UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA MECÁNICA

ADQUISICIÓN Y PROCESAMIENTO DE DATOS DE SENSORES USANDO TECNOLOGÍA WIRE

TESIS

PARA OPTAR EL TÍTULO PROFESIONAL DE:

INGENIERO MECATRÓNICO

NIEL RELLY SOTO CANTO

PROMOCIÓN

2006-2

LIMA - PERÚ

2007

TABLA DE CONTENIDO

PRÓLOGO

El principal objetivo de la presente tesis es la obtención y el procesamiento de datos que fueron recibidos de forma serial, por un dispositivo de adquisición de datos (DAQ), con el uso de tecnologías Wire.

Para mostrar el funcionamiento del uso de tecnología Wire se usarán ejemplos de simulación tanto en la obtención como en el procesamiento de los datos.

Para cumplir con el objetivo la presente tesis será divido en los siguientes temas:

El capítulo 1 introduce de manera general el contenido de la tesis en el cual se especifica el propósito.

El capítulo 2 describe los antecedentes en tecnología wire y se detalla la evolución de dicha tecnología con el uso que se da en Softwire y se motivará usar este producto de bajo coste y alto rendimiento para el procesamiento de datos.

El capítulo 3 presenta los antecedentes de adquisición de datos y los fabricantes de dichos dispositivos como también sus características y las ventajas de las DAQ de MCC (Meausrement Computing).

El capítulo 4 trata de las características del firmware de un dispositivo de adquisición de adquisición de datos donde se definirán los límites de una tarjeta, ya sea frecuencia de muestreo y velocidad de transmisión serial.

El capítulo 5 muestra los módulos de procesamiento de datos con tecnología wire detallados mediante esquemas en funcionamiento de los periféricos de una DAQ.

El capítulo 6 describe a VisualNiel2 como utilización de tecnología wire y tendrá la función de mostrar la recepción y transmisión de las señales provenientes de los sensores a través de un DAQ.

CAPÍTULO I

INTRODUCCIÓN

La presente tesis tiene el objetivo de usar tecnología Wire para que un equipo sea capaz de adquirir cualquier tipo se señales en un determinado rango de voltaje, transmitirlas en formato digital vía serial para luego procesarlas y obtener la información en la pantalla de un computador.

Las señales analógicas o digitales del computador serán transmitidas a través del Puerto serial, paralelo o Internet, utilizando protocolos ya establecidos como RS232-RS485 (serial) y TCP/IP (Internet), de manera que en el extremo receptor esta información pueda ser procesada y visualizada en la pantalla del otro computador conectado a Internet y de esa manera lograr control a distancia del equipo que adquiere y procesa datos.

A este tipo de dispositivos se les llama Tarjetas de Adquisición de Datos, por este motivo para adquisición y procesamiento se tomará como ejemplo la Tarjeta ADN2006 (Tarjeta de Adquisición de Datos de Niel), dicho término se usará en los temas que lo soliciten.

La tarjeta ADN2006 surge con la idea de tener un ejemplo peruano que use tecnología Wire con la finalidad de ser aplicado en cualquier área o proyecto de Ingeniería tales como el de adquirir y procesar datos de sensores, es decir diseñar al creador de aplicaciones, para de esa manera acelerar y optimizar la curva de productividad de cualquier proyecto y reducir ese tiempo de producción al re-usar código.

Lo anterior son características de la tecnología Wire que basa su concepto en solo conectar bloques para obtener la aplicación deseada y para este proyecto se usarán Proteus (Wire para adquisición) , Sowftwire (Wire para procesamiento) y un ejemplo de esa tecnología que será VisualNiel2.

En los antecedentes en tecnología wire se detallará la evolución de dicha tecnología como también el uso que se da de ello en Softwire que es el encargado del procesamiento de datos con la manipulación de una variedad de componentes que sólo necesitan ser conectados para obtener una aplicación tales como el de adquirir datos de sensores.

En el tema de los antecedentes de adquisición de datos se tratará de los fabricantes dichos dispositivos y también sus características, las ventajas de las DAQ de MCC (Meausrement Computing) como los pioneros del uso de tecnología wire. También se describirá el acondicionamiento de señal pro medio del filtrado y la amplificación a voltajes especificados por la DAQ.

Las características del firmware de un dispositivo de adquisición serán analizadas con cuidado ya que de esto dependerá la rapidez y el correcto entendimiento del procesamiento interno de las DAQ y acá se definirán los límites de una tarjeta, ya sea frecuencia de muestreo, velocidad de transmisión serial, ciclo de trabajo de módulos de PWM (modulación por ancho de pulso) y otras opciones.

Los módulos de procesamiento de datos con tecnología wire se encargarán de realizar un esquema para puesta en funcionamiento de los periféricos de una DAQ, donde lo único que se le pasará es el código hexadecimal o binario y para ello se realizará la simulación de adquisición y procesamiento de datos en PC.

La etapa Visualniel2 como utilización de tecnología wire tendrá la función de mostrar la recepción y transmisión de las señales provenientes de los sensores a través de un DAQ. Se adquiere datos, luego se ordena y procesa para mostrarlos en los componentes de VisualNiel2 que será muy intuitivo y amigable para como todas las tecnologías Wire.

Por lo tanto se usa tecnología wire tanto para la adquisición con los dispositivos DAQ como en el procesamiento de datos de los sensores y para ejemplo de ello se ve el Software VisualNiel2 que además tendrá acceso a programas extras aplicados a la ingeniería (Generador de PID analógico,

Generador de ecuaciones diferenciales de PID Discreto) los cuales desarrollé durante mi permanencia en la UNI. También se tendrá en cuenta que fue necesario manejar herramientas de última tecnología o las versiones más avanzadas tales como la tecnología wire y la librería universal para todo tipo de lenguajes, es el caso de Softwire que se trata a continuación.

CAPÍTULO II

ANTECEDENTES EN TECNOLOGÍA WIRE

En el mundo de la evolución del software se destaca la técnica de la programación estructurada que es usada en lenguajes de alto nivel como Pascal y C++ o interpretadores como el Visual Basic.

Teniendo en cuenta que se elige la técnica de la programación estructurada, dentro de estas existe evolución en el modo de crear software y que esto no solo sea una actividad de estrictos programadores, sino que se de saltos y mediante la reutilización de código se acelere la creación de programas dedicados al área de ingeniería.

De esta manera pasamos desde el nacimiento del ANSI C, para luego llegar a la programación orientada a objetos con C++ y dentro de esta última nace la tecnología wire, que combina la facilidad de hacer programas con sólo arrastrar, soltar y cablear.

2.1 Tecnologías de codificación antiguas

- **Programación Modular**

Podemos definir la programación modular como una serie de técnicas para descomponer un programa grande y/o complejo, en una serie de unidades más pequeñas y manejables denominadas módulos, cada una de las cuales puede, diseñarse, codificarse y probarse individualmente. Este es un método de concepción y diseño de programas basado en la descomposición del programa completo en partes que responden, generalmente a funciones específicas del programa; cada función podrá codificarse y probarse independientemente constituyendo un módulo.

- **Programación Estructurada**

En este tipo de programación se busca que la secuencia espacial de instrucciones guarde paralelismo con la secuencia de ejecución de las instrucciones, es decir no debe haber diferencia entre el orden en el listado de las instrucciones y el orden en que son ejecutadas, por lo cual se descarta el empleo de la instrucción goto. Con este tipo de programación se puede leer el programa desde el principio hasta el fin de modo continuo, facilitando con ello las pruebas, ya que existe una relación más cercana entre la descripción estática del programa y su ejecución dinámica.

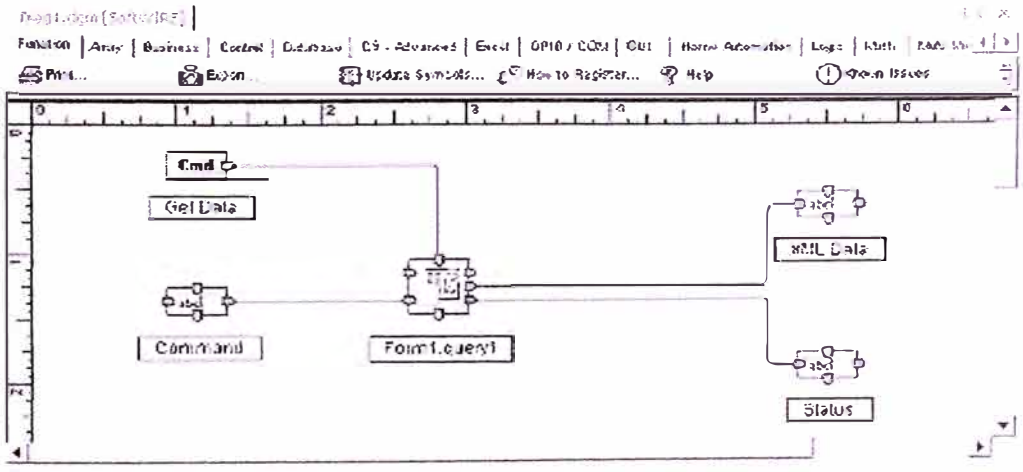
- **Programación Concurrente**

Un sistema con una unidad de proceso puede realizar solamente ejecución de instrucciones en manera secuencial, es decir en cualquier instante de

tiempo solo se ejecuta una instrucción de un proceso a la vez. Dentro de esta perspectiva se puede decir que un programa es concurrente cuando permite ser invocado en forma simultánea (real o aparente) por varios procesos, es decir la programación permite conmutar la ejecución de código de un proceso a otro, sin que el primero haya finalizado. De esta manera los programas concurrentes pueden considerarse construidos por procesos secuenciales que pueden ser ejecutados en forma simultánea en un mismo microprocesador.

2.2 Principios del uso de tecnología Wire

La gran pregunta es ¿Qué es tecnología Wire?, es la técnica más reciente que facilita la programación y lo convierte en arrastrar, soltar y cablear para de esa manera aplicar nuestros conocimientos de manera rápida y eficiente con solo tener el concepto de lo que se quiere hacer tal como se ve en la figura siguiente.



