de 24 bits, utilizado para la decodificación del ADC0808 y para la selección de la señal de entrada que realiza el ADC0808.

Las líneas de control del ADC0808, que serán utilizadas para que haya una correcta sincronía en la lectura/escritura de datos con el DSP TMS320C3x son: *START, ALE, OE, A, B, C*.

START: Se activa en nivel alto y le indica al ADC0808, que puede iniciar la conversión de la señal analógica, que previamente halla elegido el *ALE*.
ALE: Almacena la dirección del multiplexor (que señal de entrada de las 8 disponibles será muestreada).
OE : Los datos convertidos por el ADC0808 ya se encuentran disponibles y podrán ser leídos por el procesador.
A, B, C: Del menos significativo al más significativo, selecciona el canal de entrada analógica, que será muestrada, con estas tres líneas tiene hasta 8 posibles combinaciones de 0 0 0, hasta 1 1 1.

Conociendo las líneas de control de ambos dispositivos que serán empleadas, se va a determinar como interconectarlas para que puedan funcionar sin ningún problema, tomando como base el diagrama de tiempos del ADC0808 y del DSP TMS320C31, verificando también, en que nivel se activan las señales de cada dispositivo, nivel bajo o en un nivel alto para poder acoplarlos. (Fig. 3.6) y (Fig. 3.7)

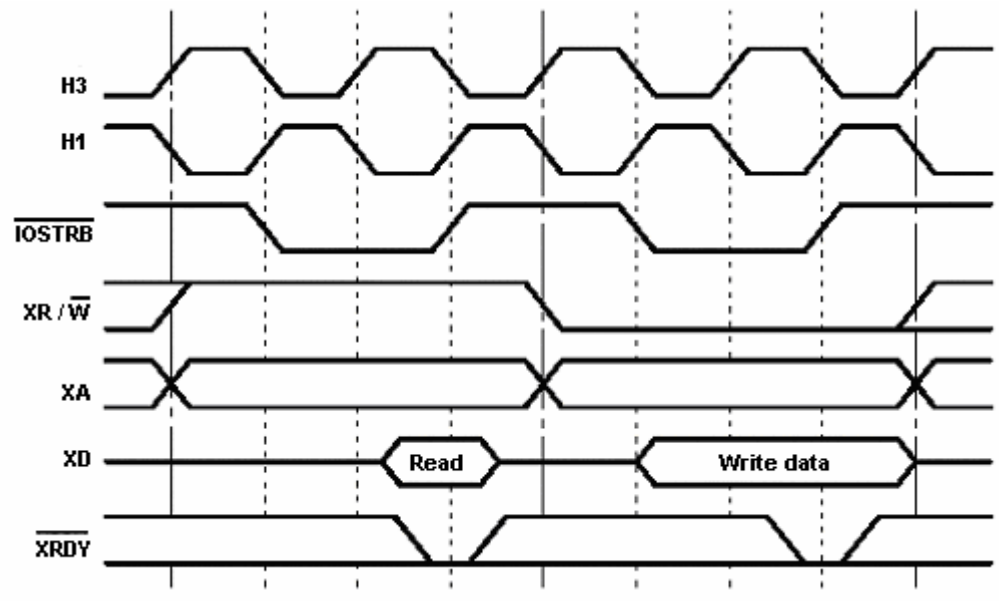


Figura 3.6 Diagrama de tiempos de lectura-escritura para el DSP TMS320C31

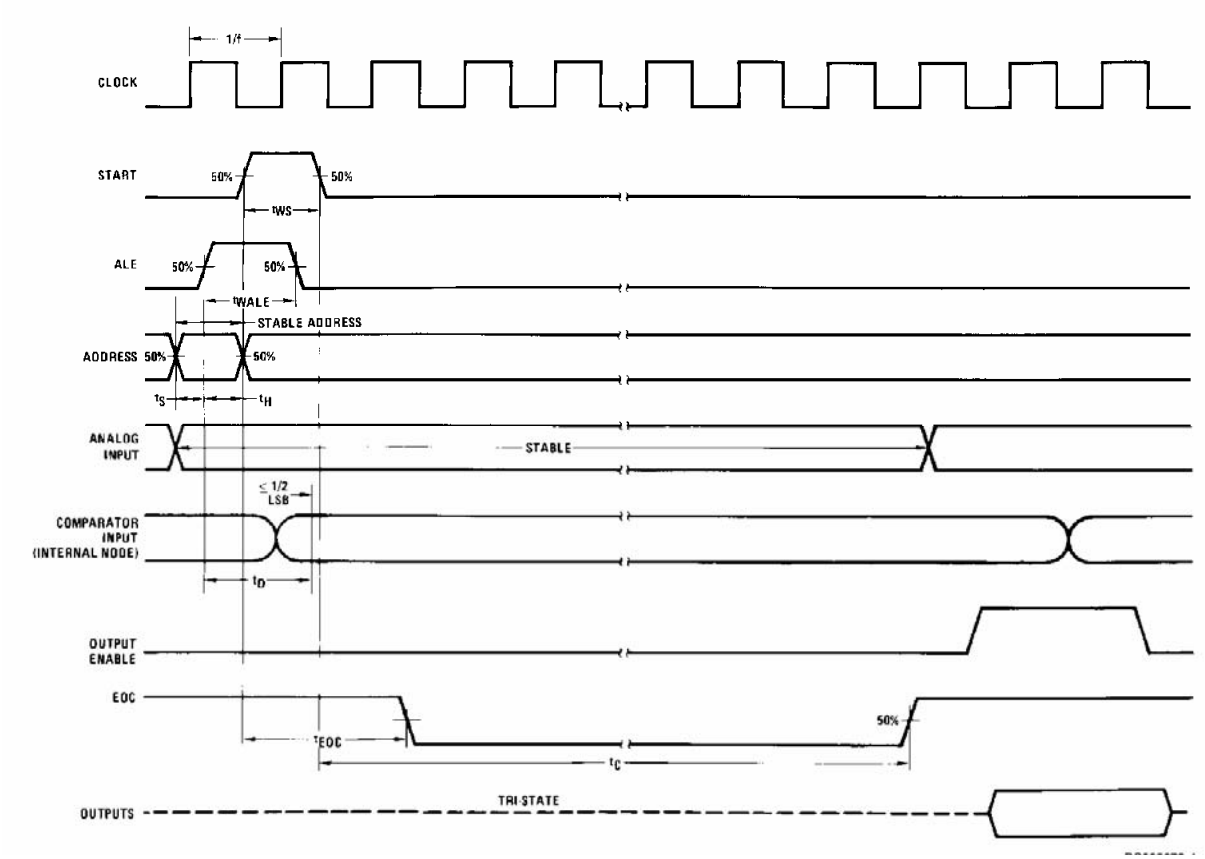


Figura 3.7 Diagrama de tiempos del ADC0808

Para poder interconectar un dispositivo externo al DSP TMS320C31 es necesario ocupar la señal de */STRB*, pues le indica al procesador cuando esta listo para poder enviar o recibir información a través de su bus de datos. Esta señal se encuentra activa en el rango de 80A000 a FFFFFFh. Y el mapa de memoria donde se encuentra mapeada la PC (vía puerto paralelo) es de FFF000h a FFFFFFh, como se puede observar en la figura 3.8, esta se encuentra dentro del rango de la señal */STRB*. Esto causa un problema, si se escribe al puerto paralelo los datos ya procesados esta escritura activara la señal de */STRB* y le dirá al ADC0808 que inicie una nueva conversión, cuando esta solo debería activarse al presentarse una lectura o una escritura al ADC0808, lo que ocasionará una perdida de información.

Figura 3.8 Rango de memoria de la señal /STRB

Este problema se puede solucionar con un decodificador 74LS138, al que se conectan las líneas de dirección : *A12-A15* y las señales de *R/W* y */STRB* del DSP TMS320C31, como se indica en la figura 3.9, con esto se asegura que al realizar una lectura o una escritura al ADC0808 se activen las conversiones y no cuando se lea o escriba al puerto paralelo.

Como el *ALE* almacena la combinación de la señal de entrada a ser muestreada y el *START* es el inicio de conversión para el ADC0808, pueden ir unidos, y serán activados cuando el DSP TMS320C31 haga una lectura-escritura al ADC0808, como se observa en la figura 3.10.

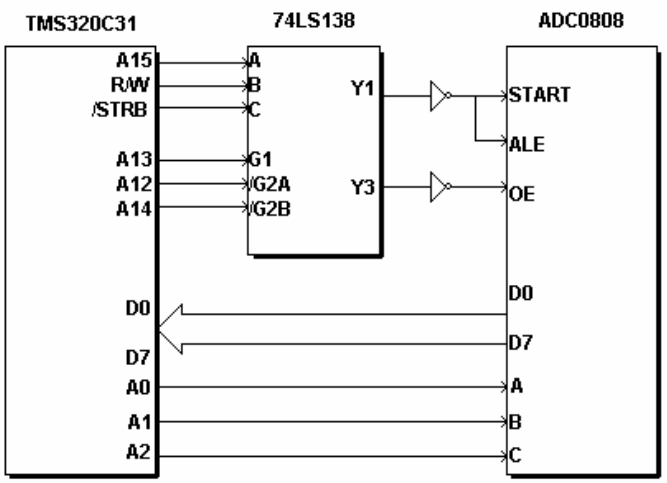


Figura 3.9 Diagrama a bloques de la interfaz ADC0808 – DSP TMS320C31

A23	A22	A21	A20	A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Direcciones
1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	FFF000 h
1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	80A000 h

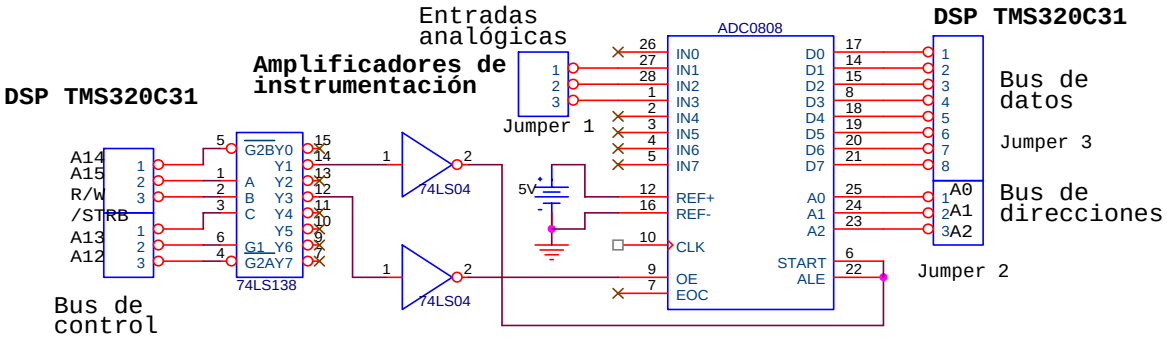


Figura 3.10 Líneas de direcciones y control conectadas al decodificador 74LS138

Cuando las líneas *A12*, *A14*, */STRB*, *R/W* están a un nivel bajo y las líneas *A13*, *A15* están a un nivel alto, llegan a coincidir con las líneas de entrada del decodificador 74LS138, activando el pin *Y1* del decodificador a un nivel bajo, habilitando al inversor que lo pone a un nivel alto que es cuando se activan las

señales de *ALE* y *START* y sucede una escritura, es decir el DSP TMS320C31 le indica al convertidor ADC0808 que almacene la dirección *ALE*, dada por *A, B, C* y que comience *START* a muestrear la señal que se encuentra en dicha dirección. Después de que las señales *ALE* y *START* fueron activadas, se espera 100 us que es el tiempo que necesita el convertidor ADC0808 para tener los datos digitales y puedan ser leídos por el DSP TMS320C31. Cuando las líneas *A12*, *A14*, */STRB* están a un nivel bajo y las líneas *A13*, *A15*, *R/W* coinciden con las líneas de entrada del decodificador, esta vez activando el pin de salida *Y3* a un nivel bajo, pasa por un inversor y se conecta al pin *OE* del ADC0808 el procesador TMS320C31 puede leer los datos que tenga disponible el ADC0808 por medio del bus de direcciones que ambos comparten. Los pines de salida del ADC0808, es decir el bus de datos, está en un tercer estado de alta impedancia, hasta que sean leídos los datos.

Las direcciones menos significativas del DSP TMS320C31 *A0*, *A1* y *A2* van conectadas a las líneas del ADC0808 *A*, *B*, y *C* respectivamente, que son las encargadas de la multiplexación de la señal de entrada, con la combinación de estas líneas de control se determina que derivación de la señal será leída (muestreada). (Fig. 3.10)

Como se observa al conectar las líneas de direcciones *A12-A15, /STRB* y *R/W* al decodificador 74LS138 se asegura que cuando se escriba o lea se accese a la dirección en que se encuentra mapeado el convertidor ADC0808. El tiempo de lectura de la TMS320C31 es de de 40 ns., como el tiempo es pequeño en comparación al tiempo de acceso del ADC0808 es necesario configurar estados de espera por software en el DSP TMS320C31 para poder accesar al dispositivo, fueron necesarios 4 estados de espera cada uno de 40 ns. Dando un total de 160 ns., esta configuración se realiza en el registro de control global del bus primario con la palabra 0x998.

El jumper 1 de la figura 3.10, sirve para meter las tres señales analógicas que nos proporcionan los amplificadores de instrumentación y que vienen directamente de las derivaciones bipolares de los electrodos.

En el jumper 3 están los datos disponibles que entrega el ADC0808 una vez hecha la conversión y esta conectado directamente al bus de datos del registro primario del DSP TMS320C31. Como el bus de datos del DSP TMS320C31 es de 32 bits y el ADC0808 es de 8 bits, se ocupan los bits menos significativos del DSP TMS320C31, es decir de *D0...D7*. En el jumper 2 de la figura 3.10 están las direcciones que se encargan de elegir que señal será muestreada, estarán conectadas con las direcciones menos significativas de *A0*, *A1*, *A2* de la DSP TMS320C31 para que puedan multiplexar la señal. El ADC0808 necesita un reloj que opere en el rango de frecuencias de 500 a 640 KHz. Que sea una señal de reloj estable, esto se logra con un cristal de 4 MHz y un contador 74LS161A que sirve como un divisor de frecuencias. (Fig. 3.11)

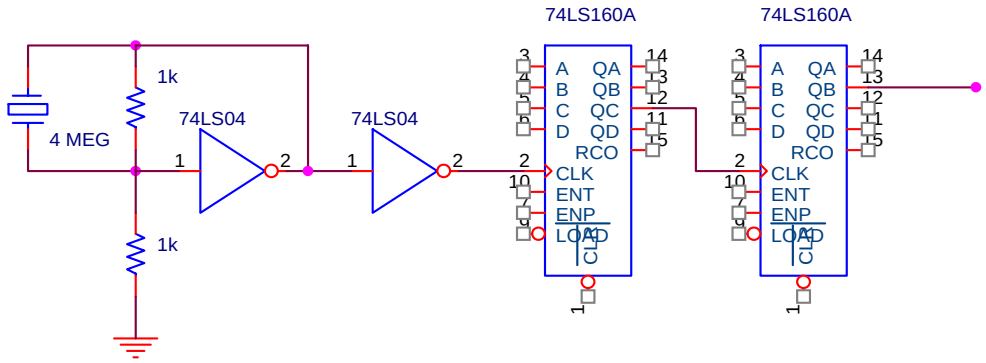


Figura 3.11 Reloj de cristal 500 KHz. utilizando contador de frecuencias 74LS161A

Como se puede observar, los inversores junto con las resistencias hacen la función de un oscilador por medio de un cristal de 4 MHz, esta señal entra a un contador de frecuencias, que funciona como un divisor de frecuencias, la salida del primer contador de frecuencias es de 2 MHZ (las salidas disponibles son: QA es 4MHz/2, QB 4MHz/4, QC 4MHz/8 y QD 4MHZ/16), esta se mete a otro contador de frecuencias para reducir aún mas la frecuencia de salida, va

conectada a la entrada CLK del contador de frecuencias No. 2, se utilizará la salida QC del contador de frecuencias No. 2, pues esta proporciona la frecuencia de 500 KHz, que es la que necesita el reloj del ADC0808.