(Figura 2.2.1) Programación con tecnología Wire

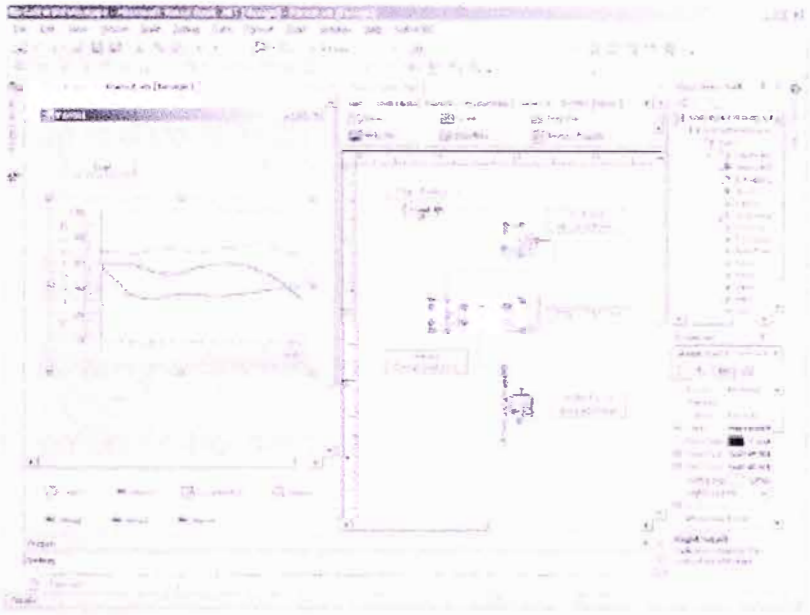
Hay empresas que le ponen otro nombre como es el caso del Simulink de MatLab que le denomina "programación con diagrama de bloques", también tenemos el LabView que le llama "programación gráfica" y en programas de diseño de esquemas y simulación electrónica tal es el caso del afamado Proteus.

Por otra parte están los no tan desarrollados en el cable virtual en el cual uno puede arrastrar y soltar pero el cable tiene que ser código, dentro de estos encontramos a los mal llamados "lenguajes de programación visual" tales como el Visual C++, Visual Basic, que en realidad sólo son un IDE (interfaz de desarrollo integrado) que se basan en lenguaje C++ y Basic respectivamente.

El gran problema de los programas como MatLab, LabView y otros es el de no generar aplicaciones independientes en su uso formal, con lo cual nos mantendrá amarrados a la fuente por siempre. Aunque cabe mencionar que el desarrollador de ADN2006 ha tenido pequeños éxitos al respecto ya que se ha logado hacer programas con GUI (Interfaz Gráfica de usuario) en Matlab y compilarlo a EXE (Ejecutable) , y en el caso de LabView (sólo basta tener el compilador de LabView).

Ante estos problemas surge la necesidad de que esta tesis de solución a ello motivando a que se desarrolle aplicaciones de ingeniería de manera rápida y

que luego serán independiente del programa en el que se generó, y con ello nos estamos refiriendo a Softwire, el programa que cambio la vida a países Europeos y Asiáticos.



(Figura 2.2.2) Programa innovador SoftWire

2.3 Utilización de Softwire

Este programa de la empresa Measurement Computing que en los últimos años hizo tambalear los cimientos del gigante National Instruments, cuyo fundador y presidente es James Truchard , se basa en la tecnología wire pero que además te da la posibilidad de utilizarlo con cualquier otro lenguaje de programación.

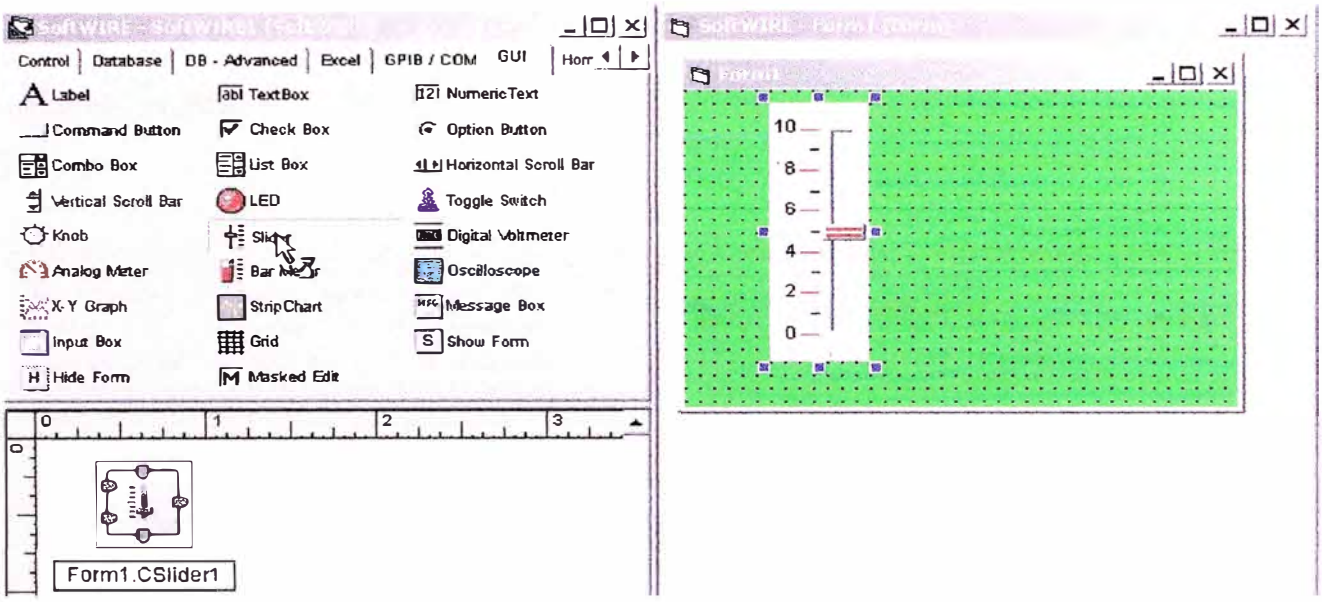
Ben Bailey, presidente y fundador de Measurement Computing que funciona, desde Middleboro, Massachussets es una empresa pionera y líder en tarjetas de adquisición de datos de bajo coste para interfaces ISA, PCI y USB para ordenadores personales. La compañía, fundada en 1989, tiene más de 70 empleados y distribuye sus productos en todo el mundo mediante venta directa, una red de distribución y a través de Internet.

Esta empresa es famosa por la creación del software innovador, SoftWire, el único sistema de programación gráfica para Visual Studio, y cualquier otro IDE de programación como Borland, Measurement Computing apuesta por la satisfacción completa de sus clientes, una gran competitividad, un estricto control de costes y campañas agresivas de bajada de precios. Actualmente su producto SoftWire también funciona con el software NI LabVIEW.

Por lo tanto SoftWire es el sistema de programación con tecnología wire para ser usado ahora en visual Studio Net, que permite crear aplicaciones DAQ en minutos sin tener que aprender un lenguaje de programación, y los proyectos pueden tener un plus de código para usuarios con uso de programación básica.

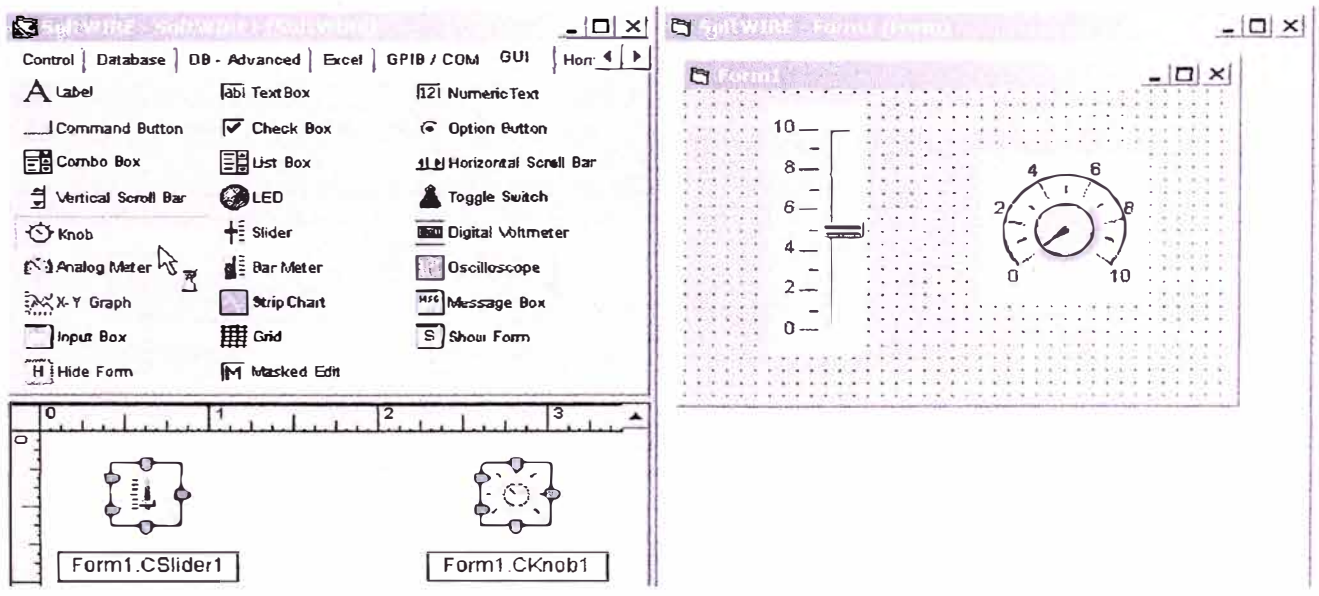
A continuación les muestra mi ejemplo básico en 5 pasos

1) Seleccione el diseñador de código de SoftWIRE. Desde los objetos disponibles escoger un Slider (deslizador) y hacer clic.



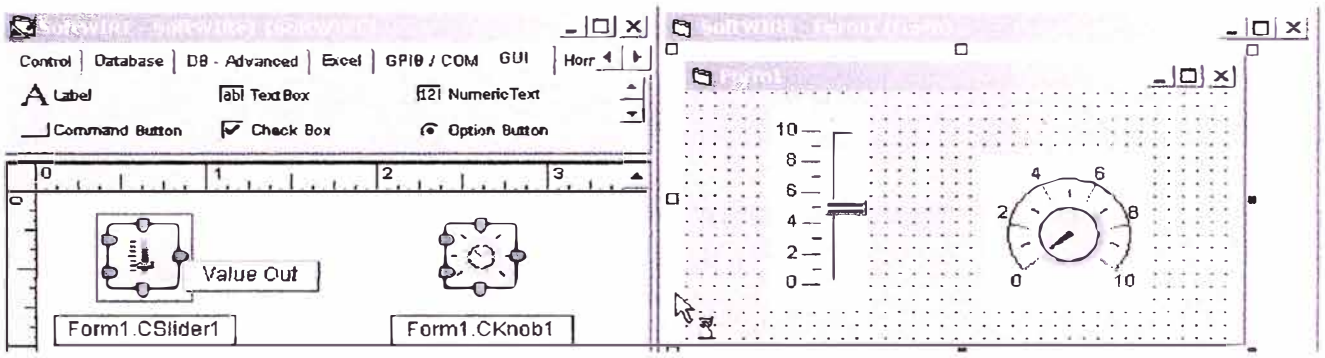
(Figura 2.3.1) Diseñador de código SoftWire y Diseñador de Formulario

2) Coloque un Knob (perilla o potenciómetro) en el diseñador de SoftWIRE



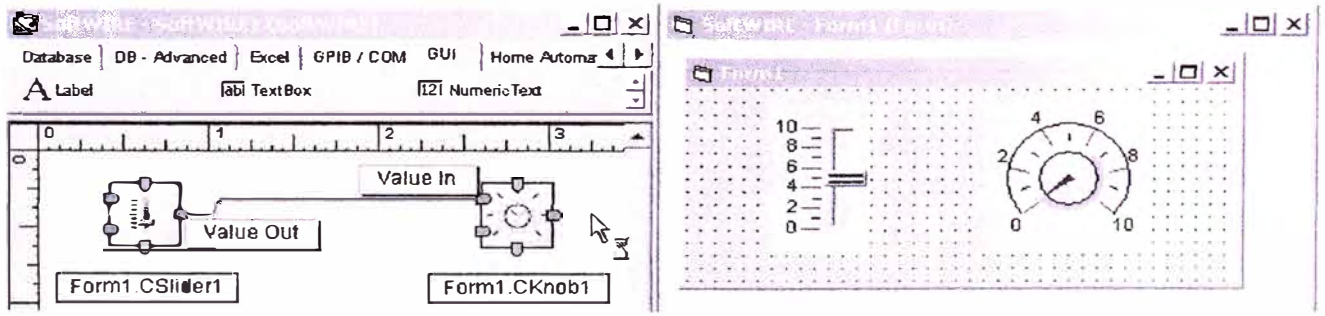
(Figura 2.3.2) Arrastrar y soltar en SoftWire

3) Coloque el mouse en el pin del Slider y aparece el mensaje indicando valor de salida.



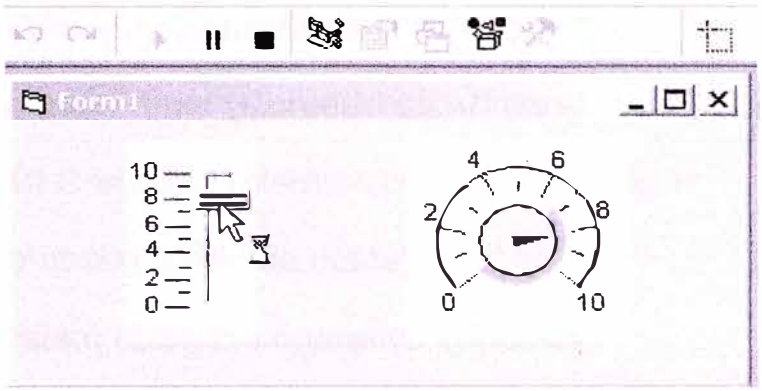
(Figura 2.3.3) Visualizador de mensaje de ayuda rápida en SoftWire

4) Arrastre el cable desde el pin del Slider hasta el pin de valor de entrada del Know.



(Figura 2.3.4) Cablear en SoftWire

5) Solo presione el botón Run del Visual Basic y verá el funcionamiento del programa



(Figura 2.3.5) Ejecución del programa hecho en SoftWire

2.4 Componentes Softwire para uso en ingeniería

SoftWIRE 6.0 ofrece más de 250 controles potentes que incrementan funcionalidad y facilidad en el uso de aplicaciones de ingeniería, incluyendo más de 100 controles de análisis.

(Figura 2.4.1) Multitud de componentes en SoftWire

Se tiene en total 23 Categorías que son las siguientes:

01) Controles Analog I/O (Analógicos de entrada y salida)

02) Controles Temperatura (Temperatura)

03) Controles Digital I/O (Digitales de entrada y salida)

04) Controles Counter/Timer (Contadores y timers)

05) Controles GUI Controls (Interfase gráfica de usuario)

06) Controles Database (Base de datos)

07) Controles System (Sistema operativo Windows)

08) Controles Network (Redes e Internet)

09) Controles Programming (Estructuras de programación)

10) Controles Math (Matemáticos)

11) Controles Statistics (Estadísticos)

12) Controles Excel (Manipulación de MS Excel)

13) Controles Array (Vectores y matrices)

14) Controles Logic (Lógicos)

15) Controles Serial IO (Serial de entrada y salida)

16) Controles Windowing (Ventanas para filtros)

17) Controles Signal Processing (Procesamiento de señales)

18) Controles Transform Function (Funciones de transformadas)

19) Controles IIR Filter (Filtro de respuesta al impulso infinita)

20) Controles FIR Filter (Filtro de respuesta al impulso finita)

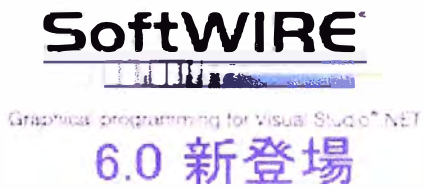
21) Controles Spectral Measurement (Mediciones espectrales)

22) Controles Curve Fitting (Ajuste de curvas)

23) Controles Signal Generation (Generación de señales)

Debido a que este software es usado mayormente en Europa y Asia, no se encuentra material disponible en castellano por lo que a continuación haré

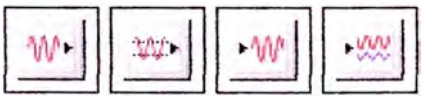
sus descripciones y de ésta manera se motiva a que se realizan aplicaciones de ingeniería de manera rápida, que es uno de los fines de esta tesis, con solo basta tener el concepto de lo que se quiere hacer y mejor si se maneja bien la teoría sin necesidad de dominar aspectos de programación (solo basta arrastrar, soltar y cablear).



(Figura 2.4.2)Muestra el logo de Softwire con uso en países asiáticos

Además estos controles pueden también ser usados con la tarjeta ADN2006.

01) Controles Analog I/O (5)



AIRead: Lee un canal de entrada analógica singular

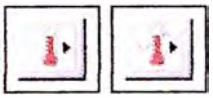
AIScan: Escanea un rango de canales de entradas analógicas

AITrigger: Lee un canal de entrada analógico singular y espera hasta que su valor excede un umbral especificado

AOUpdate: Escribe un valor singular a un canal de salida analógica

AOScan: Escribe un conjunto de valores a un rango de canales de salidas analógicas

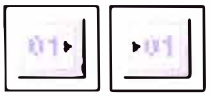
02) Controles Temperatura (2)



TempIn: Lee un valor de temperatura desde un canal de entrada de temperatura singular

TempInScan: Lee valores de temperatura desde un rango de canales de entrada de temperatura

03) Controles Digital I/O



DIReadByte: Lee el valor de un puerto digital IO

DOWriteByte: Escribe el valor especificado a un puerto digital IO

DIReadBit: Lee el valor de un bit en un puerto digital I/O

DOWriteBit: Escribe el valor especificado en un bit del puerto digital I/O

04) Controles Counter/Timer (6)

CTR8254Configure: Configura un contador en tiempo real 8254

CTR9513Configure: Configura un contador en tiempo real 9513

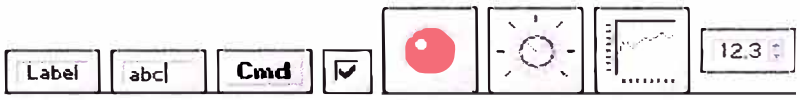
CTR9513Initialize: Ejecuta un nivel de inicialización para un contador 9513

CTRLoadValue: Carga registros especificados en un contador

CTRReadValue: Lee el valor de un contador

CTR9513ReadFrequency: Mide la frecuencia de un contador de entrada 9513

05) Controles GUI Controls (23)



Label: Muestra texto que no puede ser escrito en tiempo de ejecución (cuando su programa está corriendo).

TextBox: Encapsula a un control Edit para mostrar y habilitar al usuario para editar una cadena de texto.

Numeric TextBox: Permite al usuario entrar valores numéricos y mostrar valores numéricos que pueden ser escritos en tiempo de ejecución.

Command Button: Habilita al usuario ejecutar orden, interrumpe o finaliza un proceso clickeando en el botón en tiempo de ejecución.

CheckBox: Habilita al usuario entrar y mostrar valores que tienen configuración de True/False, Yes/No, On/Off, etc.

Radio Button: Habilita al usuario entrar y mostrar valores que tienen configuración de True/False, Yes/No, On/Off, etc. dentro de un grupo donde un solo ítem puede ser On a la vez.

Combo Box: Permite al usuario seleccionar un valor clickeando en el valor de una lista drop-down (desplegable), o seleccionar un valor escrito en la porción de texto del Combo Box.

List Box: Habilita al usuario seleccionar un valor clickeando en ello desde una lista de valores.

Horizontal Scrollbar: Permite un conjunto de valores para ser desplazado de izquierda a derecha.

Vertical Scrollbar: Permite un conjunto de valores para ser desplazado de arriba abajo

Strip Chart: Dinámicamente muestra data en el eje X y de uno para 8 ejes Y. Caja eje puede mostrar de1 a 32 plots de datos.

LED: Representa el estado de una variable binaria cambiando su color entre estados OFF y ON.

Toggle Switch: Habilita al usuario cambiar la actividad de la aplicación de on a off clickeando el botón, presionando una tecla o escribiendo código para causar una acción on/off.

Know: Un control de entrada semejando a un potenciómetro mecánico el cual usted puede rotar.

Analog Meter: Muestra gráficamente un valor analógico ubicando un puntero junto con una escala circular.

Bar Meter: Muestra gráficamente un valor analógico ubicando un puntero junto con una escala lineal.

Slider: Un control de entrada el cual puede ser movido en una dirección lineal (horizontal o vertical) por acción del usuario o programando usando un código wire.

XY Graph: Muestra desde 1 a 32 conjuntos estáticos de X-Y valores o arrays.

Bar Chart: Muestra desde 1-to-32 conjuntos estáticos de X-Y Y valores o arrays, en un forma de gráfico de barras.

Masked Edit: Habilita al usuario editar un formato de texto. La máscara permite al creador de la aplicación restringir la entrada.