El 74LS161A es un contador síncrono cuya entrada máxima de reloj puede ser de hasta 32 MHz, tiene cuatro salidas cada una de ellas del bit menos significativo al más significativo divide la señal de reloj de entrada en 2, 4, 8 y 16, obteniendo frecuencias de salida para una señal de entrada de 4 MHz de: 2 MHz, 1 MHz, 500 KHz y de 250 KHz., la división entre 8 es la señal de reloj de interés, presenta niveles de alimentación y de salida TTL.

Se escogió un cristal por presentar una gran estabilidad en la señal y por utilizar otros dispositivos compatibles con TTL tanto de alimentación, como en la señal de salida. Todos estos dispositivos desde el ADC0808 hasta el contador, son fácilmente encontrados al ser de uso muy comercial, y el precio de cada uno de ellos es muy accesible, así como resultan fácil de interconectar a cualquier dispositivo. (Fig. 3.12)

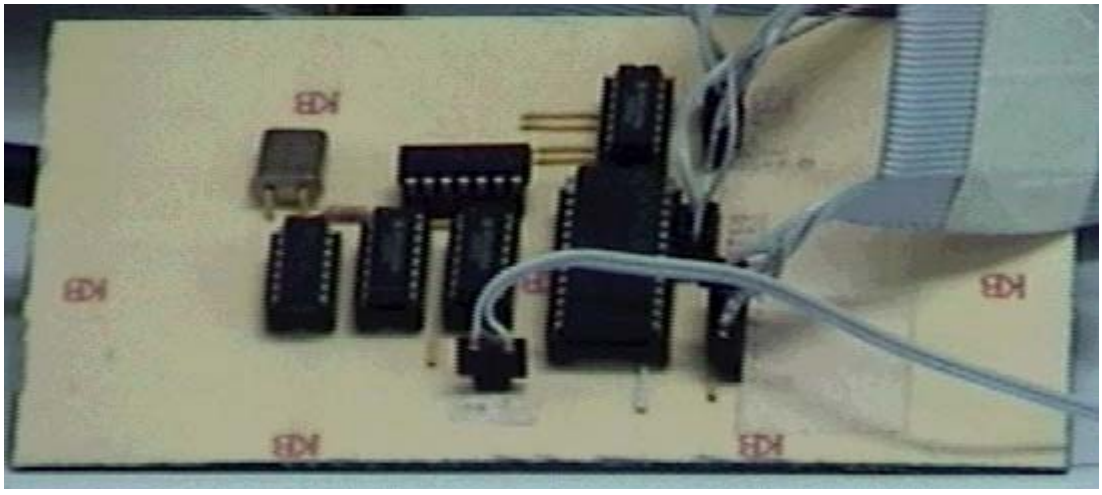


Figura 3.12 Circuito impreso del convertidor ADC0808

3.7 Diseño de los Amplificadores de instrumentación.

Uno de los amplificadores más útiles para la medición, instrumentación o control es el amplificador de instrumentación. Está diseñado con varios amplificadores operacionales y resistencias de precisión, que hacen al circuito muy estable.

Amplificador diferencial básico: El amplificador diferencial puede medir y también amplificar pequeñas señales que quedan ocultas en señales más intensas, el voltaje de salida esta dado por: (Fig. 3.13)

$$V_0 = mE_1 - mE_2 = m(E_1 - E_2)$$

El voltaje de salida del amplificador diferencial, V_0 es proporcional a la diferencia de voltajes aplicada a las entradas *(+)* y *(-)*. El multiplicador m se denomina ganancia diferencial y se establece por la relación entre resistencias.

$$m = \frac{mR}{R}$$

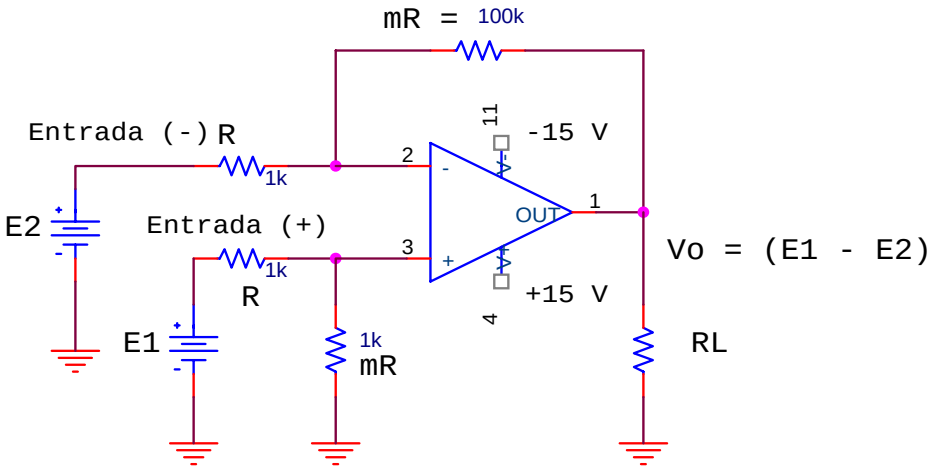


Figura 3.13 Amplificador diferencial básico

Voltaje en modo común: La salida del amplificador diferencial debe ser 0 cuando $E_1 = E_2$. Para este caso el voltaje de entrada se denomina voltaje de entrada en modo común E_{CM}. V_0 será *0* si las relaciones de resistencias son iguales. Estas relaciones de resistencias se igualan mediante la instalación de un potenciómetro en serie con una resistencia, dicho potenciómetro se afina hasta que V_0 se reduce a un valor despreciable. Esto causa que la ganancia de voltaje en modo común V_0 / E_{CM} se aproxime a *0*. Esta es la característica de un amplificador diferencial que permite que una señal débil se capte extrayéndola de una señal de ruido mucho más intensa [15]. (Fig. 3.14)

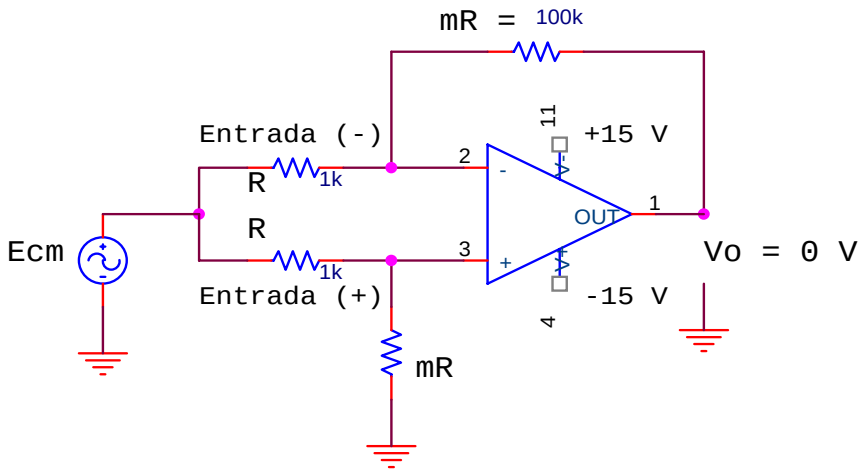


Figura 3.14 La ganancia de voltaje en modo común es cero

El amplificador diferencial básico tiene dos desventajas, tiene baja resistencia de entrada y el cambio de ganancia es complicado, 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 amplificadores operacionales en configuración de seguidor, la salida del amplificador diferencial A_1 con respecto a tierra es E_1, y la salida del amplificador operacional A_2 con respecto a tierra es E_2. El voltaje diferencial de salida V_0 es igual a la diferencia entre E_1 y E_2. Este tipo de amplificador tiene salida diferencial, ningún extremo de salida está conectado a tierra.

La segunda desventaja del amplificador diferencial básico es la falta de ganancia ajustable. Este problema se elimina al agregar tres resistencias al amplificador aislador resultando un amplificador de entrada diferencial y salida diferencial con ganancia ajustable, la alta resistencia de entrada se mantiene con los seguidores de voltaje. Para cambiar la ganancia del amplificador sólo se tiene que ajustar una resistencia única *aR*, pudiéndose conectar únicamente a cargas flotantes (no tienen ninguna terminal conectada a tierra). Para manejar cargas a tierra debe agregarse un circuito que convierta el voltaje diferencial de entrada en un voltaje con referencia a tierra, esta configuración resultante es denominado amplificador de instrumentación [15]. (Fig. 3.15)

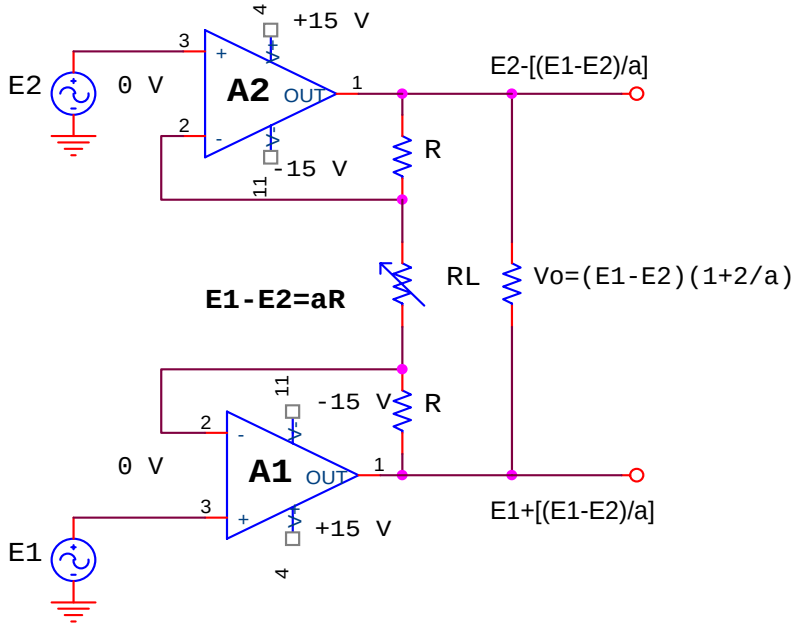


Figura 3.15 Mejoras al amplificador diferencial básico

Amplificador de instrumentación: Está hecho de tres amplificadores operacionales (TL084) y siete resistencias. Este amplificador se hace conectando un amplificador aislado a un amplificador diferencial básico, el amplificador A_3 y sus cuatro resistencias iguales *R* forman un amplificador diferencial con una ganancia unitaria. Sólo las resistencias de A_3 tienen que igualarse. La resistencia

R', puede hacerse variable para eliminar cualquier voltaje en modo común. Solo una resistencia *aR*, se usa para establecer la ganancia de acuerdo a la ecuación:

$$\frac{V_0}{E_1 - E_2} = 1 + \frac{2}{a}$$

donde $a = aR/R$

E_1 se aplica a la entrada *(+)* y E_2 a la entrada *(-)*. V_0 es proporcional a la diferencia entre los voltajes de entrada [15]. En resumen podemos destacar las características del amplificador de instrumentación:

1. La ganancia de voltaje, desde la entrada diferencial *($E_1 - E_2$)* a la salida 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_0 no depende del voltaje común a E_1 y E_2 (Voltaje en modo común), sólo en su diferencia.
4. Los capacitores C1 y C2 presentes, eliminan el nivel de CD. que pudieran presentar los seguidores. (Fig. 3.16) y (Fig. 3.17)

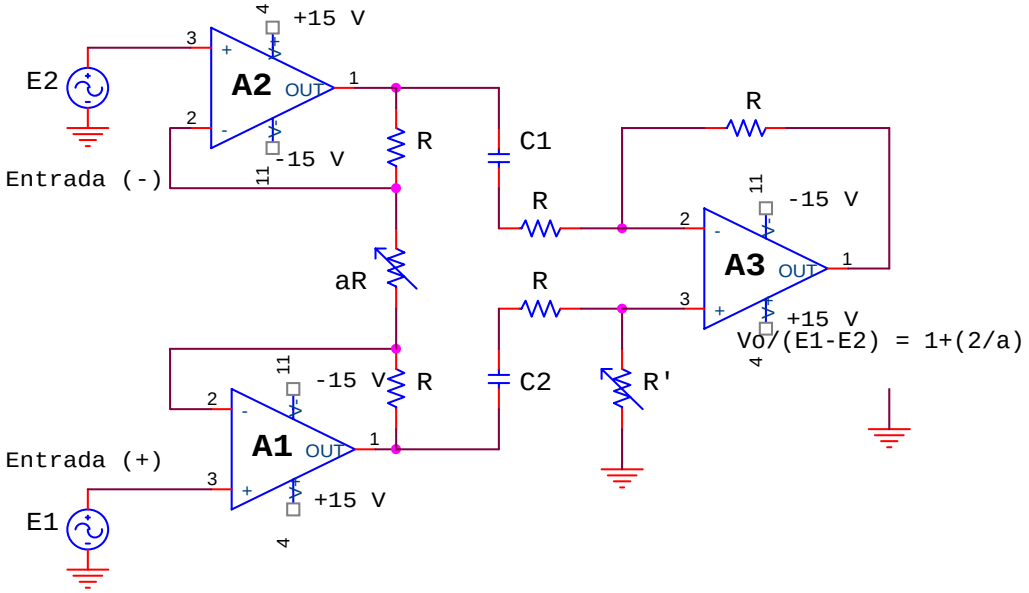


Figura 3.16 Amplificador de instrumentación

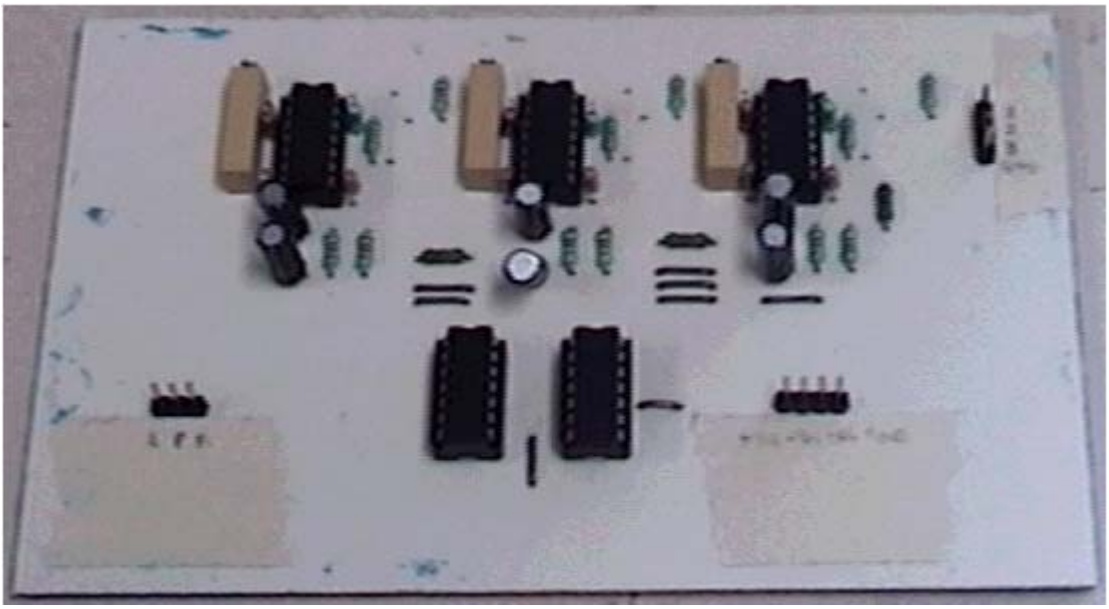


Figura 3.17 Circuito impreso del amplificador de instrumentación

3.8 Circuito para el voltaje referencial de salida.

En algunas aplicaciones es deseable desviar el voltaje de salida a un nivel de referencia diferente de 0V(positivo o negativo), esto puede realizarse con bastante facilidad agregando un pequeño circuito que a continuación describimos:

La señal que entregan los amplificadores de instrumentación tienen un nivel de referencia positivo, debido a que el ADC0808 en sus entradas analógicas no puede tomar valores negativos, de ser así estaría recortando la señal, aceptando únicamente valores mayores a 0V. El circuito muestra un amplificador operacional con un nivel de offset en la entrada no inversora, con un factor de ganancia de 2, debido a que las resistencias presentan los mismos valores [14], este circuito de voltaje referencial fue realizado con un TL081. (Fig. 3.18)