DateTimePicker: Permite al usuario seleccionar fecha y hora, y mostrar eso en el formato especificado.

NumericUpDown: Contiene un valor numérico que puede avanzar o descender clickeando los botones de up y down del control.

Status Bar: Provee hasta 4 paneles de texto de estados que pueden ser conectados al botón de una aplicación.

06) Controles Database (10)

Read from Database: Abre una base de datos y lee el valor de una tabla, registro o campo especificado.

Write to Database: Abre una base de datos y escribe el valor de una tabla, registro o campo especificado.

FindData: Busca una fuente de dato para el valor especificado.

StoredProcedure: Ejecuta un procedimiento almacenado de una base de datos.

Database ListBox: Permite al usuario acceder y mostrar dato directamente desde un conjunto de registros en uno o más tablas de base de datos.

Database ComboBox: Permite al usuario acceder y mostrar dato directamente desde un conjunto de registros en uno o más tablas de base de datos.

Query Database: Permite al usuario crear y ejecutar una pregunta.

Database Gris: Permite al usuario acceder y mostrar dato directamente desde un conjunto de registros en uno o más tablas de base de datos.

Filter DataTable: Filtra un recordset en criterio del usuario.

Database TextBox: Permite al usuario editar un campo del recordset capturando entradas en un edit box y escribiendo al recordset.

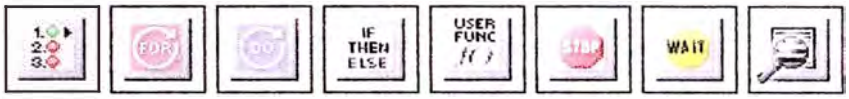
07) Controles System

Who Am I, Timer, Count, Date/Time Function, Read from File, Write to File. Execute Program, Error Catcher, Validator, FileSystemWatcher, Beeper. CopyFile, DeleteFile, MoveFile, OpenFileDialog, SaveFileDialog PeriodicTimerEx.

08) Controles Network (4)

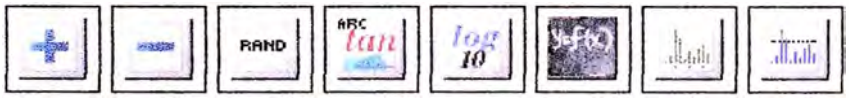
Send E-mail, Broadcast, Tuner, WebService.

09) Controles Programming



Start, Sequencer, String Builder, InString, Trim String, Sub String, For Loop, Do Loop, If Then Else, User Function, Stop Program, Wait, Switch, Lookup,

10) Controles Math



Add, Subtract, Multiply, Divide, Modulus, Not Equal To, Greater Than, Greater Than or Equal To, Equality, Less Than or Equal To, Less Than, Random, Constant, MathConstants, Variable, Statistics, Histogram, AbsoluteValue, Round, Ceiling, Floor, Max, Min, Degrees to Radians, RadiansToDegrees, ArcCosine, ArcSine, ArcTangent, Cosine, Sine, Tangent, Square Root, Exponent, Log, Log10, LogX, Power, Formula, Function Generador, PeakDetector, ThresholdPeakDetector.

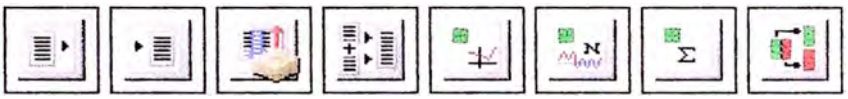
11) Controles Statistics

Mean, Median, Mode, Moment, RootMeanSquare.

12) Controles Excel

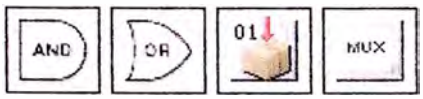
Excel Read, Excel Write.

13) Controles Array



GetElement, SetElement, AppendArray, InsertArray, UnpackArray, SubArray, ConcatArray, GetIndex, Collector, GetRowCol, ArraySize, ArrayClip, CopyColumn, CopyRow, ArrayMaxMin, ArrayLinearScale, ArrayPolynomialScale, ArrayNormalize, ArrayQuickScale, ArrayScale, ArrayShift, ArraySum, SplitColumns,

14) Controles Logic

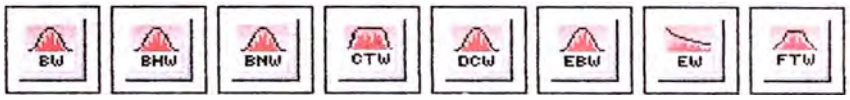


And, Nand, Or, Xor, Nor, Not, BitPack, BitUnpack, Mux, Demux.

15) Controles Serial IO

SerialOpen, SerialClose, SerialIn, SerialOut,

16) Controles Windowing



ScaledWindow, BlackmanWindow, BlackmanHarrisWindow, BlackmanNuttalWindow, CosineTaperedWindow, DolphChebyshevWindow, ExactBlackmanWindow, ExponentialWindow, FlatTopWindow, ForceWindow, GaussWindow, GeneralCosineWindow, HammingWindow, HanningWindow, KaiserWindow, TriangularWindow,

17) Controles Signal Processing

AutoCorrelate, Convolve, CrossCorrelate, Decimate, Deconvolve, Differentiate,

Integrate: Calcula la integración discreta de una seña muestreada usando la Regla de Simpson.

PulseParameters, UnwrapPhase.

18) Controles Transform Function

CrossSpectrum: Calcula en doble-lado la potencia cruzada del espectro de potencia de las entradas array.

FastHartley: Calcula la Transformada rapida de Hartley del array de entrada .

FastHilbert: Calcula la Transformada rapida de Hilbert del array de entrada.

ComplexFFT:Calcula la Transformada rapida de Fourier Compleja(FFT).

RealFFT:Calcula la Transformada rapida de Fourier Real (FFT)

ImpulseResponse, InverseFastHartley, InverseFastHilbert, InverseFFT, InverseRealFFT, PowerSpectrum.

19) Controles IIR Filter (20)

BesselBandpassFilter: Implementa un filtro IIR Bessell Bandpass.

BesselBandstopFilter: Implementa un filtro IIR Bessell Bandstop.

BesselHighpassFilter: Implementa un filtro IIR Bessell Highpass.

BesselLowpassFilter: Implementa un filtro IIR Bessell Lowpass.

ButterworthBandpassFilter: Implementa un filtro IIR Butterworth Bandpass.

ButterworthBandstopFilter: Implementa un filtro IIR Butterworth Bandstop.

ButterworthHighpassFilter: Implementa un filtro IIR Butterworth Highpass.

ButterworthLowpassFilter: Implementa un filtro IIR Butterworth Lowpass.

ChebyshevBandpassFilter: Implementa un filtro IIR Chebyshev Bandpass.

ChebyshevBandstopFilter: Implementa un filtro IIR Chebyshev Bandstop.

ChebyshevHighpassFilter: Implementa un filtro IIR Chebyshev Highpass.

ChebyshevLowpassFilter: Implementa un filtro IIR Chebyshev Lowpass.

EllipticBandpassFilter: Implementa un filtro IIR Elliptic Bandpass.

EllipticBandstopFilter: Implementa un filtro IIR Elliptic Bandstop.

EllipticHighpassFilter: Implementa un filtro IIR Elliptic Highpass.

EllipticLowpassFilter: Implementa un filtro IIR Elliptic Lowpass.

InverseChebyshevBandpassFilter:Implementa un filtro IIR InverseChebyshev Bandpass.

InverseChebyshevBandstopFilter: Implementa un filtro IIR InverseChebyshev Bandstop.

InverseChebyshevHighpassFilter: Implementa un filtro IIR InverseChebyshev Highpass.

InverseChebyshevLowpassFilter: Implementa un filtro IIR InverseChebyshev Lowpass.

20) Controles FIR Filter (12)

EquiRippleBandpassFilter: Implementa un filtro FIR EquiRipple Bandpass.

EquiRippleBandstopFilter: Implementa un filtro FIR EquiRipple Bandstop.

EquiRippleHighpassFilter: Implementa un filtro FIR EquiRipple Highpas.

EquiRippleLowpassFilter: Implementa un filtro FIR EquiRipple Lowpass.

KaiserWindowFIRBandpassFilter: Implementa un filtro FIR de ventana Kaiser Bandpass.

KaiserWindowFIRBandstopFilter: Implementa un filtro FIR de ventana Kaiser Bandstop.

KaiserWindowFIRHighpassFilter: Implementa un filtro FIR de ventana Kaiser Highpass.

KaiserWindowFIRLowpassFilter: Implementa un filtro FIR de ventana Kaiser Lowpass.

WindowedFIRBandpassFilter: Implementa un filtro FIR ventanado Bandpass.

WindowedFIRBandstopFilter: Implementa un filtro FIR ventanado Bandstop.

WindowedFIRHighpassFilter: Implementa un filtro FIR ventanado Highpass.

WindowedFIRLowpassFilter: Implementa un filtro FIR ventanado Lowpass.

21) Controles Spectral Measurement (8)

ACDCEstimator: Estima niveles de señal de entrada AC y DC.

AmplitudePhaseSpectrum: Calcula el espectro de amplitude y fase de una señal en tiempo real.

AutoPowerSpectrum: Calcula el espectro de auto power de una señal en tiempo real.

CrossPowerSpectrum: Calcula el espectro de cross power de 2 señales en tiempo real.

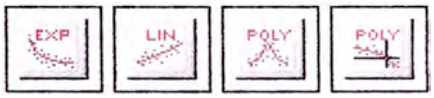
HarmonicAnalyzer: Calcula los components armónicos y fundamentales (amplitude y frecuencia) presentes en el espectro auto power de entrada.

SpectrumUnitConversion: Convierte la potencia, amplitud, o ganancia (Amplitud de ratio) espectral para alternar incluyendo formatos Log (decibel y dbm) y densidad espectral.

Transfer Function: Calcula la función de transferencia (respuesta en frecuencia) de una red bajo prueba.

PowerFrequencyEstimate: Calcula la potencia y frecuencia estimada alrededor de un pico en el espectro de potencia en tiempo real.

22) Controles Curve Fitting (6)



ExponentialFit: Calcula los valores de una curva exponencial de un conjunto de coeficientes, amplitudes y amortiguaciones exponenciales, el cual describe la curva exponencial que representa mejor el conjunto de datos entrados usando la solución de mínimos cuadrados.

LinearFit: Calcula los valores lineales el cual describe la línea que representa mejor el conjunto de datos entrados usando la solución de mínimos cuadrados

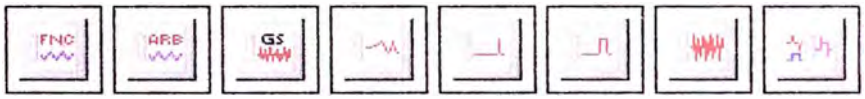
PolynomialFit: Calcula los valores de una curva polinomial y los coeficientes ajustados del polinomio, el cual describe el polinomio que representa el conjunto de datos entrados.

PolynomialInterpolation: Interpola y en x usando una función polinomial.

RationalInterpolation: Interpola y en x usando una función racional.

SplineInterpolation: Calcula el valor interpolado y en el definido valor x para el único polinomio pasando a través de los puntos (InputXData, inputYData) usando una interpolación de spline cúbico.

23) Controles Signal Generation (11)



FunctionGenerator: Genera una forma de onda usando parámetros.

ArbitrarySignalGenerator: Genera una señal arbitraria.

GaussianNoiseGenerator: Genera un modelo de ruido Gausiano.

ChirpGenerator: Genera un modelo Chirp.

ImpulseGenerator: Genera una forma de onda con un modelo de impulso.

PulseGenerator: Genera una forma de onda con un modelo de pulso.

RampGenerator: Genera una forma de onda con un modelo de rampa.

SincGenerator: Genera una forma de onda con un modelo de sinc.

UniformNoiseGenerator: Genera una señal de ruido de distribución uniforme entre 0.0 y 1.0.

WhiteNoiseGenerator: Genera una señal de ruido blanco.

CompositeSignalGenerator: Genera una señal compuesta de hasta 5 señales de entradas separadas.

CAPÍTULO III

ANTECEDENTES DE DISPOSITIVOS DE ADQUISICIÓN DE DATOS

En el campo de las DAQ existen empresas con un largo recorrido en la fabricación de hardware para diferentes buses de PC internos, como también puertos externos.

3.1 Fabricantes de tarjetas de adquisición de datos

- **Computer Boards**

Es llamada también Measurement Computing, fabrica una serie de productos A/D, D/A, de I (entrada)/ O (salida) a muy bajo precio. El Hardware y Software son compatibles con otros productos y son numerados semejantemente a otros fabricantes populares, notables como National Instruments y Keithley.

Measurement Computing es una empresa pionera y líder en tarjetas de adquisición de datos de bajo coste para interfaces ISA, PCI y USB para ordenadores personales. La compañía, fundada en 1989 por Ben Bailey,

tiene más de 70 empleados y distribuye sus productos en todo el mundo mediante venta directa, una red de distribución y a través de Internet. Conocida por el software innovador, como SoftWIRE, el único sistema de programación gráfica para Visual Studio. Measurement Computing apuesta por la satisfacción completa de los clientes, una gran competitividad, un estricto control de costes y campañas agresivas de bajada de precios.

- **National Instruments**

Son bien conocidos por su línea extensa de productos de software para adquisición de los datos. Ellos también tienen una variedad de hardware que distribuyen desde la sede en Austin, Texas, National Instruments tiene más de 3400 empleados y cuenta con operaciones directas en 40 países.

James Truchard, presidente y cofundador de National Instruments (www.ni.com) que es una empresa pionera y líder en la tecnología de la instrumentación virtual, ha dado un concepto revolucionario durante más de veinte años, giró el enfoque que ingenieros y científicos tienen de las aplicaciones de medida y automatización, tanto en el sector industrial como académico o gubernamental. Aprovechando la potencia de la computadora y otras tecnologías informáticas, la instrumentación virtual aumenta la productividad y reduce los costes de clientes de todo el mundo con software fácil de integrar, como el entorno de desarrollo gráfico LabVIEW, y hardware modular, como los módulos para adquisición de datos, control de instrumentos y visión artificial.

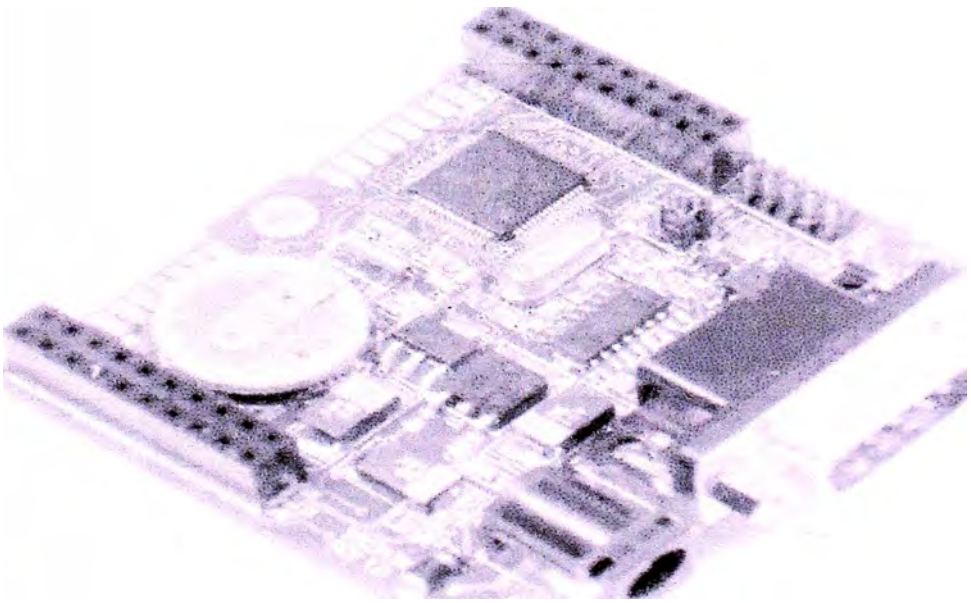
- **Keithley**

Es uno de los líderes originales en productos industriales de adquisición de datos bajo buses de PC (PCI e ISA).

- **Data Translation**

Tiene una serie de tarjetas de buses PCI y es acompañando de drivers con el software.

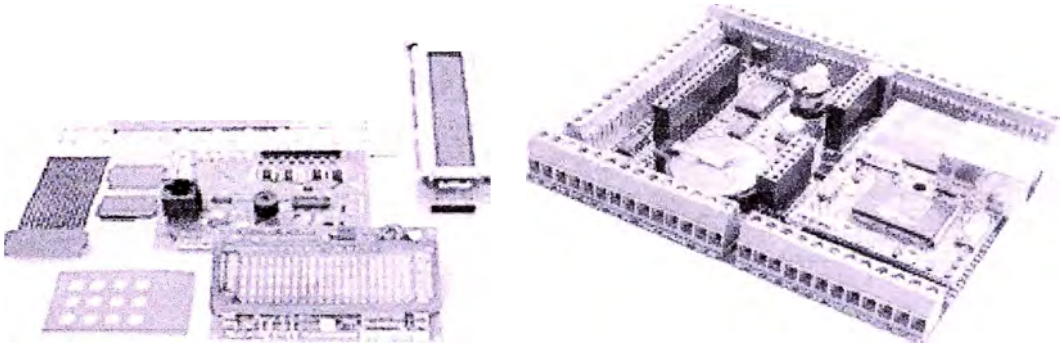
CyberResearch , Industrial Computer Source, Advantech , Acquisition Control Communications Engineering, DATEL , Intelligent Instrumentation.
Son fabricantes y vendedores de hardware de calidad industrial y software para la adquisición de los datos y mando. Algunos de los productos tienden a tener un mayor rango de extremo que otros fabricantes.



(Figura 3.1.1) DAQ serial de otros fabricantes

IOTech , Analogic , Omega Engineering, SuperLogics , United Electronic Industries, Innovative Integration , CONTEC Microelectronics.

Estos fabricantes tienen varios conversores D/A, A/D e I/O digital en tarjeta para PCs. PLCs , otros usan un DSP incluido para la adquisición de los datos de gran velocidad y puede mantener una solución comparativamente económica en algunas aplicaciones.



(Figura 3.1.2) Módulos de venta de otros fabricantes

3.2 Ventajas y características de las DAQ de MCC

(Figura 3.2.1) Sede de MCC en Middleboro, Massachussets

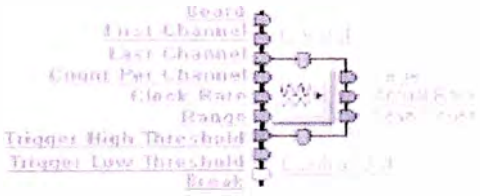
Measurement Computing Corporation (MCC) ha sido una empresa líder y pionera en adquisición de datos de bajo coste que funciona, desde Middleboro, Massachussets, y tiene grandes ventajas respecto a sus competidores que a continuación detallamos

-Todos sus productos funcionan con el software NI LabVIEW, líder en el sector.

-Tienen una gran cantidad de componentes de un fácil manejo en su software "Softwire" para adquisición de datos con sus DAQ.

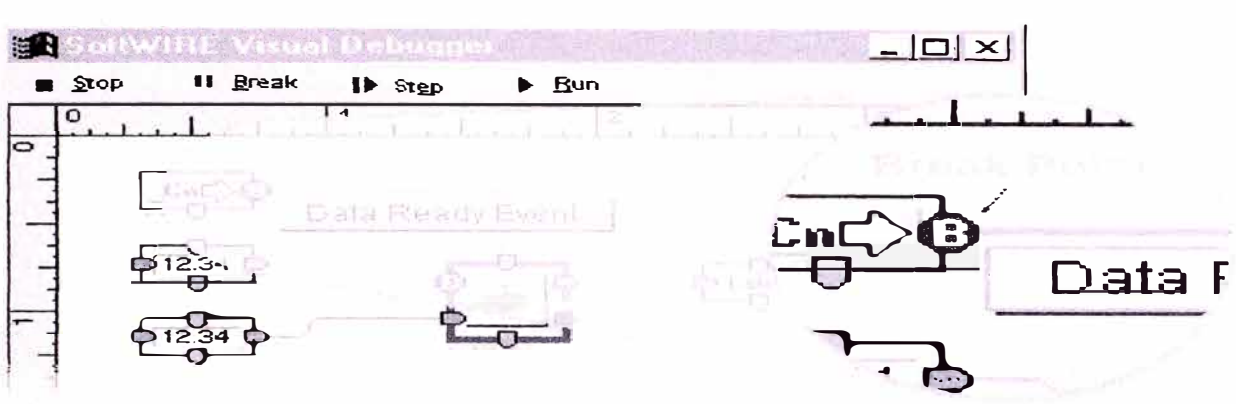
(Figura 3.2.2) Componentes SoftWire

-Sus componentes fueron diseñados intuitivamente mostrando ayuda con hints (pequeño cuadrado de ayuda que puede mostrar la funciones para conectar el cable)