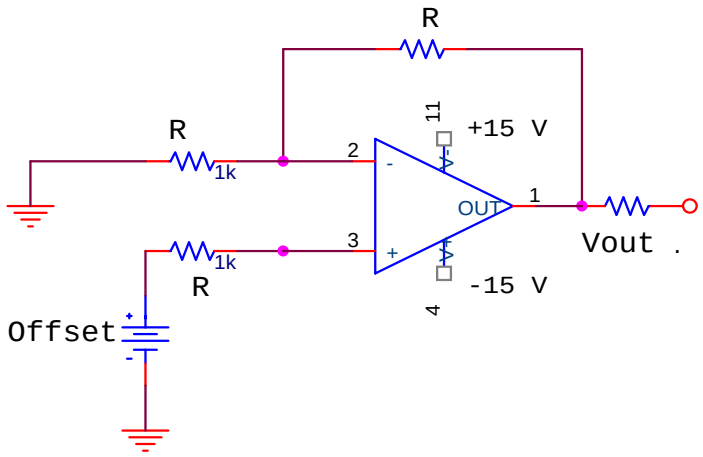


Figura 3.18 Circuito que presenta un nivel de offset a la salida

3.9 Descripción del software realizado.

Fir.asm: este archivo es un programa para el filtrado digital de señales que viene integrado con el software que proporciona la tarjeta TMS320C3x Starter Kit, lo que se necesita son los coeficientes del filtro, estos coeficientes determinan el tipo de filtro (pasa bajas, pasa altas, pasa banda, etc.), estos coeficientes fueron obtenidos con el programa Matlab utilizando la función del filtro *Remez* [9], que permite diseñar filtros digitales tipo FIR de fase lineal ó IIR de fase no lineal, entre más grande sea el orden del filtro mejor será su respuesta en frecuencia y presentará menos rizado en la banda de paso y en la banda de rechazo, y la caída en la banda de transición es más abrupta, sin embargo esto aumenta el tiempo de cómputo para el filtrado digital pues genera mas coeficientes del filtro y por ende más operaciones de suma de productos. Una vez obtenidos los coeficientes del filtro estos van a ser probados en el programa *fir.asm*, los coeficientes son para un filtro digital FIR pasa bajas que corta en 50 Hz y es de orden 14, los coeficientes generados son de 15 números de punto flotante. Por medio del generador de señales conectado a la entrada analógica RCA de la tarjeta TMS320C3x Starter Kit y el osciloscopio conectado a la salida analógica RCA, se puede determinar si efectivamente el filtro digital cumple con las características deseadas, se introduce una señal de prueba senoidal con frecuencia de 1Hz., con un voltaje

pico-pico (Vpp) de 1.5V, se va incrementando la frecuencia de la señal senoidal a través del generador de señales hasta llegar a los 100 Hz.. Esta señal debe empezar a disminuir su amplitud a medida que se acerca a la frecuencia de 50 Hz., de tal forma que en esta frecuencia su amplitud debe estar atenuada –3 dB abajo de la señal original de entrada, es decir, si la amplitud de la señal de entrada a 1 HZ., es de 1.5 Vpp, la señal de salida en 50 Hz., que es donde corta el filtro digital debe tener una amplitud Vp de 1.06 V.

Si presentara alguna deficiencia se recurrirá nuevamente al Matlab para hacer un nuevo diseño, se debe hacer notar que entre más grande sea el orden del filtro, más tiempo de computo necesitará, por este motivo el filtro será de un orden suficiente como para eliminar el ruido de manera efectiva en la señal sin ocupar demasiado tiempo de cómputo. Como se ha mencionado en secciones anteriores es importante notar que los coeficientes del filtro digital deben provenir de una función de filtro FIR, pues este tipo no distorsiona la señal que se está presentando por ser de fase lineal, esto se debe a que el retardo temporal que experimentan los componentes de frecuencia de la señal de entrada a través del sistema, es aproximadamente cero, de lo contrario se corre el riesgo de que la señal adquirida por los amplificadores de instrumentación a través de los electrodos no sea la correcta. Una vez obtenidos los coeficientes adecuados y de haber comprobado con una señal de prueba que el filtro funciona correctamente, es decir, empieza la atenuación de –3 dB/década a partir de los 50 Hz, se puede concluir que los coeficientes son los correctos y se pueden utilizar para futuras pruebas. Durante el desarrollo de los programas basados en el set de instrucciones del DSP TMS320C31, se fueron realizando programas pequeños tratando que la complejidad de los mismos sea mínima debido a la poca experiencia que en programación de procesadores de la familia de TI se tenía. La figura 3.19 muestra el diagrama a bloques de la secuencia general que se tuvo en la elaboración de los programas para el filtrado digital de señal que se pretendía. A continuación se describe el primer programa realizado que fue como obtener datos del ADC0808.

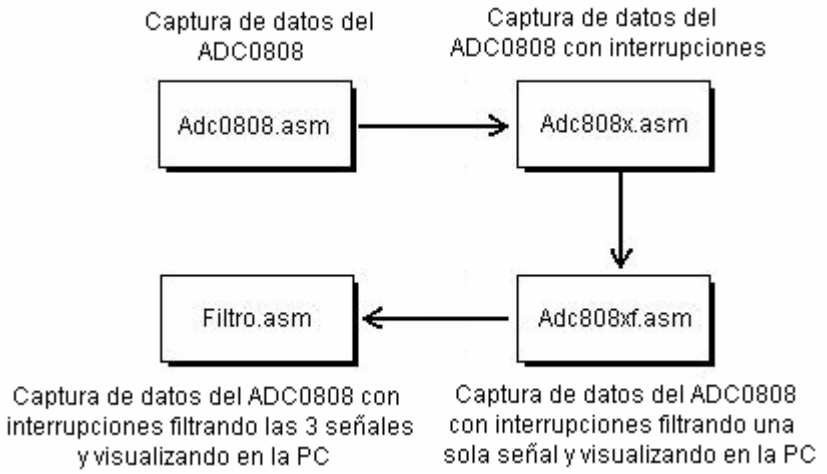


Figura 3.19 Diagrama a bloques de los pasos en la realización del software

Adc0808.asm: este programa permite obtener datos digitales del convertidor ADC0808, lo primero que se hace es limpiar los registros a ser utilizados, después se configura el bus primario, esta configuración es necesaria pues el ADC0808, es un dispositivo externo y lento a ser interconectado con respecto a la tarjeta TMS320C3x Starter Kit. Por medio del bus primario se configura cuantos estados de espera necesita el ADC0808, para poder ser accesado y de que modo son los estados de espera, se escogieron 4 estados de espera por software de 40ns cada uno, de no añadir los estados de espera, el DSP TMS320C31 no podría acceder al ADC0808. La dirección del puerto paralelo, así como el dato a ser capturado son direccionados a los registros auxiliares *AR0..ARn*, para poder tener acceso a las localidades de memoria y localidades donde se encuentra mapeada la PC, cualquier otro registro diferente a los auxiliares no tiene acceso a ello. El siguiente código de bloque es un claro ejemplo.

```
sti  R0,@RCG          ; RCG-->0,Reset al Bus Primario
ldi  @RCGVAL,R1       ; R1-->(RCGVAL-->0x09B8)
sti  R1,@RCG          ; RCG-->0x09B8, Config el Bus, 135 ns->ADC0804
ldi  1h,R1            ; se limpia R1
ldi  @DATO,AR0        ; para poder accesar a un disp. periférico
```

```
                                ; se utilizan los registros auxiliares

conv    sti  R1,*AR0        ; WR manda tms orden de conversion
        call espera         ; espera que int convierta (100uS)
        ldi  *AR0,R2        ; RD, TMS lee la conversion
        br   conv

;--------- Mapa de memoria a utilizar --------

RCG       .set 0x808064         ;Reg de control del Bus Primario
RCGVAL    .word 0x00000998   ; Espera por Software con 4 estados
DATO      .word 0x0080A000   ; bloque donde se habilita la señal estroboscópica
```

Paso 1: Se manda un dato de inicio (cada dato de inicio corresponde a su respectiva derivación) al puerto paralelo por medio de los registros auxiliares previamente asignados, este dato de inicio será reconocido en el Programa *filtro.cpp* y determinará a que derivación corresponde, así será graficada la señal correspondiente; el programa en C sabrá que cada dato de inicio corresponde a una derivación diferente según sea el caso, el valor de inicio es incrementado después de cada lectura del DSP TMS320C31, este dato empieza con un valor de 0x51 derivación 1, (0x indica que es un número en hexadecimal) hasta 0x53 derivación 3, al llegar a esta última derivación este contador es inicializado a 0x51. Así se asegura que los datos serán graficados con su correspondiente derivación. Se debe hacer notar que el programa fue realizado a partir del diagrama de tiempos del ADC0808, en que nivel alto o bajo se activan las señales de *START* y *ALE* y en que momento esta lista la conversión para ser leída por el DSP TMS320C31 tomando en cuenta también las señales de control de éste, en que nivel se activan, nivel bajo o alto, pudiendo adaptar algunos inversores cuando las líneas de control de ambos dispositivos se activan en niveles diferentes.

Paso 2: Después de mandar el dato de inicio se escribe al lugar donde se encuentra mapeado el ADC0808, que se encuentra a partir de la dirección 0x80A000, con esta escritura se activan las señales de *START* y *ALE* del

ADC0808, el primero le indica que puede iniciar la conversión de la señal que ALE ha seleccionado y que será muestreada, en seguida se llama una función de retardo que espera 100 μS., tiempo en que el ADC0808 realiza una conversión, después de que la DSP TMS320C31 espera este tiempo, lee la dirección donde se encuentra mapeado el ADC0808 y obtiene el dato digital previamente codificado, mandándolo directamente por el puerto paralelo para que pueda ser leído con el programa *filtro.cpp*. La lectura de datos se hace en tres localidades diferentes, cada una de ellas corresponde a diferente derivación, se empieza a leer en la dirección 0x80A001(dato del derivador 1), hasta la dirección 0x80A003 (derivador 3), reinicializándose la dirección de lectura al llegar a este último. El paso 1 y el paso 2 están dentro de un ciclo para que la lectura de las señales sea indefinido, se tomarán tantas muestras como se deseen. El siguiente bloque de programa muestra lo explicado en el programa *Adc0808.asm.*

```
        ldi  @paralelo,AR1   ; direccion del puerto
ini     ldi  @DATO,AR0       ; accesar a un disp. periférico
        ldi  51h,R1          ; electrodo_1
conv
        sti  R1,*AR1         ; dat. inicio al Pto. paralelo
        sti  R0,*AR0         ; se activa ALE y START
        call espera          ; espera que int convierta (100uS)

        ldi  *AR0,R2         ; RD, TMS lee la conversion  se activa OE
        sti  R2,*AR1         ; se escribe al Pto. Paralelo
        addi 01h,AR0
        addi 01h,R1
        cmpi 54h,R1
        bne  conv
        br   ini
```

Adc808x.asm: una vez obtenido un programa base, es necesario determinar la frecuencia de muestreo y la frecuencia con la que serán transmitidos los datos adquiridos hacia el puerto paralelo, para ello es necesario implementar al programa anterior un nuevo programa con interrupciones del Timer 0. Las

interrupciones determinan la frecuencia de muestreo de la señal, al ser señales de baja frecuencia, la frecuencia de muestreo es relativamente baja, en promedio 380 muestras por segundo serán necesarias. Las interrupciones pueden realizarse de diferentes formas: internas o externas, una de ellas presente en el DSP TMS320C31 son las interrupciones internas, conocidas como timers. Tomando como base el programa anterior serán agregadas pequeñas líneas de programa para convertirlo en un programa con interrupciones.

En la cabecera del programa se le agrega la dirección del timer a utilizar, en este caso se trata del timer 0, ubicada en la dirección 0x809FC9. Son asignados a sus respectivas direcciones, tres registros que permiten configurar el timer: Registro de control global del timer que determina el modo de operación del timer, monitorea el estado del timer y controla la función de su pin de I/O; el Registro de periodo que especifica la frecuencia que tendrá la señal del timer y el Registro de contador, que contiene el valor del incremento y se incrementa su valor en cada flanco de subida o de bajada de la señal de reloj, el contador del timer es puesto a cero (reset) automáticamente, siempre que sea igual al registro de periodo. Una vez configurados hay un ciclo infinito (loop), que se encuentra ciclado, únicamente saltará al cuerpo del programa (captura de la muestra), cuando suceda la interrupción, el cuerpo del programa es similar al descrito anteriormente (*ADC0808.asm*), en los pasos 1 y 2 se agregan algunos registros en la cabecera del programa y se inicializan con un valor. El programa estará en un loop hasta que suceda la interrupción, al ocurrir saltará al cuerpo del programa descrito en (*ADC0808.asm*).

```
;--------- Mapa de memoria a utilizar --------
TGCR0   .set  0x808020        ; Timer 0 global control register
TCNT0   .set  0x808024        ; Timer 0 counter register
TPR0    .set  0x808028        ; Timer 0 period register
TIMERPER .set  55             ; 384 muestras Timer period reg 55->37dec
TIMVAL   .word 0x3c1          ; Timer global control register value

or   2000h,ST        ; Global interrupt enable
```

```
sti  R4,@TGCR0       ; Reset timer 0
ldi  TIMERPER,R7
sti  R7,@TPR0        ; Store timer 0 period
sti  R4,@TCNT0       ; Reset timer 0 counter
ldi  @TIMVAL,R7      ; Load timer control value
sti  R7,@TGCR0       ; Start timer 0
or   100h,IE         ; Enable RINT interrupts
loop
       br   loop
```

Adc808xf.asm: este programa toma como base el programa anterior, con algunas modificaciones, va adquirir únicamente una entrada, una sola derivación, después de la lectura del dato hará la llamada a la funcion *DAC1* que filtra la señal digital pasa bajas, una vez que nuestros resultados sean positivos se generaliza para las tres señales. Después de la cabecera del programa donde se define en que localidad empezará el programa y donde viene la dirección del Timer 0, es necesario definir un bloque de punto flotante con algún nombre que los identifique, donde serán almacenadas las muestras que se irán obteniendo, este bloque será del orden del filtro más uno, además se define el bloque donde se encuentran los coeficientes del filtro, como se menciona anteriormente, estos coeficientes son generados en el Matlab con la función *remez*, únicamente para filtros FIR. Estos dos bloques son de tipo flotante. Se le asigna el tamaño del bloque (ambos son iguales) a una variable, y se inicializan los apuntadores a las localidades de memoria donde se encuentra ubicados estos datos. El cuerpo es igual al programa adc808x.asm excepto que después de que el DSP TMS320C31 lea el dato del ADC0808, llama a la función DAC1 para filtrar la señal.

```
ADC_recv    .float    0.0       ; espacio para los datos de entrada que serán almacenados
                .float    0.0       ;
                .float    0.0       ;
FIR_coef    .float    0.0218    ; coeficientes del filtro
            .float   -0.0196
            .float   -0.0196
END_coef
```

Para visualizar como funciona el direccionamiento circular y por consiguiente el funcionamiento de un filtro digital se explica el siguiente bloque de programa:

```
0            stf   R3,*AR4++
1            ldi   @SIZE,BK
2            ldi   @SIZE,RC
3            rptb  FIR21
4           mpyf3 *AR0++(1)%,*AR2++(1)%,R0
5  FIR21 || addf3 R0,R2,R2
6           addf  R2,R0
```

Se comienza con el paso 0, donde se pone el dato muestreado por el ADC080 en el espacio reservado para las muestras de entrada *x[n]* y se incrementa para poner el siguiente dato, una vez que se llena el bloque este vuelve a empezar desde el inicio, al estar activado el registro *BK*, que corresponde al direccionamiento circular [10]. (Fig. 3.20a) y (Fig. 3.20b)

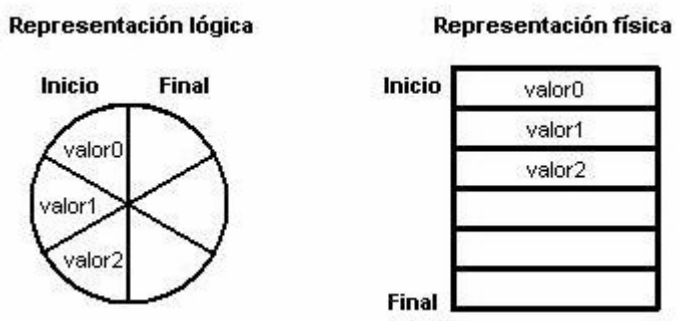


Figura 3.20a Implementación del direccionamiento circular

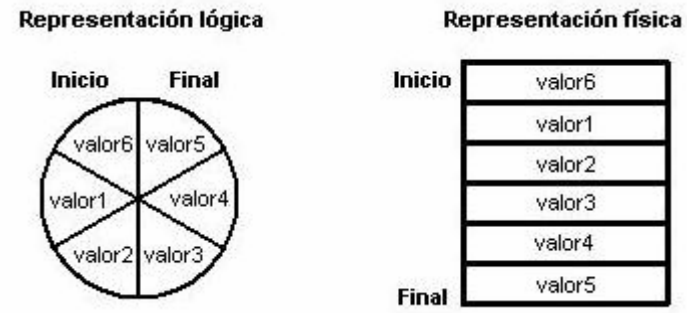


Figura 3.20b Almacenando datos de entrada x[n] a ser filtrados

En el paso 1 se le asigna el tamaño del bloque al registro *BK* (cuantas veces será repetido el bloque), mientras que el registro *RC* llevará el valor del decremento cada que se ejecuta el bloque *FIR21.* Este bloque básico representa la multiplicación de los coeficientes (paso 4) con los datos de entrada y su respectiva suma (paso 5), lo que nos dará el dato de salida. En el paso 6 se suman todo los productos acumulados y se obtiene el dato de salida ya filtrado.