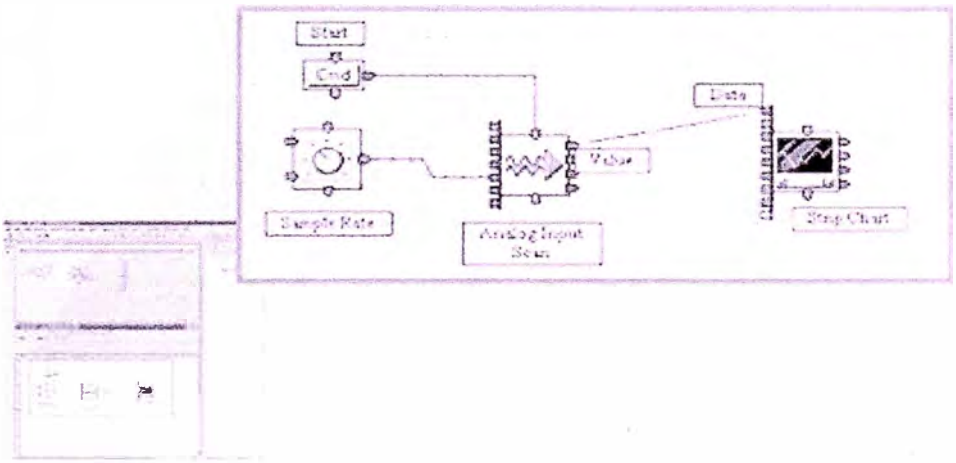
(Figura 3.2.3) Ayudas contextuales o hints

-Tiene un depurador de errores de conexión con manejo de puntos de ruptura (Break points) que permite correr el programa sólo en partes del error.



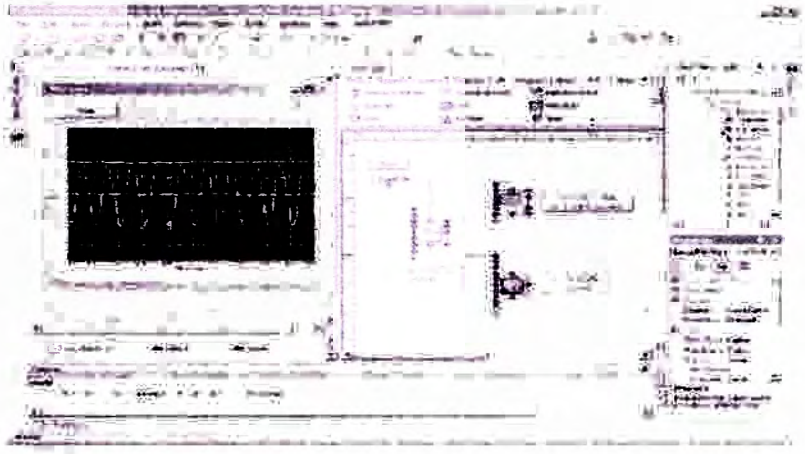
(Figura 3.2.4) Depurador de errores

-Su diseñador de código tiene una interfaz de usuario más intuitivo que sus adversarios como Labview.



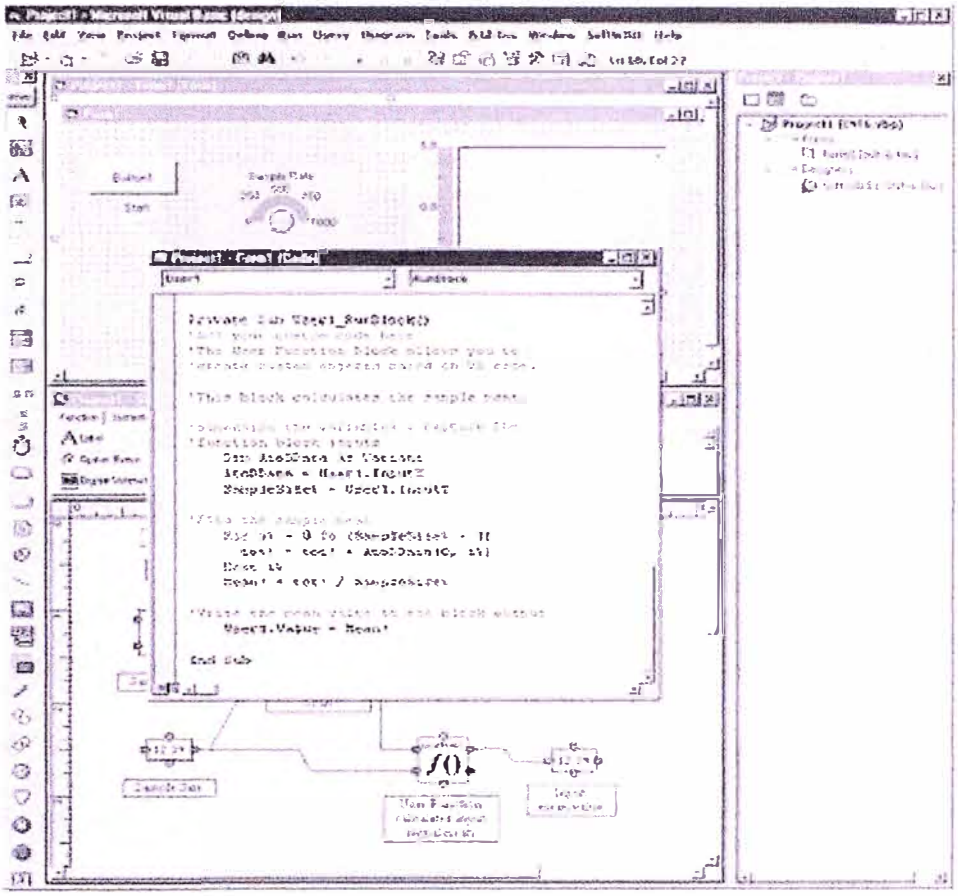
(Figura 3.2.4) Diseñador Intuitivo

-Con sólo algunos componentes ya se puede tener un programa funcional



(Figura 3.2.4) Programa funcional

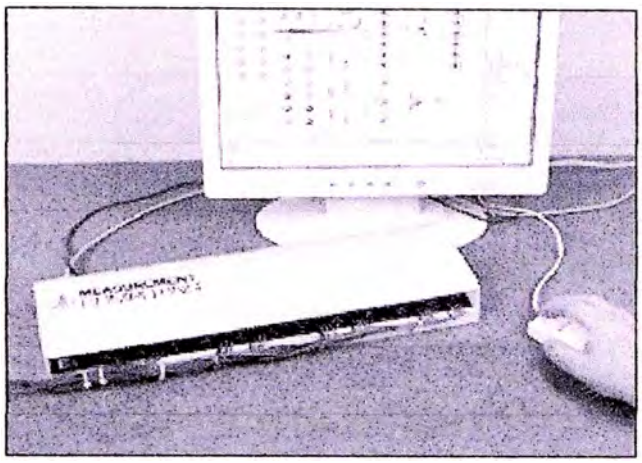
-Tiene la posibilidad de expansión de código sin interfaz gráfica para programas donde sea mejor escribir algunas líneas que jalar muchos componentes (la programación gráfica no es del 100% favorable).



(Figura 3.2.5) Programación con otros lenguajes

-Sus tarjetas DAQ Tienen un bajo precio (por debajo de la mitad del costo de las tarjetas de National instruments)

USB-SSR24 (relevador de estado sólido) es un módulo con 24 pines de I/O. $299.00



(Figura 3.2.6) DAQ en funcionamiento con SoftWire

- **Características de DAQ USB-SSR24**

Es uno de los nuevos dispositivos de Measurement Computing para la adquisición de datos (DAQ) USB-SSR24, ofrece 24 canales aislados de entrada/salida, utilizando módulos de relevadores de estado sólido de tamaño estándar.

Los sistemas de adquisición de datos, básicamente son instrumentos diseñados para muestrear las señales provenientes de un sistema o un fenómeno del mundo real, y convertirlos en información que pueda ser manipulada por una computadora.

Los componentes de estos sistemas incluyen sensores apropiados que convierten las señales del mundo real en una señal eléctrica y un sistema para muestrear esta señal y convertirla en datos digitales y enviaros a una computadora para que procese la información, por medio de software.

(Figura 3.2.7) DAQ USB-SSR24

Las principales características físicas de los nuevos módulos DAQ son: un empaquetado robusto, completamente de aluminio para proteger los componentes electrónicos y el cableado del sistema y la inclusión un puerto USB integrado que permite al usuario encadenar hasta cinco dispositivos sin la necesidad de utilizar concentradores externos.

El sistema USB, comparado con el equivalente sistema basado en PCI, es más fácil de transportar e instalar, ya que no se necesita destapar el gabinete de la PC para poder utilizarlo.

(Figura 3.2.8) Borneras para DAQ

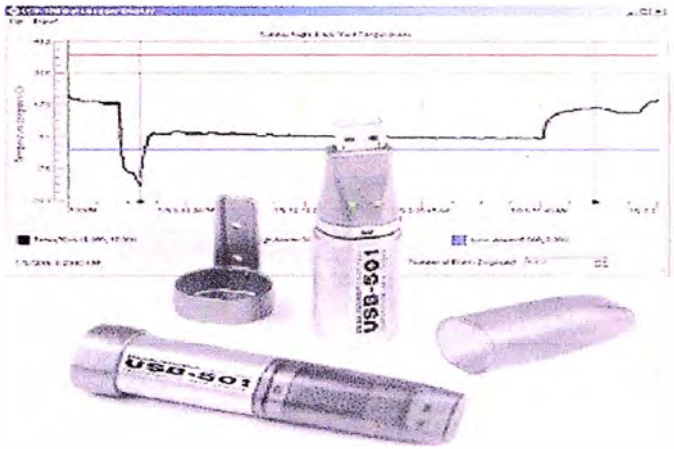
Los sistemas incluyen herramientas de software desarrolladas por la compañía, como la aplicación de almacenamiento de datos TracerDAQ y SoftWIRE una interfaz gráfica de programación para Visual Studio .NET.

Los DAQ también incluyen Universal Library, una interfaz de programación que permite al usuario desarrollar aplicaciones en los lenguajes de programación más populares, utilizando una sintaxis de alto nivel fácil de comprender.