Una vez que el programa *Adc0808xf.asm* es capaz de filtrar una señal digital, se generaliza para las tres derivaciones del electrocardiograma, es decir, a cada señal digital se le asigna un filtro digital del mismo orden, convirtiéndose en el programa *Filtro.asm* que adquiere los datos del convertidor ADC0808 los filtra y los envía por el puerto paralelo para ser visualizados en el monitor de la PC mediante el programa *Filtro.cpp*.

Filtro.cpp: Como hemos descrito anteriormente, el Electrocardiograma es visualizado en la computadora (3 derivaciones bipolares), este programa de interfaz fue realizado en *Borland C* bajo *MS-DOS*. Al ejecutar el programa, aparece una pantalla en modo gráfico para que el usuario introduzca los datos generales del paciente, como son: su nombre, apellido paterno, apellido materno, edad, peso, talla, presión arterial, etc., y con esto se podría construir una base de datos de todos los pacientes con su respectiva señal de electrocardiograma, creando un historial médico al ir almacenando las muestras, pudiéndose comparar posteriormente con muestras que le antecedieron para verificar la evolución del paciente, su posible mejoría y un tratamiento más preciso, haciendo todo este proceso más rápido y confiable. (Fig. 3.21)

Las figuras 3.24, 3.25, 3.26 y 3.27 fueron tomadas a un paciente cuando se le conectaron los electrodos y se obtuvieron las señales electrocardiográficas bipolares correspondientes a cada derivación D1, D2 y D3, para poder ser visualizados en el monitor de la PC., la figura 3.23 muestra el hardware realizado.

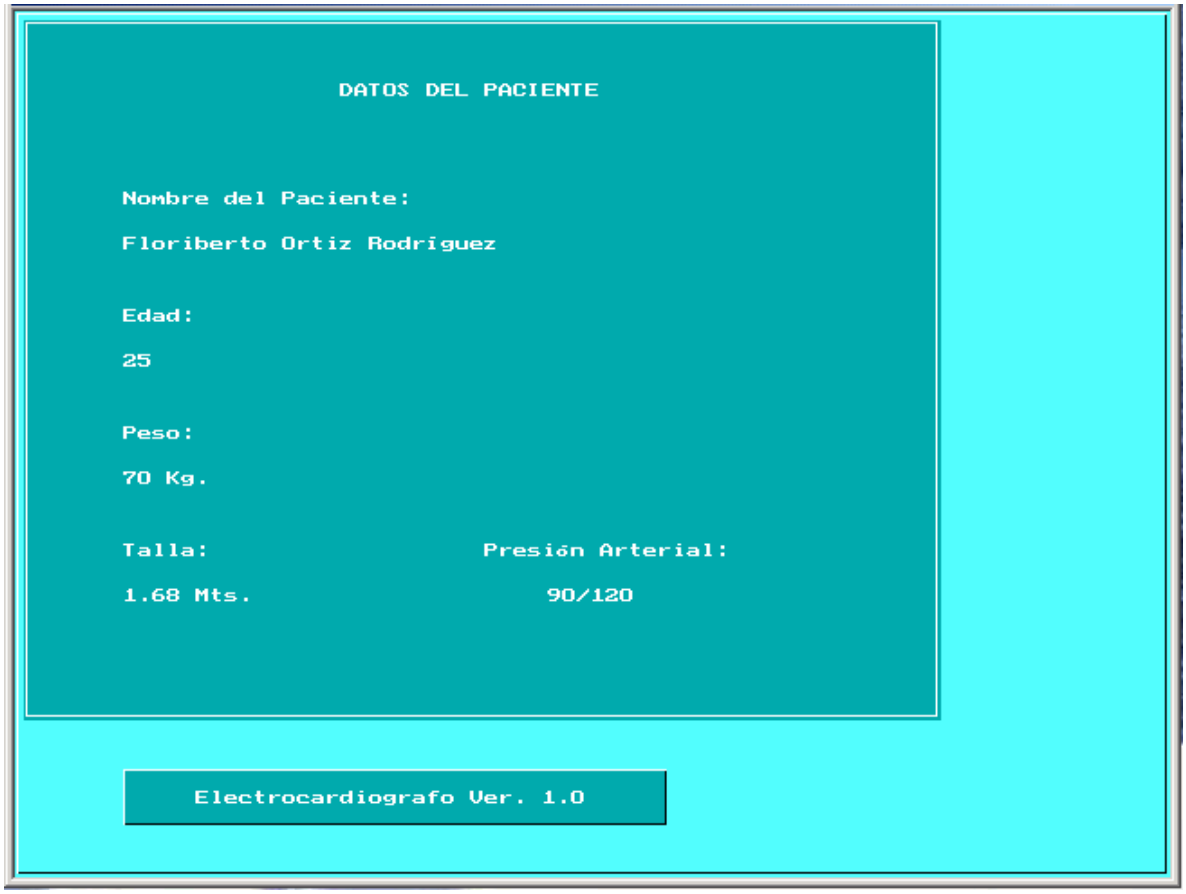


Figura 3.21 Pantalla que captura datos del paciente

Después de introducir los datos del paciente automáticamente muestra una nueva pantalla gráfica para visualizar las señales electrocardiográficas. (Fig. 3.22)

Esta pantalla muestra algunas opciones que puede ejecutar. El boton “inicio” indica el momento en que se desea iniciar la captura de datos de las respectivas derivaciones y su correspondiente visualización en la pantalla. El boton “pausa” permite que la señal se quede fija y el médico tenga tiempo suficiente de interpretar la gráfica que se presenta, esto es un electrocardiograma estático, como si estuviera impreso en papel, solo que aquí es visualizado en el monitor de la PC. Si se elije pausa se puede reiniciar de nuevo pulsando el botón inicio. El boton salir (S), es para terminar la sesión del programa de visualización de las señales electrocardiográficas.

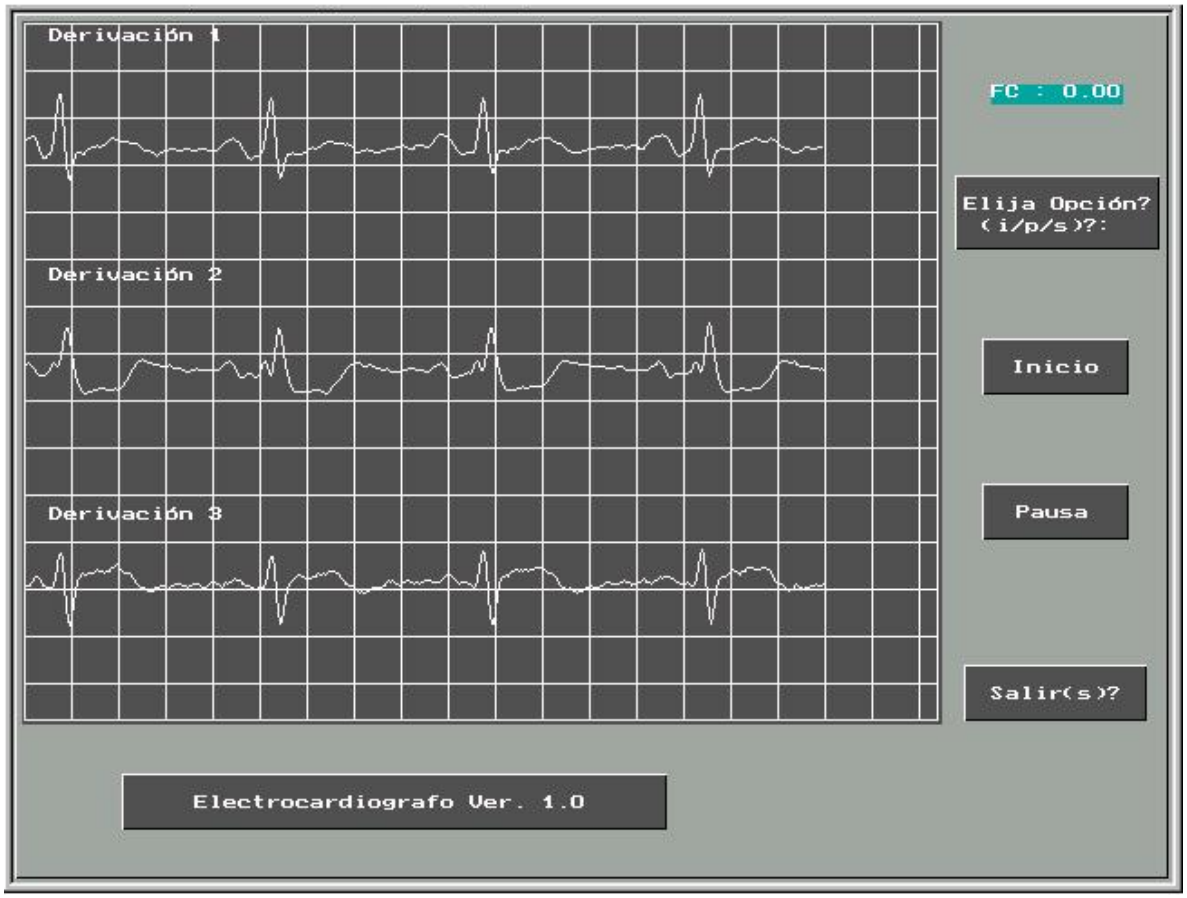


Figura 3.22 Pantalla que muestra las señales electrocardiográficas

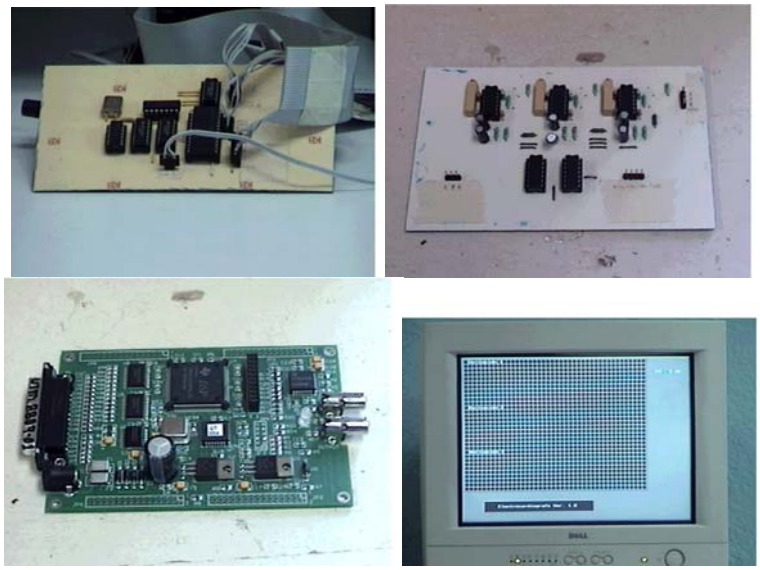


Figura 3.23 Componentes del electrocardiógrafo realizado

Figura 3.24 Hardware del electrocardiógrafo en operación

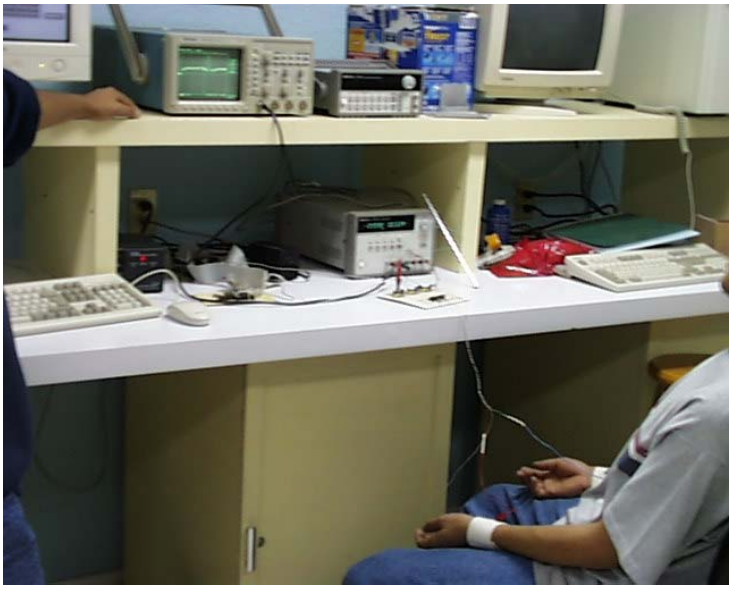


Figura 3.25 Señal del electrocardiograma observada en el osciloscopio

Figura 3.26 Adquisición de las muestras de las derivaciones bipolares

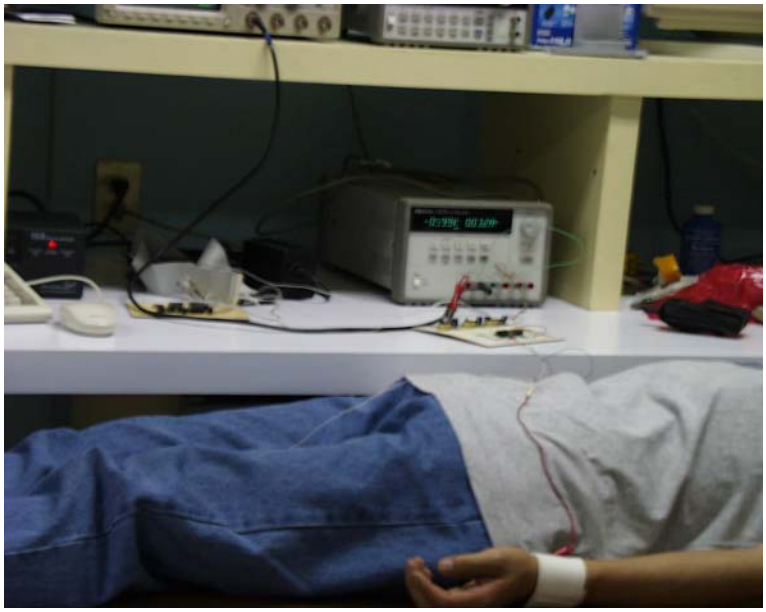


Figura 3.27 Muestras electrocardiográficas de las derivaciones bipolares D1, D2 y D3

Figura 3.28 Electrocardiograma con 3 derivaciones unipolares V1, V2 y V3, tomada con un electrocardiógrafo comercial instalado en el hospital del ISSSTE, de la región mixteca

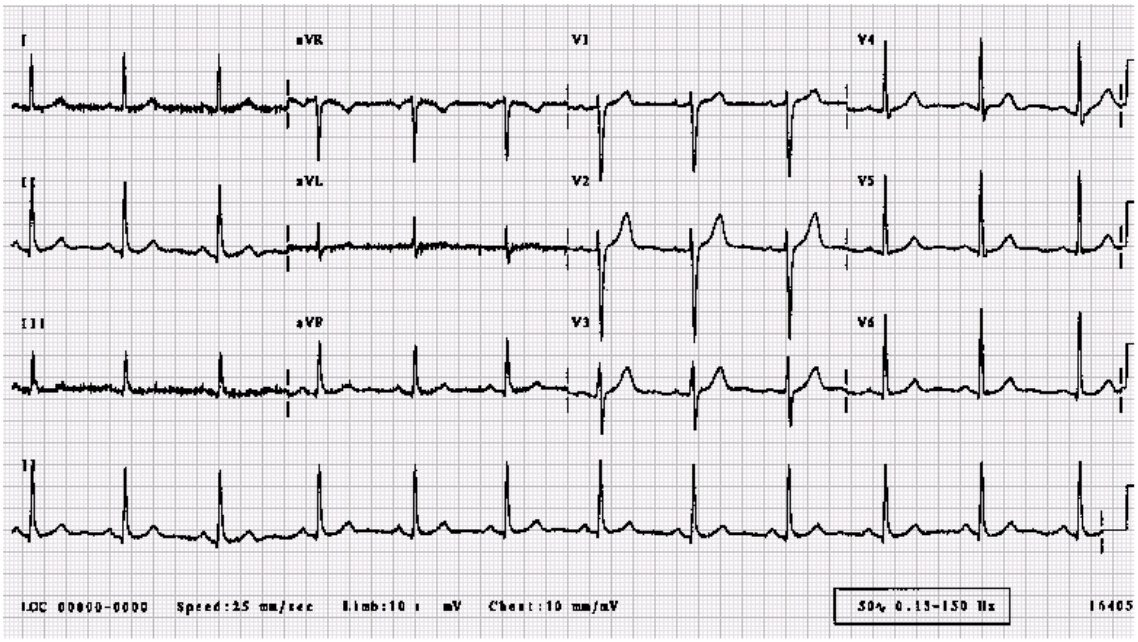


Figura 3.29 Electrocardiograma con 12 derivaciones, obtenido de un electrocardiógrafo comercial

Figura 3.30 Electrocardiógrafo comercial con una sola derivación

BPM-700 MONITOR DE PACIENTES **DATASCOPE 2100** ECG/BP/TEMP/HR

Figura 3.31 Modelos de electrocardiógrafos comerciales

Figura 3.32 Pantalla de un electrocardiógrafo comercial, basado en una PC.

Las figuras 328, 3.29 y 3.32 corresponde a señales electrocardiográficas de equipos comerciales, y las figuras 3.30 y 3.31 corresponden a diferentes modelos de electrocardiógrafos existentes en el mercado. La figura 3.28 muestra una señal electrocardiográfica de 3 derivaciones unipolares, tomada a un paciente con un electrocardiógrafo instalado en el Hospital del ISSSTE de la región mixteca y las figuras 3.29 y 3.32 muestran 12 derivaciones: 3 bipolares, 3 unipolares y 6 precordiales, de electrocardiógrafos comerciales.

3.10 Problemas encontrados durante la elaboración de la tesis.

Uno de los problemas más grandes encontrados al implementar los filtros digitales, es que no existe la bibliografía necesaria que indique como filtrar y procesar más de una señal analógica en el DSP TMS320C31, no hay información en la página de productos DSP de TI., tampoco en libros de la biblioteca o en el manual de usuario de la tarjeta. La tarjeta TMS320C3x Starter Kit trae un programa base llamado *FIR.ASM*, el cuál sirve para filtrar y procesar una señal analógica que este presente en el conector *RCA*. La manera de como funciona este programa es que se calculan los coeficientes del filtro digital en un programa diferente, en este caso se utilizará Matlab, por ser el programa utilizado en la obtención de coeficientes de filtro. Una vez obtenidos estos coeficientes utilizando alguna función en Matlab de un filtro FIR, se copian y se colocan dentro de la estructura del programa reservado para tal propósito, que tiene como encabezado *FIR_coef*. Para determinar la frecuencia de muestreo con la cuál será adquirida la señal a través del CODEC interno de la tarjeta se utiliza el programa *AICCALC*. Una vez dados estos datos se obtienen los valores que serán puestos en los registros *TA, TB* en el registro de periodo del timer, cuando se tienen todos estos datos se le mete una señal a la tarjeta TMS320C3x Starter Kit por medio de un generador de señales y se observa en el osciloscopio conectado a la salida del conector *RCA* si efectivamente el diseño del filtro cumple con las espectativas del diseñador, de no ser así, se empieza por calcular de nuevo los coeficientes del filtro.

Una vez que se ha entendido el funcionamiento del filtrado digital en el DSP TMS320C31 se procede a implementar una estructura nueva e igual para los coeficientes de una segunda señal analógica y también se deben reservar los espacios necesarios donde se pondrán las entradas (muestras *x(n)*), así como las funciones *DAC* y *ADC* que es en donde realmente se realiza el filtrado, se debe de copiar cambiando todos los apuntadores, registros y etiquetas por nuevos que no se encuentren en la estructura del primer filtro.

Este es uno de los pasos más críticos pues se encontrarán con los más grandes problemas, es aquí donde se utilizará el emulador *DSK3D* para ver lo que va sucediendo línea a línea con cada uno de los apuntadores. En primer lugar no se podrán ver los registros al estar las interrupciones activadas, se tiene que hacer un programa que no ocupe interrupciones a la hora de muestrear una señal, este programa que viene anexo en el apéndice correspondiente. Durante la simulación los apuntadores del filtro de la señal 1, no presentan problema alguno, sin embargo en la señal 2, por alguna razón los apuntadores de los coeficientes del filtro pierden la dirección durante el direccionamiento circular y llegan a apuntar a los coeficientes y al espacio reservado para las muestras *x1(n)* del filtro1, después de emular línea por línea el programa, se encontró que con un factor de corrección de +2 a los siguientes apuntadores del filtro 2, funcionan los apuntadores correctamente:

```
ADC_first2    .word   (ADC_recv2+2)
ADC_end2      .word    FIR_coef2
ADC_last2     .word   (ADC_recv2+2)
FIR_coefx2    .word    FIR_coef2
```

Con las siguientes pruebas en el emulador se constató que los apuntadores a los coeficientes del filtro 2, no sufrían ninguna pérdida durante el direccionamiento circular del registro *BK*. Este factor de corrección permaneció también para los apuntadores del filtro 3.

Es importante que si desean implementar más de un filtro digital en el programa conserven la estructura del programa *FILTRO.ASM* mostrado en el apéndice B, pues ha sido ampliamente probado. En caso contrario se podrían tener errores y la señal de salida tendería a presentar algunos errores, visualizándose como distorsión. Otro dato importante es que la frecuencia de la señal electrocardiográfica es relativamente baja, por consiguiente el tiempo en el procesamiento digital es amplio y permite fácilmente procesar digitalmente las tres señales sin distorsionarlas. Una vez que se demostró que se pueden procesar digitalmente más de una señal en el DSP TMS320C31, se le agregaron las interrupciones al programa, pues estas interrupciones determinaban la frecuencia de transmisión de datos por el puerto paralelo. Para que los datos puedan ser leídos y puedan ser enviados a la dirección del puerto paralelo es necesario definir la dirección del puerto como *.word* y después, en el cuerpo del programa redireccionarlo con los registros *ARn*. Cuando se ocupa más de un bloque para el filtrado de dos o más señales se recomienda que estos filtros sean del mismo orden. Si conservan el mismo orden al registro *BK* no será necesario cambiar su tamaño cada vez que se cambie de filtro, si los filtros son de orden diferente la complejidad computacional se incrementará notablemente y el registro *BK* perderá su eficiencia de direccionamiento circular, teniéndose que ocupar otros registros y será necesario modificar considerablemente los bloques de procesamiento digital donde se realiza el filtrado.

Dado que el set de instrucciones y la estructura de los programas del DSP TMS320C3x es complejo, se recomienda ampliamente documentar los programas, no importando si es línea por línea.

Para el cálculo de la frecuencia cardiaca del corazón del paciente, se realizó una función llamada *autocorrelación()*; dentro del programa de visualización de la señal electrocardiografica en C.(*filtro.cpp*), para esto se necesita conocer la frecuencia con la cuál se recibían los datos, es decir cuantas muestras por segundo se están adquiriendo. Dentro del programa de ensamblador

se especifica a través de las interrupciones del Timer 0, cada que tiempo se tiene que transmitir el dato procesado. Un problema por la cuál esta función daba datos poco precisos, pues no coincidía la frecuencia cardiaca (FC) mostrada en la PC con la del osciloscopio, es porque en realidad se hacían mas interrupciones de las que mostraba el programa *AICCALC.exe*, el cuál da la frecuencia del Timer 0. Se corrigió aumentando el tamaño del arreglo que contiene las muestras de la señal sobre la cuál se aplica la autocorrelación, para que sea más precisa a la hora de detectar el siguiente pico más alto, se aumentó también dentro de la función el número que representa las muestras por segundo, la cuál se utiliza para dividir al periodo y obtener la frecuencia.

Correlación de señales discretas.

La correlación es una operación entre dos secuencias, que permite medir el parecido que existe entre dos señales y extraer determinada información que dependerá exclusivamente de la aplicación concreta considerada [6]. Supóngase que se tienen dos secuencias *x(n)* e *y(n)* que se quieren comparar, en radar y sonar , por ejemplo, *x(n)* puede representar muestras de la señal que serán transmitidas y *y(n)* muestras de la señal que serán recibidas. Si existe un blanco en el espacio explorado por el radar o sonar, la señal recibida *y(n)* es una versión retardada de la señal transmitida, reflejada por el blanco y corrompida por ruido aditivo. Por otra parte , si no existe ningún blanco en el espacio explorado por el radar o el sonar, la señal recibida constará únicamente de ruido [6]. El cálculo de la correlación cruzada realiza las operaciones de desplazamiento de una de las secuencias, multiplicación de ambas y suma de todos los términos de la secuencia producto. En el caso especial de que *y(n)* = *x(n)*, se tiene la autocorrelación de *x(n)* [6].

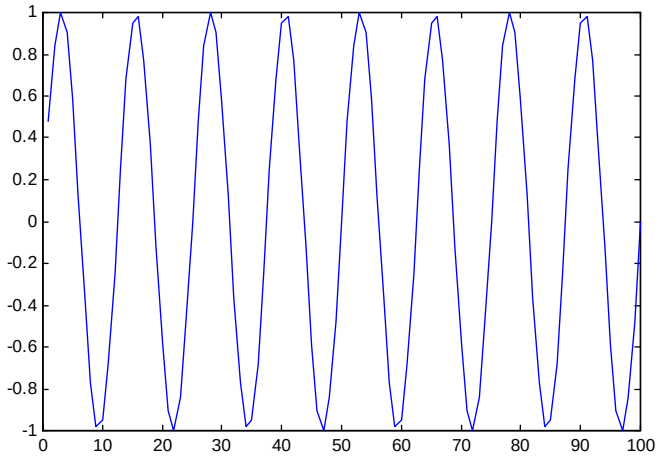


Figura 3.33 Señal senoidal de entrada

Con la explicación dada anteriormente, se podrá comprender de manera más sencilla la utilización de esta operación en el programa. Lo que se hizo fue utilizar la operación de autocorrelación en una señal de entrada, conocida como derivación 2 (D2), con la finalidad de amplificar dicha señal y poder más fácilmente programar la función que determine la frecuencia cardiaca del paciente. Computacionalmente es más sencillo determinar la frecuencia de una señal autocorrelacionada, pues a partir del punto más alto que presenta dicha señal (que se encuentra a la mitad, según la figura 3.34, en el número 100), se mide cuanto tiempo tarda en llegar al siguiente pico (aproximadamente, al número 118 de la misma figura), este tiempo se calculó con ayuda de la frecuencia de transmisión de datos, y al conocerlo se puede conocer el periodo y por consiguiente la frecuencia. Al resultado final (frecuencia) se multiplica por 60 y se obtiene las pulsaciones por minuto del corazón del paciente. La figura 3.33 muestra una señal con un periodo P, la figura 3.34 muestra la misma señal autocorrelacionada con el mismo periodo P.

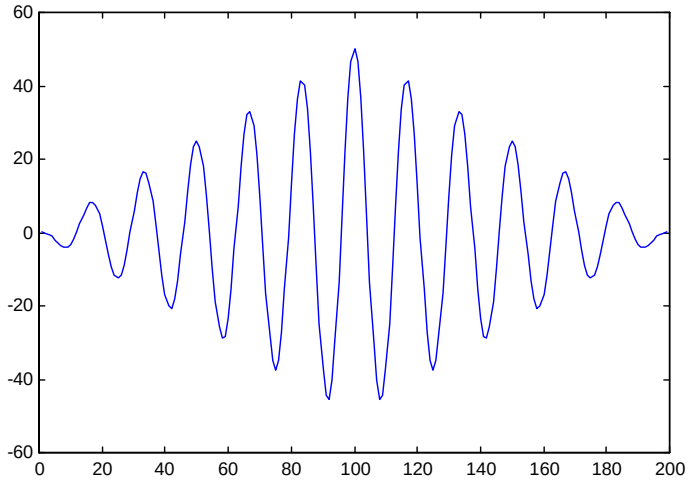


Figura 3.34 Señal senoidal con autocorrelación

Otro problema que se presentó fue al leer los datos del puerto paralelo en el programa en C, este no los capturaba, se leían datos que no correspondían y rara vez se leían los datos requeridos, para que el puerto paralelo fuera bidireccional, es decir poder enviar datos (como en el caso de una impresora) y poder recibirlos (como es el caso, recibir la señal electrocardiográfica), se configuró el puerto paralelo desde el bios como *EPP* en la opción modo del puerto, pues este modo es bidireccional. El segundo problema se presenta cada vez que se lee el puerto paralelo con la instrucción *inportb(0x378)* se coloco un retardo con la función *delay()* de C, para que el puerto tuviera el tiempo suficiente para leer el nuevo dato enviado por el DSP TMS320C31.

Configuración de la cuadrícula de estandarización del electrocardiograma.

El registro de la señal electrocardiográfica presenta una cuadrícula patrón. En la cuadrícula las líneas delgadas están separadas por 1 mm, y las gruesas por 5 mm. La distancia entre dos líneas horizontales delgadas corresponden a 0.1 mV., para conocer la duración de un fenómeno, se cuentan las líneas verticales

que separan los puntos donde empieza y donde termina, dicho fenómeno es interpretado por el médico. (Fig. 3.35)

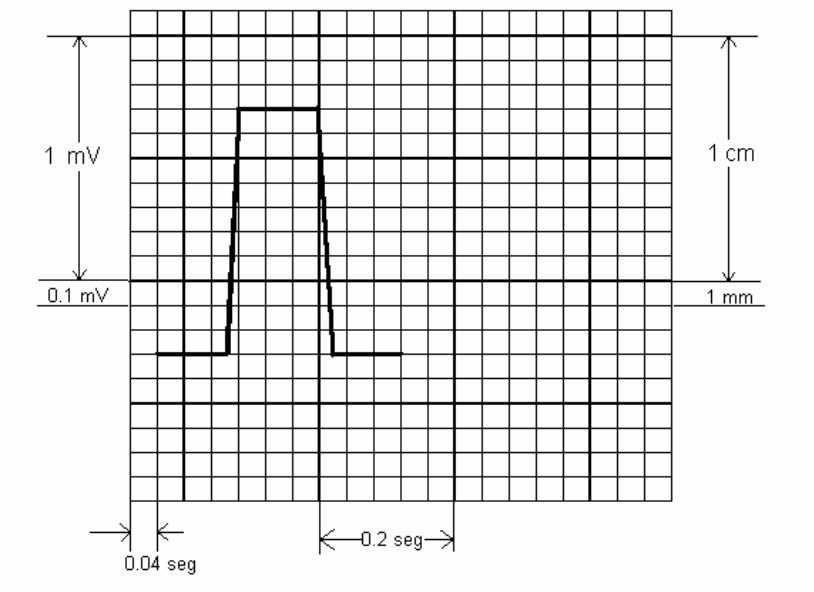


Figura 3.35 Onda de estandarización.

La señal electrocardiográfica se configuró con una señal cuadrada de prueba, como se ve en la figura 3.35, que entrega el generador de señales, esta señal presenta una frecuencia de 1 Hz. con una amplitud de 1 Vpp. y un nivel de offset de 500 mV. Tomando en cuenta este patrón y con la señal visualizada en el monitor de la PC. Se dibujan dichas líneas, en el programa *filtro.cpp*.