Esta tarjeta complementar la línea de dispositivos USB de adquisición de datos de Measurement Computing que incluía un dispositivo similar al USB-SSR24, pero de 8 canales, el USB-SSR08 y los DAQ USB-ERB08.

- **Características de DAQ PCI-DAS6036**

Tarjeta de 16 Entradas Analógicas, 16 Bit, 200 KHz de frecuencia de muestreo, 8 Líneas Digitales, 2 Contadores 16 Bit, 2 Salidas Analógicas a 10 KHz. Donde una de sus aplicaciones puede ser con el uso de memorias registradoras USB para la medición de temperatura que almacena 16,382 lecturas de temperatura y soporta -35 to 80 °C (+/- 1 °C de exactitud).



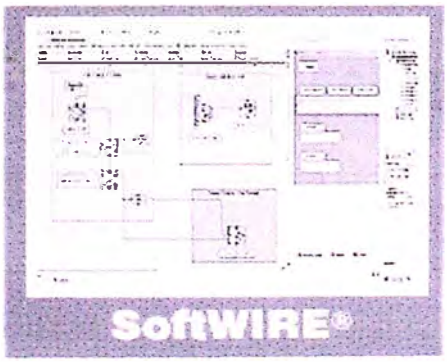
(Figura 3.2.9) Aplicación de una DAQ

Esta aplicación puede ser utilizado en supermercados para monitorear los alimentos perecederos como las frutas, alimentos refrigerados u otros, también puede ser utilizado en almacenes, en cajones de camiones, en camiones de refrigeración, en invernaderos y en muchos otros lugares.

3.3 Herramientas de desarrollo de Measurement Computing

- **SoftWIRE**

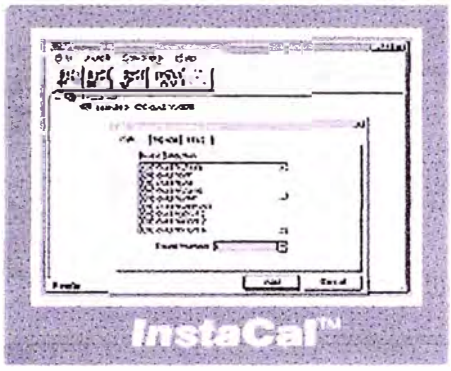
Software gráfico para Visual Basic (Necesario Visual Basic 6.0 profesional).Y ahora interfaz gráfica de programación para Visual Studio® .NET.



(Figura 3.3.1) SoftWire

- **INSTACAL**

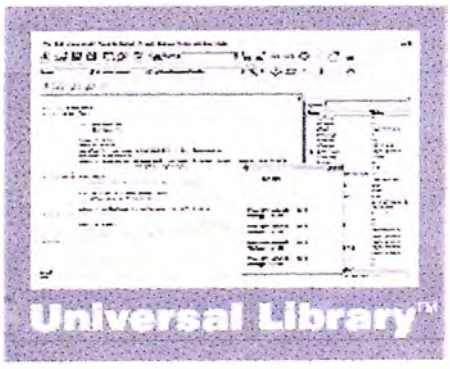
Software de Instalación, Calibración y Test. (Incluido, sin costo, al adquirir cualquier tarjeta).



(Figura 3.3.2) InstaCal

- **UNIVERSAL LIBRARY**

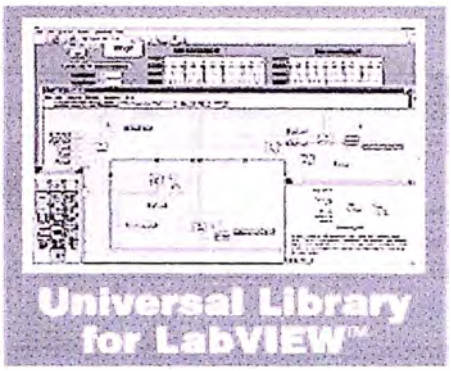
Se proporciona el Lenguaje Universal de Ayuda en Programación para soportar diferentes lenguajes de programación en Windows.



(Figura 3.3.3) Librería Universal

- **UNIVERSAL LABVIEW**

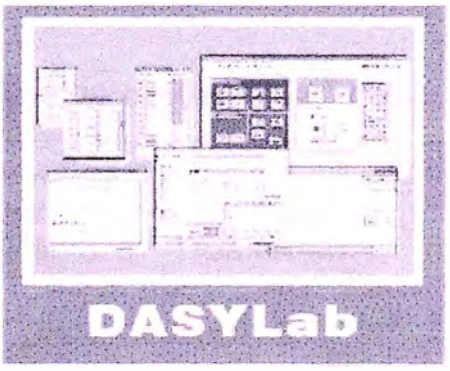
Interface de Labview para Universal Library. Drivers y VIs de LabVIEW incluido.



(Figura 3.3.4) Librería universal para LabView

- **DASYLab**

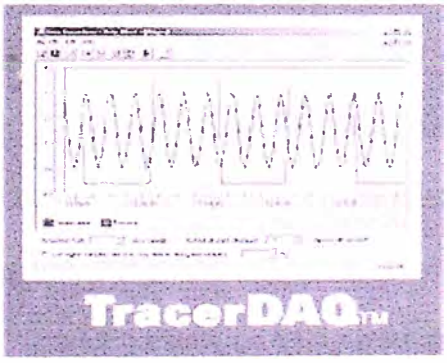
Incrementa la potencialidad de SoftWire con más de 1000 componentes.



(Figura 3.3.5) DASYLab

- **TracerDAQ**

Registrador profesional de datos con interfaz gráfica de despliegue.



(Figura 3.3.6) TracerDAQ

- **DAS-WIZARD**

Adquisición de Datos, Directo en Excel y Matlab, Combinado con VIX-COMponents para Programadores VBA.

UL PARA LabVIEW 65.86 dólares

UNIVERSAL-LIB/CD 65.86 dólares

SSR-RACK08 133.06 dólares

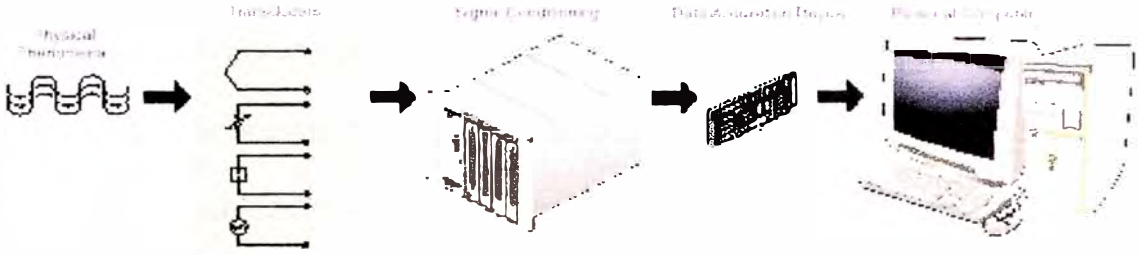
3.4 Acondicionamiento de señal y fuente de alimentación

Los sistemas de adquisición de datos (DAQ) basados en PC son usados en un amplio rango de aplicaciones en los laboratorios, en el campo y en el piso de una planta de manufactura. Típicamente, los dispositivos DAQ son instrumentos de propósito general diseñados para medir señales de voltaje.

- **Acondicionamiento de señal**

La mayoría de los sensores y transductores generan señales que debe acondicionar antes de que un dispositivo DAQ pueda adquirir con precisión la señal. Este procesamiento al frente, conocido como acondicionamiento de señal, incluye funciones como amplificación, filtrado, aislamiento eléctrico y multiplexeo. Además, existen otros sensores que requieren de excitación de voltaje o corriente, completar una configuración de puente, linealización o amplificación para que puedan operar de manera correcta. Es así que la

mayoría de los sistemas DAQ basados en PC incluyen algún tipo de acondicionamiento de señal además del dispositivo DAQ y la PC, como lo muestra la Figura.



(Figura 3.4.1) Diagrama de bloques para el acondicionamiento de señal

- **Fuente de alimentación**

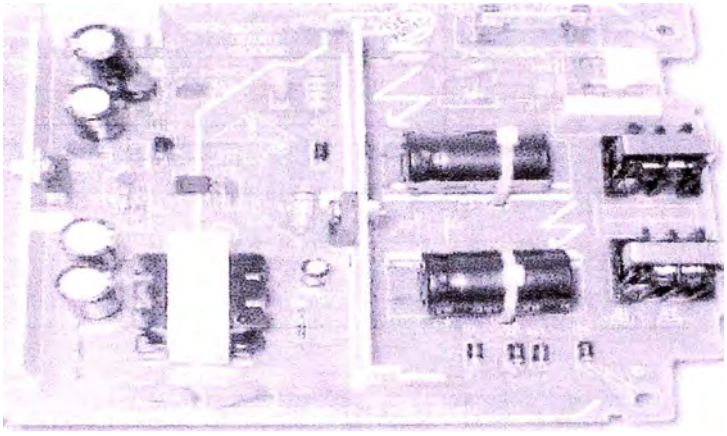
Una fuente de alimentación es un dispositivo o subsistema que convierte la corriente alterna de la red de distribución de la energía eléctrica en otro tipo de corriente eléctrica adecuado para el uso que se le vaya a dar. La función de una fuente de alimentación es convertir la tensión alterna en una tensión continua y lo mas estable posible, para ello se usan los siguientes componentes: 1.- Transformador de entrada; 2.- Rectificador a diodos; 3.- Filtro para el rizado; 4.- Regulador (o estabilizador) lineal. Este último no es siempre imprescindible.