CAPÍTULO IV.

Perspectivas y conclusiones.

4.1 Perspectivas.

Una de las perspectivas más viables, aprovechando la capacidad de la PC., es la creación de un módulo de sistema experto, este módulo será capaz de hacer un diagnóstico médico para enfermedades cardiacas que el paciente padece, ha padecido o puede llegar a padecer a partir de la gráfica del electrocardiograma, hacer un seguimiento del comportamiento del corazón y dar un posible tratamiento para poder subsanar este problema de salud. Está enfocado a médicos generales que requieran asistencia en la detección de enfermedades cardiacas, será un auxiliar muy importante para médicos especialistas en el área cuando se trate de hacer un diagnóstico y de recetar posibles medicamentos, así como elaborar un tratamiento efectivo para cada caso en particular.

Otra perspectiva es hacer el electrocardiógrafo móvil, interconectando a la tarjeta TMS320C3x Starter Kit dispositivos de memoria suficientes para poder almacenar datos de las señales electrocardiográficas del paciente, para después bajar estos datos a la PC, vía puerto paralelo (o cualquier otro puerto) y poder observar la gráfica. Para esto no es necesario que el paciente acuda al lugar donde se encuentra el electrocardiógrafo, un médico general con los conocimientos necesarios en el uso del mismo, pueda tomar las muestras y enviarlas después con el especialista para que haga el diagnóstico acerca del padecimiento del enfermo a través de las muestras adquiridas.

De los electrocardiógrafos consultados, ninguno de ellos presentaba alguna opción para hacer un seguimiento de las señales electrocardiográficas de los

pacientes, para poder comprender si su evolución ha sido satisfactoria o ha presentado algunas complicaciones. Una perspectiva interesante dentro de la programación será crear una base de datos de todos los pacientes, de sus señales electrocardiográficas que se le hayan tomado, comparar como han ido evolucionando y hacer el seguimiento de cada una de sus posibles enfermedades.

El electrocardiógrafo sería más versátil si se le pudiera interconectar una pantalla de LCD. (display de cristal líquido), para que las muestras sean visualizadas en el momento, sin necesidad de esperar a contar con una PC. para poder hacerlo, este LCD. sería ligero y pequeño, pudiéndose transportar a cualquier parte, con esto se tendría un electrocardiógrafo móvil.

Si por alguna razón se tienen todos estos aditamentos, pero no se cuenta con un especialista y solo se puede localizarlo por teléfono (o internet), una perspectiva a futuro será hacer un dispositivo que permita que las señales electrocardiográficas puedan viajar por la línea telefónica (y en su caso por internet) y que esta línea este conectada a la PC. por el otro extremo para poder graficarla, sin que haya alguna necesidad de que el especialista se mueva de su consultorio, con esto se asegura que el diagnostico será el correcto además de minimizar los gastos que se pudieran generar al traer a un especialista o que el paciente tenga que viajar.

4.2 Conclusiones.

El electrocardiógrafo realizado en este proyecto de Tesis comprendío en primer lugar el entendimiento del set de instrucciones del lenguaje ensamblador del DSP TMS320C31 y de la estructura que presentan los programas, tomándose como base algunos de estos que vienen incluidos dentro del software de la tarjeta TMS320C3x Starter Kit. El siguiente paso fue interconectar un dispositivo externo al DSP TMS320C31 a través de su bus primario, en este caso fue el convertidor ADC0808, con el previo entendimiento de los tiempos de acceso, de conversión, y de las líneas de control de cada uno de los dispositivos. Se le implementó un reloj de 500 KHz. al convertidor ADC0808, el cuál está compuesto por un cristal de 4 MHz, para darle estabilidad a la señal de reloj, y de un dispositivo TTL 74LS161A que funciona como un divisor de frecuencias, reduciendo la frecuencia de entrada de 4 MHz. que produce el cristal a una frecuencia de salida de 500 KHz., que proporciona el divisor de frecuencias y que va conectada a la entrada *CLK* del ADC0808, este convertidor soporta un reloj que oscile en un rango de 500 KHz. a los 640 KHz.

Cuando se comprobó que estos programas y que la interfaz ADC0808/DSP TMS320C31 funcionaban de manera correcta, se armó un circuito de instrumentación que permitiera obtener la señal de electrocardiograma del cuerpo humano, a través de sus 3 derivaciones bipolares conocidas como D1, D2 y D3. Se utilizaron electrodos adheridos a la piel que obtienen los impulsos eléctricos y que llevan esta señal a los amplificadores de instrumentación, los cuáles se hicieron con un TL084, que trae integrados 4 TL081, 3 fueron utilizados para el circuito de instrumentación y 1 para dar un nivel de offset a la señal, pues el ADC0808 solo acepta datos positivos. Se desarrolló también un programa en C que permitiera visualizar los datos de la señal electrocardiográfica, comparándolo previamente con la proporcionada por el osciloscopio.

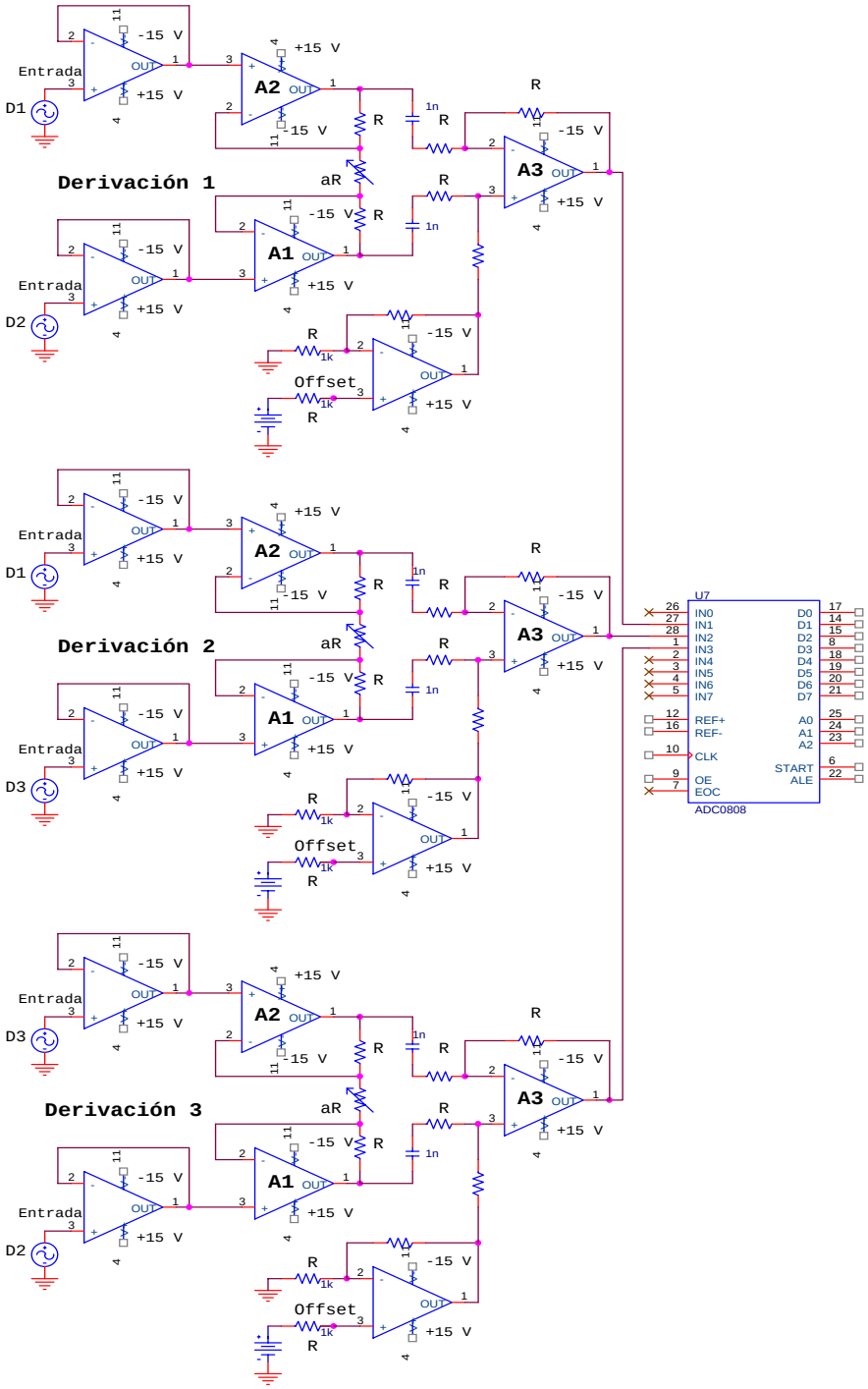
Cuando el sistema funcionaba correctamente, los circuitos armados previamente, como la interfaz ADC0808/DSP TMS320C31 y los amplificadores de instrumentación se montaron definitivamente en un circuito impreso que fue realizado en Orcad9 y se corrigieron algunos detalles del programa en C, como la función de autocorrelación que proporciona la frecuencia cardiaca del paciente y la estandarización de la cuadricula sobre la cuál se dibuja la señal digital del electrocardiograma.

Con base en los objetivos planteados inicialmente, se considera que estos fueron cumplidos al obtener una señal electrocardiográfica con niveles de ruido poco perceptibles, similares a la presentada por los equipos médicos comerciales que se muestran en las figuras: 3.28, 3.29 y 3.32. Como puede observarse en el capitulo III, sección 3.9: la figura 3.22 corresponde a las derivaciones bipolares D1, D2 y D3 del electrocardiógrafo basado en la tarjeta TMS320C3x Starter Kit. Las señales electrocardiográficas presentes en dicha figura, fueron validadas por el Dr. Juan Carlos Ramos A., médico anestesiólogo que labora en la región mixteca. Quedando abierta la posibilidad de que la interfaz visual del programa en Borland C, pudiera mejorarse como lo muestra la figura 3.32, así como la presentación final del hardware, según se observa en los ECG comerciales correspondientes a las figuras 3.30 y 3.31.

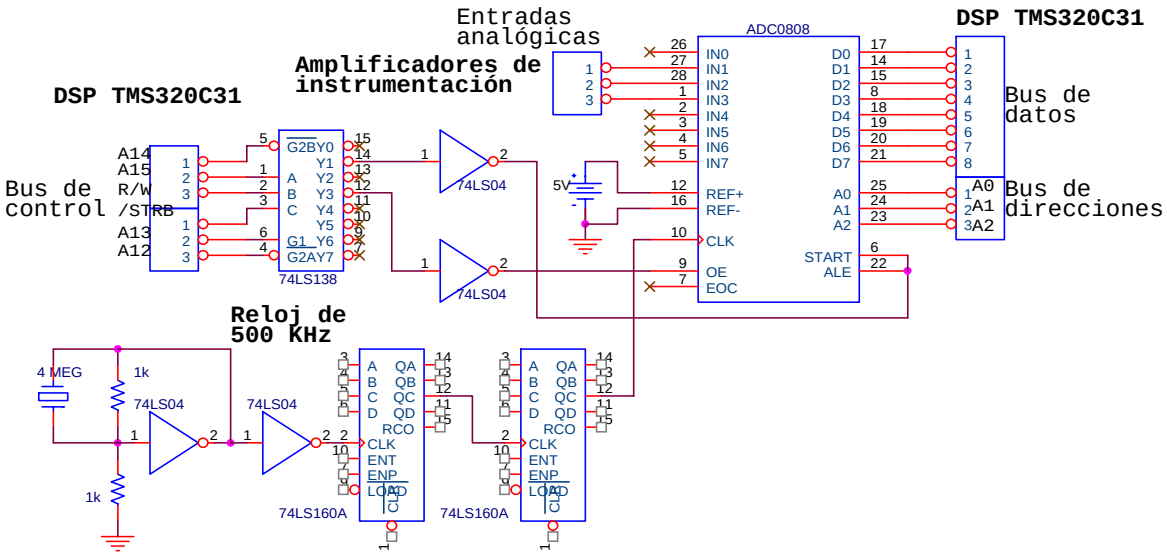
APÉNDICE A.

Esquemáticos e impresos de los circuitos realizados.

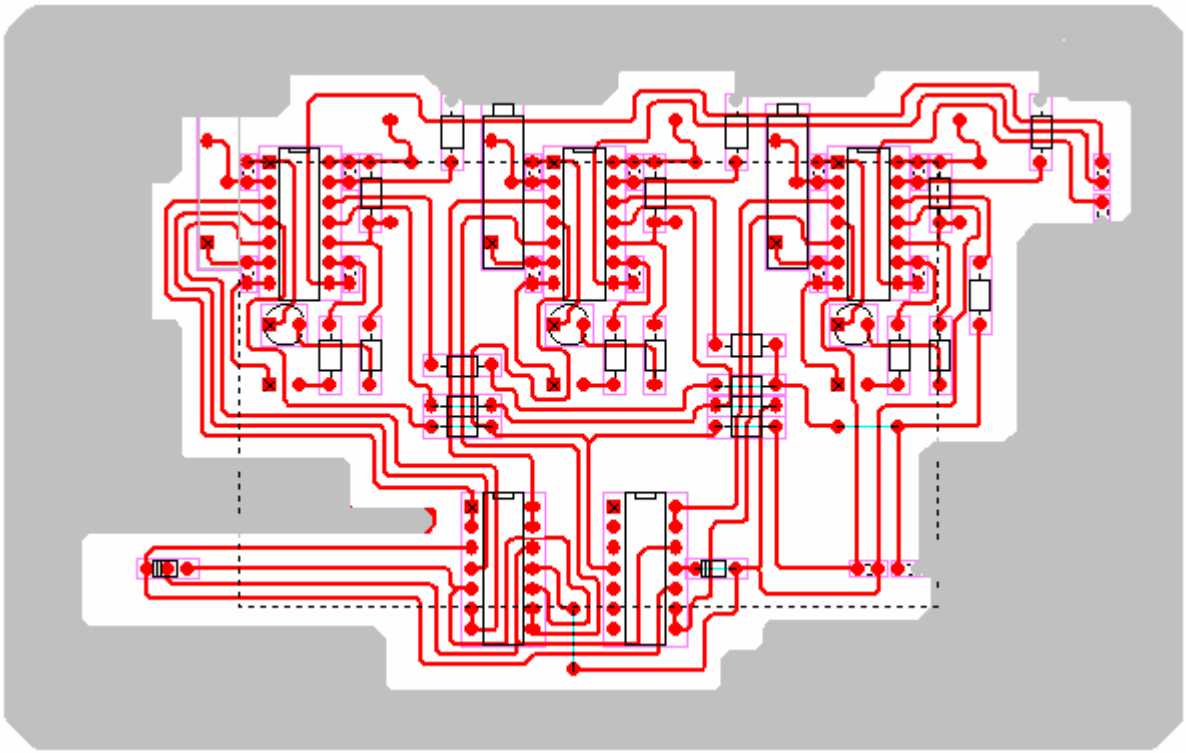
Esquemático de los circuitos de instrumentación.



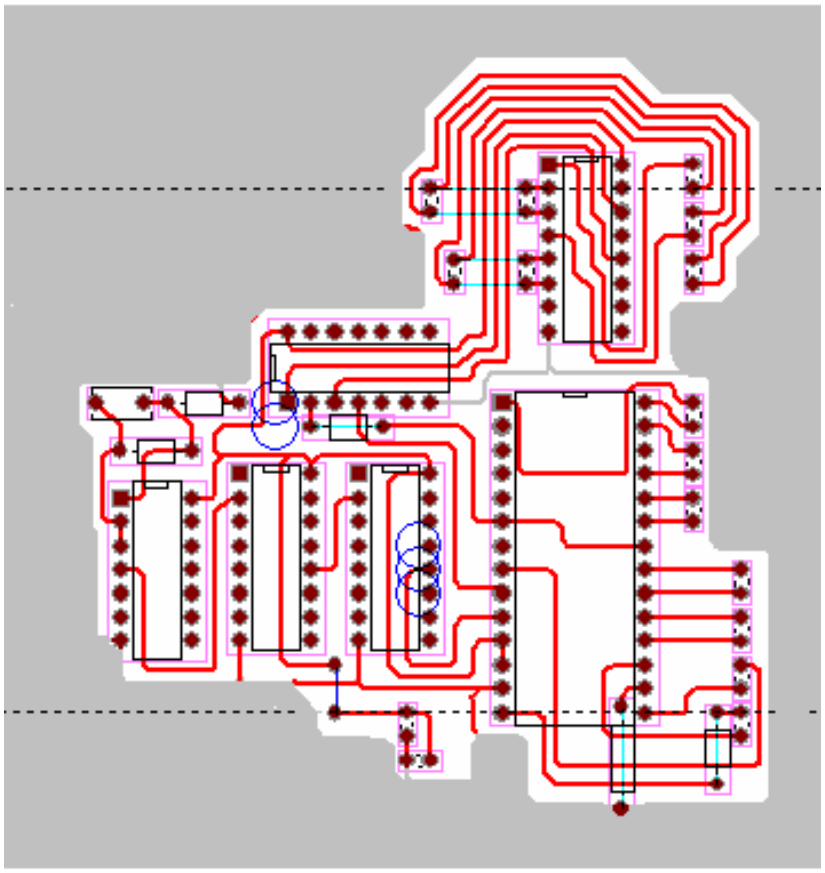
Esquemático de la interfaz DSP TMS320C31- ADC0808.



Circuito impreso de los amplificadores de instrumentación.



Circuito impreso de la interfaz DSP TMS320C31- ADC0808.



APÉNDICE B.

Código fuente de los programas realizados.

Programa de adquisición, filtrado y transmisión de datos a la PC.

Filtro.asm

```
;-----------------------------------------------
  .start ".text",0x809810      ; inicio de mem. en el chip
  .start ".servect",0x809FC9  ; Timer 0 interrupt vectors
  .sect ".text"
;-----------------------------------------------
;--------- Mapa de memoria a utilizar --------
ADC_recv   .float    0.0          ; el filtro es de orden 15, por lo tanto se reservan 16 espacios en memoria
           .float    0.0          ; para los datos de entrada que serán filtrados
           .float    0.0          ; estos espacios llevan un identificador
           .float    0.0          ; en este caso se llaman ADC_recv, ADC_recv2 y ADC_recv3
           .float    0.0          ;
           .float    0.0          ;
           .float    0.0          ;
           .float    0.0          ; bloque para filtrar la señal proveniente de la derivación 1
           .float    0.0          ;
           .float    0.0          ;
           .float    0.0          ;
           .float    0.0          ;
           .float    0.0          ;
           .float    0.0          ;
           .float    0.0          ;
;- - - - - - - - - - - - - - - - - - - - - - - -
; FIR filter coefficients
;- - - - - - - - - - - - - - - - - - - - - - - -
FIR_coef   .float    0.0218       ;  espacio reservado para los coeficientes del filtro
           .float   -0.0196       ; estos coeficientes fueron obtenidos con el software
           .float   -0.0235       ;  de Matlab
           .float   -0.0207       ; están identificados al inicio de bloque con los nombre de variables
           .float    0.0285       ;  FIR_coef, FIR_coef2 y FIR_coef3
           .float    0.1272       ;  estos bloques están limitados por los nombres de variables
           .float    0.2232       ;  END_coef, END_coef2 y END_coef3
           .float    0.2768
           .float    0.2232
           .float    0.1272
           .float    0.0285
           .float   -0.0207
           .float   -0.0235
           .float   -0.0196
           .float    0.0218
END_coef
;--------- Mapa de memoria a utilizar --------
ADC_recv2  .float    0.0          ; bloque para filtrar la señal proveniente de la derivación 2

           .float    0.0          ;
           .float    0.0          ;
           .float    0.0          ;
           .float    0.0          ;
           .float    0.0          ;
           .float    0.0          ;
           .float    0.0          ;
           .float    0.0          ;
```

```
            .float    0.0           ;
            .float    0.0           ;
            .float    0.0           ;
            .float    0.0           ;
            .float    0.0           ;
            .float    0.0           ;
            .float    0.0           ;
            .float    0.0           ;
;- - - - - - - - - - - - - - - - - - - - - - - -
FIR_coef2  .float    0.0218      ; los coeficientes son los mismos para los
            .float   -0.0196       ; tres filtros, su frecuencia de corte será la misma
            .float   -0.0235
            .float   -0.0207
            .float    0.0285
            .float    0.1272
            .float    0.2232
            .float    0.2768
            .float    0.2232
            .float    0.1272
            .float    0.0285
            .float   -0.0207
            .float   -0.0235
            .float   -0.0196
            .float    0.0218
END_coef2
;--------- Mapa de memoria a utilizar --------
ADC_recv3  .float    0.0          ; bloque para filtrar la señal proveniente de la derivación 2

            .float    0.0           ;
            .float    0.0           ;
            .float    0.0           ;
            .float    0.0           ;
            .float    0.0           ;
            .float    0.0           ;
            .float    0.0           ;
            .float    0.0           ;
            .float    0.0           ;
            .float    0.0           ;
            .float    0.0           ;
            .float    0.0           ;
            .float    0.0           ;
            .float    0.0           ;
            .float    0.0           ;
;- - - - - - - - - - - - - - - - - - - - - - - -
FIR_coef3  .float    0.0218
            .float   -0.0196
            .float   -0.0235
            .float   -0.0207
            .float    0.0285
            .float    0.1272
            .float    0.2232
            .float    0.2768
            .float    0.2232
            .float    0.1272
            .float    0.0285
            .float   -0.0207
            .float   -0.0235
            .float   -0.0196
            .float    0.0218
END_coef3
```

```
;----------------------------------------
;- - - - - - - - - - - - - - - - - - - - -
BufSz         .set      END_coef - FIR_coef
;- - - - - - - - - - - - - - - - - - - - -
SIZE         .word    BufSz             ; se obtiene el tamaño del filtro
ADC_first    .word    ADC_recv          ; apuntadores a los filtros
ADC_end      .word    FIR_coef
ADC_last     .word    ADC_recv
FIR_coefx    .word    FIR_coef
;- - - - - - - - - - - - - - - - - - - - -
ADC_first2    .word    (ADC_recv2+2)   ; apuntadores al filtro 2 con su respectivo
ADC_end2      .word    FIR_coef2       ; factor de corrección para que los apuntadores
ADC_last2     .word    (ADC_recv2+2)   ; no se pierdan y obtengan los datos correctos
FIR_coefx2    .word    FIR_coef2
;- - - - - - - - - - - - - - - - - - - - -
ADC_first3    .word    (ADC_recv3+2)   ; apuntadores al filtro 3
ADC_end3      .word    FIR_coef3
ADC_last3     .word    (ADC_recv3+2)
FIR_coefx3    .word    FIR_coef3

  .entry start
;-----------------------------------------------
;--------- Mapa de memoria a utilizar --------
TGCR0   .set  0x808020        ; Timer 0 global control register
TCNT0    .set  0x808024       ; Timer 0 counter register
TPR0     .set  0x808028       ; Timer 0 period register

RCG     .set  0x808064        ; Reg de control del Bus Primario
          .text

start    ldi  0h,R1
        ldi  0h,R4
        ldi  0h,R5
        ldi  0h,AR3
        ldi  0h,AR4
        sti  R4,@RCG          ; RCG-->0,Reset al Bus Primario
        ldi  @RCGVAL,R1       ; R1-->(RCGVAL-->0x0978)
        sti  R1,@RCG          ; RCG-->0x0978, Config el Bus,
        ldi  51h,R1
        ldi  @DATO,AR6        ; accesar a un disp. periférico
        ldi  @paralelo,AR7    ; direccion del puerto

        or   2000h,ST         ; Global interrupt enable
        sti  R4,@TGCR0        ; Reset timer 0
        ldi  TIMERPER,R7
        sti  R7,@TPR0         ; Store timer 0 period
        sti  R4,@TCNT0        ; Reset timer 0 counter
        ldi  @TIMVAL,R7       ; Load timer control value
        sti  R7,@TGCR0        ; Start timer 0
        or   100h,IE          ; Enable RINT interrupts
loop
        br   loop

conv     cmpi 54h,R1
        calleq Re_Dir
        sti  R4,*AR6          ; se activa ALE y START
        call espera
        ldi  *AR6,R5          ; RD, TMS lee la conversion  se activa OE

        sti  R1,*AR7          ; dat0 inicio al Pto. paralelo

        cmpi 51h,R1           ; se compara el dato de inicio para llamar al filtro correspondiente
```

```
        calleq  DAC1       ; función para filtrar los datos de la derivación 1

        cmpi 52h,R1
        calleq  DAC2       ; función para filtrar los datos de la derivación 2

        cmpi 53h,R1
        calleq  DAC3

        sti  R5,*AR7       ; se escribe al Pto. Paralelo

        addi 01h,R1        ; se suma 1 para indicar que corresponde a la siguiente
        addi 01h,AR6       ; derivación, si llega a la más grande, reinicializa con el primero
        reti

espera   ldi  01f4h,R4       ; 1f4h, ciclo de espera de 100 uS
decrem   subi 01h,R4         ; tiempo en el que el Adc0808 convierte el dato
        bnz  decrem           ; analógico a discreto
        rets

Re_Dir   ldi  51h,R1
        ldi  @DATO,AR6
        rets

;--------------------------------
DAC1      push  ST
        push  R0            ; función para filtrar la señal de la derivación 1
        pushf R0            ;
        push  R2            ;
        pushf R2            ;
        push  R3            ;
        pushf R3            ;
        push  AR0
        push  AR2
        push  AR4
        ldi   R5,R3         ; Dato Recibido

        ldi   @ADC_last,AR4   ; asigna dirección de bloque
        float R3,R3
        stf   R3,*AR4++       ; pone dato en bloque e incrementa

        cmpi  @ADC_end,AR4    ; compara dirección de coef, con dato de  inicio
        ldige @ADC_first,AR4  ; si es eq redirec.
        sti   AR4,@ADC_last

        ldi   @ADC_last,AR2   ;
        ldi   @FIR_coefx,AR0  ;
        ldi   @SIZE,BK            ; se le asigna al registro BK, el tamaño del bloque

FIR1      mpyf3 *AR0++,*AR2++(1)%,R0
        ldf   0.0,R2
        ldi   @SIZE,RC
        subi  2,RC
        rptb  FIR21               ; indica que bloque será repetido
        mpyf3 *AR0++(1)%,*AR2++(1)%,R0       ; realiza la multiplicación entre los coeficientes y los datos
FIR21 ||  addf3 R0,R2,R2                     ; realiza la suma de los datos para obtener finalmente
        addf  R2,R0                          ; el dato de salida filtrado y(n)

        fix   R0,R0
        ldi   R0,R5         ; Output the new DAC value
        pop   AR4
        pop   AR2
        pop   AR0
```

```
        popf  R3            ;
        pop   R3            ;
        popf  R2            ;
        pop   R2            ;
        popf  R0            ;
        pop   R0            ;
        pop   ST            ;
        rets              ;
;--------------------------------
DAC2    push  ST
        push  R0            ; R0 --> R1
        pushf R0            ;
        push  R2            ; R2 --> R7
        pushf R2            ;
        push  R3            ; R3 --> R6
        pushf R3            ;
        push  AR0
        push  AR2
        push  AR4

        ldi   R5,R3         ; Dato Recibido

        ldi   @ADC_last2,AR4   ; asig. dir. de bloque
        float R3,R3
        stf   R3,*AR4++       ; pone dato en bloque e inc

        cmpi  @ADC_end2,AR4    ; cmp dir. coef, con dat ini
        ldige @ADC_first2,AR4  ; si es eq redirec.
        sti   AR4,@ADC_last2

        ldi   @ADC_last2,AR2   ;
        ldi   @FIR_coefx2,AR0  ;
        ldi   @SIZE,BK

FIR2    mpyf3 *AR0++,*AR2++,R0
        ldf   0.0,R2
        ldi   @SIZE,RC
        subi  2,RC
        rptb  FIR22
        mpyf3 *AR0++,*AR2++(1)%,R0
FIR22 ||  addf3 R0,R2,R2
        addf  R2,R0

        fix   R0,R0
        ldi   R0,R5         ; Output the new DAC value

        pop   AR4
        pop   AR2
        pop   AR0
        popf  R3            ;
        pop   R3            ;
        popf  R2            ;
        pop   R2            ;
        popf  R0            ;
        pop   R0            ;
        pop   ST            ;
        rets              ;
;--------------------------------
DAC3    push  ST
        push  R0            ;
        pushf R0            ;
        push  R2            ;
```

```
        pushf R2            ;
        push  R3            ;
        pushf R3            ;
        push  AR0
        push  AR2
        push  AR4

        ldi   R5,R3         ; Dato Recibido

        ldi   @ADC_last3,AR4   ; asig. dir. de bloque
        float R3,R3
        stf   R3,*AR4++       ; pone dato en bloque e inc

        cmpi  @ADC_end3,AR4    ; cmp dir. coef, con dat ini
        ldige @ADC_first3,AR4  ; si es eq redirec.
        sti   AR4,@ADC_last3

        ldi   @ADC_last3,AR2   ;
        ldi   @FIR_coefx3,AR0  ;
        ldi   @SIZE,BK

FIR3    mpyf3 *AR0++,*AR2++,R0
        ldf   0.0,R2
        ldi   @SIZE,RC
        subi  2,RC
        rptb  FIR23
        mpyf3 *AR0++,*AR2++(1)%,R0
FIR23 ||  addf3 R0,R2,R2
        addf  R2,R0

        fix   R0,R0
        ldi   R0,R5         ; Output the new DAC value

        pop   AR4
        pop   AR2
        pop   AR0
        popf  R3            ;
        pop   R3            ;
        popf  R2            ;
        pop   R2            ;
        popf  R0            ;
        pop   R0            ;
        pop   ST            ;
        rets              ;

TIMERPER .set  55          ; (768)384 muestras Timer period reg 55->37dec
TIMVAL     .word 0x3c1        ; Timer global control register value
RCGVAL   .word 0x00000998   ; Espera por Software con 3 estados
DATO       .word 0x0080A001   ; se habilita la señal estroboscópica
Paralelo     .word 0x00fff000   ; mapea los datos p/Pto. Paralelo

          .sect   ".servect"
          b conv            ; Timer 0
```

Programa que despliega los datos en el monitor de la PC.

Filtro.cpp

```
#include <graphics.h>         // cabecera de librerías a utilizar en el programa
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <conio.h>
#include <math.h>
#include <dos.h>
#include <time.h>
#include <stdarg.h>
#include "c:\dsktools\gtext.h"

#define  P_paralelo 0x378          // dirección del puerto paralelo y de sus líneas de control
#define  status 0x379
#define  P_control  0x37A

#define tam 500                        // tamaño del arreglo de autocorrelación
//---------------------------------------------------------------
//Programa que configura el puerto paralelo como bidireccional
//          modo del BIOS EPP
//---------------------------------------------------------------
int ventana(int x1, int y1, int x2, int y2);       // declaración de las funciones
int ini_graficos(void);
int grafico(void);
int adquiere(void);
int paciente(void);

int autocorrelacion(void);

int temp=0,dato=0,color;


char op,op1;
int fmin=0,k1=1,k2=4,i,ban=0;
int n,l=0,cont,xy=0;
float s=0,maximo, frec,s1,s2;
float v[tam],r[tam],y1[tam],y2[tam],y3[tam];
            char ch;
            int x0=7,x2=7,x3=7;
            int r1,c1,r2,c2,r3,c3;

            char *nombre,*edad,*peso,*talla,*presion;
int main(void)
{
            color=10;
            clrscr();
            ini_graficos();                    //inicializaci¢n de gr ficos
            paciente();                        //  función que captura los datos del paciente
            grafico();                          // función que muestra la cuadrícula
  do
            {

             op=getch();

             switch(op)
                        {
                         case 'i':
                                                                  grafico();        //muestra pantalla gráfica
```

```
                                                adquiere();break;   // función que captura datos del
                                                                    // puerto y los manda al monitor
                    case 's': exit(0);

                    default :
                                        outtextxy(522,150,"Dato no v lido");

                                        break;
                   }

          }while(op!='i'&&op!='s');

          grafico();        //muestra pantalla gráfica
          adquiere();       //adquiere datos de TMs
          return 0;
}

int paciente(void)
{
            setfillstyle(SOLID_FILL, 11);    //estylo del relleno y color
            bar(0,0,getmaxx(),getmaxy());
            ventana(1,1,getmaxx()-2, getmaxy()-2);

            setfillstyle(SOLID_FILL, 3);    //estylo del relleno y color
            bar(5,5,512,392);
            setlinestyle(SOLID_LINE,1,1);  //Grafica de escala
            rectangle(6,6,510,390);

            ventana(60,420,360,450);
            outtextxy(100,432,"Electrocardiografo Ver. 1.0");

                    outtextxy(180,40,"DATOS DEL PACIENTE");

                    outtextxy(60,100,"Nombre del Paciente: ");   //se obtienen datos del teclado
                      gscanfxy(60,125, nombre);
                    outtextxy(60,165,"Edad: " );
                      gscanfxy(60,190, edad);
                    outtextxy(60,230,"Peso: " );
                      gscanfxy(60,255, peso);
                    outtextxy(60,295,"Talla: " );
                      gscanfxy(60,320, talla);

                    outtextxy(260,295,"Presi¢n Arterial: " );
                      gscanfxy(295,320, presion);


 return 0;
}

int grafico(void)
{    int x,y;
            setfillstyle(SOLID_FILL, 7);    //(7)estylo del relleno y color
            bar(0,0,getmaxx(),getmaxy());
            ventana(1,1,getmaxx()-2, getmaxy()-2);

            setfillstyle(SOLID_FILL, 8);    //(8)estylo del relleno y color
            bar(5,5,512,392);

                    setlinestyle(SOLID_LINE,1,1);
            /*---1 cm----*/
            for(x=6; x<=510; x=x+26)          //dibuja las líneas para la cuadrícula
                    {                         // puede modificarse para configurar dicha cuadrícula
```

```
                        line(x,7,x,390 );
                }

        for(y=6; y<=390; y=y+26)
                {
                        line(7,y,510,y );
                }

        outtextxy(20,9,"Derivaci¢n 1");
        outtextxy(20,141,"Derivaci¢n 2");
        outtextxy(20,273,"Derivaci¢n 3");

        ventana(520,90,632,130);
        outtextxy(525,102,"Elija Opci¢n?");
        outtextxy(535,114,"(i/p/s)?:");

        ventana(535,180,615,210);
        outtextxy(552,192,"Inicio");        // botón de inicio

        ventana(535,260,615,290);
        outtextxy(554,272,"Pausa");      // botòn de pausa

        ventana(525,360,625,390);
        outtextxy(540,372,"Salir(s)?");

        ventana(60,420,360,450);
        outtextxy(100,432,"Electrocardiografo Ver. 1.0");

        gprintfxy(540,40,"FC : %.2f",frec);
        setlinestyle(SOLID_LINE,1,1);  //Grafica de escala
        rectangle(6,6,510,390);

       return 0;
}


int adquiere(void)
{

        while(!kbhit())
                {

        outportb(P_control,(0x24));
        delay(1);
        dato=inportb(P_paralelo);    //lee dato de inicio para determinar a que derivación
        outportb(P_control,(0x25));  // corresponde y graficarla correctamente

        if(dato==0x51||dato==0x52||dato==0x53)
                {
                 outportb(P_control,(0x24));
                 delay(1);
                 temp=inportb(P_paralelo);    //lee datos filtrados a ser graficados
                 temp=(temp*(-1));
                 outportb(P_control,(0x25));   // control de las líneas para poder leer los datos


                 switch(dato)
                 {
                        case 0x51:           // derivación 1 a ser graficada
                                if(x0==7)
                                  { r1=x0;c1=temp+110;}
```

```
                                        //putpixel(x1,((temp+110)),color); //x1++;
                                        line(r1,c1,x0,(temp+110));              // dibuja la línea
                                        r1=x0;c1=temp+110;x0++;

                                                break;

                                case 0x52:     // derivación 2 ser graficada
                                        if(xy<tam & dato==0x52)     //se llena el arreglo para la autoccorrelación
                                         {                          // y poder determinar la frecuencia cardiaca
                                                y1[xy]=temp;
                                                y2[xy]=temp;
                                                xy++;
                                         }

                                        if(xy>=tam & dato==0x52)
                                         {
                                         //     autocorrelacion();                        //llama FC
                                                gprintfxy(540,40,"FC : %.2f",frec);

                                                xy=0;
                                         }

                                        if(x2==7)
                                         { r2=x2;c2=temp+236;}
                                                //putpixel(x1,((temp+110)),color); //x1++;
                                        line(r2,c2,x2,(temp+236));//}
                                        r2=x2;c2=temp+236;x2++;

                                         break;
                                case 0x53:                  // derivación 3 a graficar
                                        if(x3==7)
                                         { r3=x3;c3=temp+350;}
                                        //putpixel(x1,((temp+110)),color); //x1++;
                                        line(r3,c3,x3,(temp+350));//}
                                        r3=x3;c3=temp+350;x3++;

                                         break;
                         }
                        }
        dato=0;

        if(x0==486 || x2==486 || x3==486)
                {//cleardevice(); se limpia la pantalla y reinicializa
                x0=7,x2=7,x3=7;r1=7; grafico();}

                 }

        getch();op='\0';

 return 0;
}

int autocorrelacion()
{
        for(l=0;l<=(tam-1);l++)     //tama¤o del arreglo de autocorrelaci¢n
                 {

                        for(n=0;n<=(tam-1);n++)//tama¤o del arreglo de las se¤ales y1[n]=y2[n]
                                {
                                 v[n]=y1[n]*y2[n];  //producto de las dos se¤ales
                                 s=v[n]+s;          //suma todos los valores de n
                                }
```

```
                    r[l]=s;                 //arreglo de autocorrelacion
                    s=0;                    //se limpia suma de valores de n
                    //printf("%.2f ",r[l]);
                    for(n=0;n<=(tam-1);n++) //corrimiento a la derecha de y2[n]
                              {
                               y3[n+1]=y2[n];     //y3[n] se corre a la derecha
                               y2[n]=y3[n];       //y2[n] actualiza nuevo corrimiento
                               v[n]=0;            //se limpia v[n]=y1[n]*y2[n]
                              }
          }

for(n=1;n<=8;n++,k1++)  //k1 inicia en 1
          {
           s1=r[k1]+s1;       //s1 = suma 12 valores desde 5 hasta 17
          }

for(i=0;i<=tam;i++)
          {

                    for(n=1;n<=8;n++,k2++)  //k2 inicia en 6
                              {
                               s2=r[k2]+s2;         //s2 = suma desde 10 hasta 22
                              }

                    if (s1 >= s2)
                              {
                               if(s1 > maximo)
                                        {
                                         l=k1-8;          //4, l almacena el índice del valor máximo
                                         maximo=s1;       //máximo contiene el valor máximo
                                        }
                               if( ban == 1 )
                                         {
                                          l=(k2-4)-l;
                                          i=tam;
                                         }
                              }
                    else
                              {
                               ban=1;
                               if(s2 > maximo)
                                        {
                                         l=k2-4;
                                         maximo=s2;
                                        }
                              }
                    s1=s2;
                    s2=0;
                    k2=(k2-8)+1;
          }
          for(n=0;n<tam;n++)   //se limpian arreglos
                    {
                     r[n]=0;
                     y1[n]=0;
                     y2[n]=0;
                    } s1=0;s2=0;k1=1;k2=4;
                      ban=0;maximo=0;n=1;
                      i=0;

 frec=l;
 l=0;
```

```
            frec=1/(frec/160);   // retorna la frecuencia cardiaca

           return frec;
}

int ventana(int x1, int y1, int x2, int y2)
{

            //setfillstyle(SOLID_FILL, 7);    //estylo del relleno y color_Gris
            bar(x1,y1,x2,y2);   //Ventana

            setcolor(0);
            line(x1,y2,x2,y2);//sombra negro_Hor
            line(x2,y2,x2,y1);//sombra negro_Vert

            setcolor(15);
            line(x2,y1,x1,y1);//sombra blanco_Hor
            line(x1,y1,x1,y2);//sombra blanco_Vert
            return 0;
}

int ini_graficos(void)
{

           int gdriver = DETECT, gmode, errorcode;
           initgraph(&gdriver, &gmode, "");

           errorcode = graphresult();

           if (errorcode != grOk)  /* an error occurred */
            {
                     printf("Graphics error: %s\n", grapherrormsg(errorcode));
                     printf("Press any key to halt:");
                     getch();
                     exit(1);             /* return with error code */
            }
           return 0;
 }

int gscanfxy(int xloc, int yloc, char *fmt, ...)
            {
             va_list argptr;
             char str[BUFSIZE];
             int cnt;
             int oldx, oldy;

             oldx= xloc;
             oldy= yloc;
             moveto(xloc,yloc);
             va_start(argptr,fmt);
             ggets(str);
             cnt= vsscanf(str, fmt, argptr);
             va_end(argptr);
             moveto(oldx,oldy);
             return(cnt);
            }

int gprintfxy(int xloc, int yloc, char *fmt, ...)
            {
             va_list argptr;
             char str[BUFSIZE];
             int cnt;
```

```
        struct fillsettingstype oldfill;
        char userfillpattern[8];

        va_start(argptr,fmt);
        cnt= vsprintf(str, fmt, argptr);
        if (str[0] == NULL) return 0;

        getfillsettings(&oldfill);
        if (oldfill.pattern == USER_FILL)
                getfillpattern(userfillpattern);
        setfillstyle(SOLID_FILL,3); //relleno de cursor "_"

        bar(xloc, yloc, xloc+textwidth(str), yloc+textheight("H")*5/4);
        if (oldfill.pattern == USER_FILL)
                setfillpattern(userfillpattern,oldfill.color);
        else
                setfillstyle(oldfill.pattern, oldfill.color);
                outtextxy(xloc, yloc, str);
                va_end(argptr);
                return(cnt);
       }
int gprintf(char *fmt, ...)

       {
        va_list argptr;
        char str[BUFSIZE];
        int cnt;
        struct fillsettingstype oldfill;
        char userfillpattern[8];
        int xloc, yloc;


              va_start(argptr,fmt);
        cnt= vsprintf(str, fmt, argptr);
        if (str[0] == NULL) return 0;
        xloc= getx(); yloc= gety();
        getfillsettings(&oldfill);
        if (oldfill.pattern == USER_FILL)
                getfillpattern(userfillpattern);
        //setfillstyle(SOLID_FILL,getbkcolor());
        setfillstyle(SOLID_FILL,3);             //relleno color de letras
        bar(xloc, yloc, xloc+textwidth(str), yloc+textheight("H")*5/4);
        if (oldfill.pattern == USER_FILL)
                setfillpattern(userfillpattern,oldfill.color);
        else
                setfillstyle(oldfill.pattern, oldfill.color);
                outtext(str);
                va_end(argptr);
                return(cnt);
       }

int gputch(int c)
       {
        char buffer[2];

        sprintf(buffer,"%c",c);
        gprintf(buffer);
        return(c);
       }
char *ggets(char *buffer)
        {
        int currloc, maxchars, oldcolor;
```

```
struct viewporttype view;
char ch, charbuff[3];

buffer[0]='\0';
currloc=0;
getviewsettings(&view);
maxchars = (view.right-getx())/textwidth("M")-1;

if (maxchars <= 0) return (NULL);
//gprintfxy(getx(),gety(),"_");
gprintfxy(getx(),gety(),"");
while((ch=getch())!=CR) {
        if(ch==BS) {
                if (currloc > 0) {
                        currloc--;
                        if(currloc<=maxchars) {
                          oldcolor=getcolor();
                          setcolor(getbkcolor());

                          sprintf(charbuff,"%c",buffer[currloc]);
                          gprintfxy(getx()-textwidth(charbuff),gety(),
                                                                "%c_",buffer[currloc]);
                          setcolor(oldcolor);
                          moveto(getx()-textwidth(charbuff),gety());
                         }
                 }
                }
                else {
                  if (currloc<maxchars) {
                        oldcolor=getcolor();
                        setcolor(getbkcolor());
                        //gprintfxy(getx(),gety(),"_");
                        gprintfxy(getx(),gety(),"");
                        setcolor(oldcolor);

                        buffer[currloc]=ch;
                        gputch(ch);
                        currloc++;
                       }
                       else
                         putch(0x07);
                }
                if (currloc < maxchars)
                        //gprintfxy(getx(),gety(),"_");
                         gprintfxy(getx(),gety(),"");
        }
        if (currloc <= maxchars) {
        oldcolor= getcolor();
        setcolor(getbkcolor());
        //gprintfxy(getx(),gety(),"_");
        gprintfxy(getx(),gety(),"");
        setcolor(oldcolor);
        }
        buffer[currloc]='\0';
        return(buffer);
}
```

Referencias bibliográficas.

[1]. Anatomía Humana. Tomo II, Testut Latarjet.

[2]. Tratado de fisiología médica. Guyton Hall.

[3]. Interpretación de electrocardiogramas. Dr. Joseph Wartak, págs. 1-19 Ed. Interamericana. 1983.

[4]. Electrocardiografía clínica.

[5]. Fisiología Médica. William F. Ganong, pág. 493-510 (Circulación). Ed. Manual Moderno. 1992.

[6]. Tratamiento digital de señales. Principios, algoritmos y aplicaciones . John G. Proakis – Dimitris G. Manolakis. Ed. Prentice Hall. 3ra. Edición. 1998.

[7]. Discrete – Time Signal Processing. Alan V. Oppenheim – Ronald W. Schafer Edit. Prentice – Hall, Signal Processing Series. 1989.

[8]. Sistemas de control en Tiempo Discreto. Katsuhiko Ogata. Edit. Prentice Hall. Cap. I. Págs. 1- 20. 1996.

[9]. Matlab Help Desk. Signal Processing Toolbox Reference. 1998.

[10]. TMS320C3X Guía de Usuario para el procesamiento digital de señales Texas Instruments. 1994.

[11]. The TMS320C30 Floating Point Digital Signal Processor. Panos Papamichalis – Ray Simar, Jr. Application Report. Texas Instruments 1997.

[12]. TMS320C3x DSP Starter Kit User's Guide. Texas Instruments. 1994.

[13]. Hojas de especificaciones del convertidor A/D ADC0808/ADC0809 de 8-Bits Compatible con µ Procesadores con 8-Canales Multiplexados. 1999.

[14]. Operational Amplifiers, Design and Applications: Tobey-Graeme-Huelsman Ed. Burr-Brown – Mc Graw Hill, International Student Edition. 1971.

[15]. Amplificadores Operacionales y circuitos integrados lineales: Robert F. Coughlin – Frederick F. Driscoll. Prentice Hall Hispanoamericana, S.A. 1993.

Glosario.

Polarización: Haciendo referencia a las células, la pared interior de estas es negativo con respecto a la pared celular exterior.

Despolarización: La pared interior de las células es positivo con respecto a la pared celular exterior.

Repolarización: Movimiento de electrolitos a su estado normal

Potenciales de acción iterativos: Generación de impulsos repetitivos.

Onda P: Despolarización auricular.

Complejo QRS: Despolarización ventricular.

Onda T: Repolarización ventricular.

K^+: Símbolo químico del Potasio.

Na^+: Símbolo químico del sodio.

Nodo sinoauricular (NSA): Punto anatómico del tejido nervioso donde se genera el impulso eléctrico, que determina el ritmo del corazón.

Nodo auriculoventricular: Punto anatómico del tejido nervioso donde se recibe el impulso proveniente del NSA y lo propaga.

Artefacto: Tecnicismo médico que significa interferencia presente en la señal de electrocardiograma.

Precordial: Significa un lugar anatómico que se encuentra en la parte anterior al tórax.

CTg.: Central terminal de Goldberger. Esta central consiste en unir mediante resistencias las otras dos derivaciones. Por consiguiente, cada derivación unipolar registra una diferencia de potencial entre el miembro explorado y el potencial promedio de los otros dos miembros [3].

CTw.: Central terminal de Wilson. La central terminal de Wilson se forma uniendo las tres derivaciones mediante resistencias, de modo que su potencial combinado pueda considerarse siempre igual a cero. Por consiguiente, cada derivación precordial registra la diferencia de potencial eléctrico que va presentándose en el punto explorado [3].

Esp. IC.: Espacio intercostal.