(Figura 3.4.2) Diagrama de bloques de una fuente alimentación

Las fuentes de alimentación se pueden clasificar atendiendo a varios criterios:

- Según el tipo de salida
    - Fuentes de salida fija: su salida en una corriente o tensión que no puede ser modificada.
    - Fuentes de salida ajustable: el valor de la salida puede ser modificado.
    - Fuentes de salida programable: se puede indicar que la salida pase, a lo largo del tiempo y de forma automática por varios valores.
    - Fuentes de salida simple: una única salida
    - Fuentes de salida múltiple: tienen varias salidas independientes.
    - Fuentes de salida continua: la salida es una corriente o tensión cuyo valor no cambia en el tiempo.
    - Fuentes de salida alterna: la salida es una forma de onda periódica.
    - Fuentes lineales: trabajan en régimen lineal.
    - Fuentes conmutadas: trabajan en régimen de conmutación.
- Según la tecnología empleada
    - Fuentes lineales: trabajan en régimen lineal.
    - Fuentes conmutadas: trabajan en régimen de conmutación.
- Según el método de control

- Fuentes digitales: sus sistemas de control son, al menos en parte, digitales.
- Fuentes analógicas: sus sistemas de control son analógicos



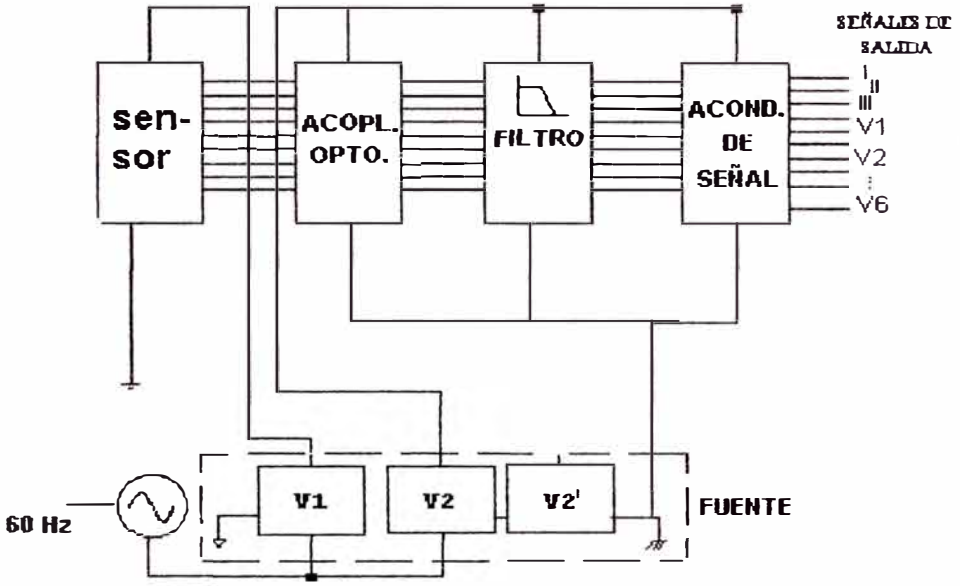
(Figura 3.4.3) Circuito de una fuente de alimentación

3.5 Filtrado y amplificación de señales

El hardware debe adquirir todo tipo de señales en el rango de 0 a 5 voltios para ello hay casos en que la señal se debe amplificar, eliminar o reducir al mínimo el ruido presente en las mismas, haber eliminado las señales cuya frecuencia no interesa a la aplicación, y acondicionar las señales de adquisición dentro de niveles TTL para que puedan ser convertidas digitalmente de una manera adecuada y óptima por el microcontrolador.

En la figura 3.5.1 se observa el diagrama de bloques, para un tipo de aplicación donde se realiza el proceso de adquisición, estas señales pueden variar de 0 a 5v. Dichas señales recorren las distintas etapas de

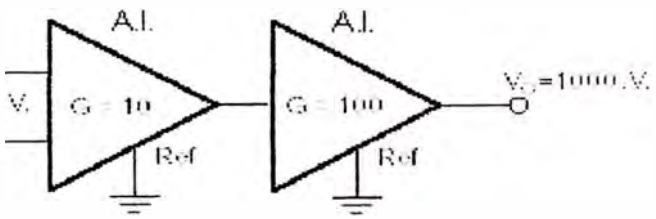
acondicionamiento para luego tener una entrada al conversor Analógico/ Digital de la tarjeta de adquisición de datos que desarrollamos.



(Figura 3.5.1) Diagrama de bloques de un proceso de adquisición

- **Amplificación de señales**

Esta etapa se encarga de amplificar las señales analógicas del sensor, empleando para ello sensores capacitivos, inductivos, etc. Con esta etapa se amplifican las señales, las cuales se encuentra en el orden de los milivoltios, micro voltios dependiendo del sensor.

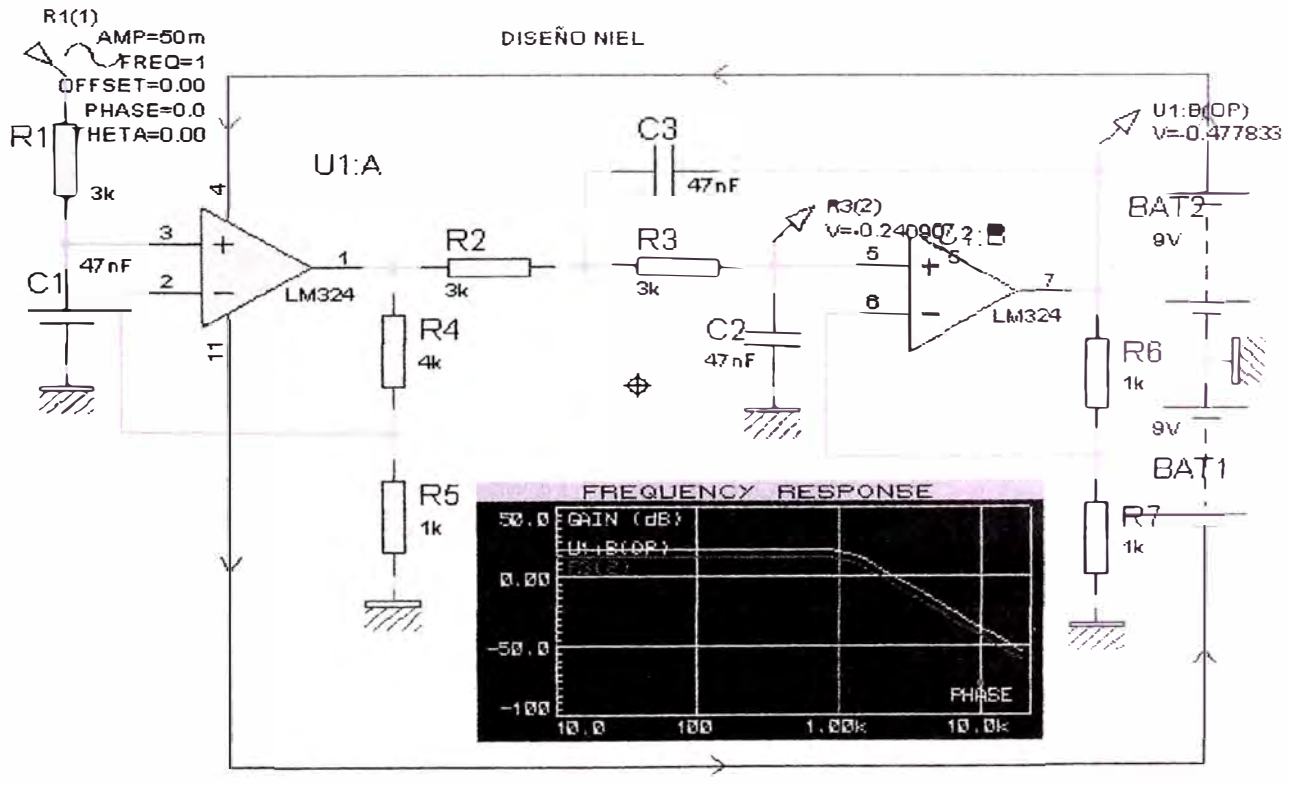


(Figura 3.5.2) Amplificación de 2 etapas

- **Filtrado de señales**

Luego de la amplificación es necesario eliminar o reducir el ruido presente en la fuente de señal, para de esa manera trabajar con datos relevantes. Para ello utilizaremos filtros que solo peritan el paso de las frecuencias bajas, o aquellas frecuencias que la aplicación del usuario requiera.

Por ejemplo acá mostramos el esquema de un filtro pasa bajo de 3 polos con frecuencia de corte de 500 hz. Tenga en cuenta que cada usuario hará sus cálculos respectivos, ya que la tarjeta ADN2006 es genérica, por lo tanto procesa los datos de 0 a 5 voltios que ingresan a través de sus canales analógicos y con una frecuencia máxima limitada por el microcontrolador usado que en nuestro caso es el PIC16F877A.



(Figura 3.5.3) Filtrado de señales

CAPÍTULO IV

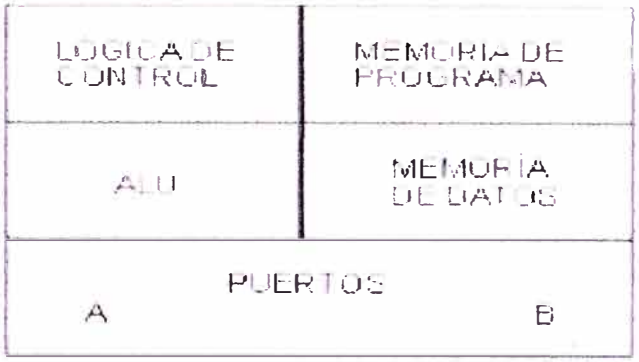
CARACTERÍSTICAS DEL FIRMWARE DE UN DISPOSITIVO DE ADQUISICIÓN

4.1 Características del microprocesador y microcontrolador

- **¿Qué es un microcontrolador?**

Son componentes sumamente útiles en la electrónica de consumo, normalmente llamados circuitos integrados programables. Aún cuando son conocidos desde hace más de veinte años, existen en la actualidad nuevos tipos que cumplen con una serie de requisitos y características sumamente útiles.

Como una primera aproximación podemos definir a un PIC como "un chip que me permite obtener un circuito integrado a mi medida", es decir puedo hacer que el PIC se comporte como un procesador de luminancia o un temporizador o cualquier otro sistema mediante un programa que le grabo en una memoria ROM interna.



(Figura 4.1.1) Arquitectura simplificada de un Microcontrolador Microchip

Los microcontroladores PIC son en el fondo procesadores similares a otros tipos, como por ejemplo la familia de los microprocesadores X86, 80486, Pentium I, II, III, IV o las famosas Dual Score (Multiprocesador) y muchos otros que usan una arquitectura interna del tipo Von Neumann. En este tipo de arquitectura los datos y la memoria del programa se encuentran en el mismo espacio de direcciones.

- **¿Qué es un microprocesador?**

En realidad un microprocesador y un microcontrolador no son la misma cosa. Los PICs son microcontroladores, es decir, una unidad que posee en su interior al microprocesador y a los elementos indispensables para que pueda funcionar como una mini computadora en un solo chip.

Un microprocesador es solamente la unidad central de procesos o CPU.

La memoria, los puertos y todos los demás periféricos son exteriores. La programación de un microprocesador es, por lo tanto, una tarea compleja porque deben controlarse todos estos dispositivos externos. Es decir el

microprocesador es un sistema abierto mientras que un microcontrolador es un sistema cerrado.

Un microcontrolador integra la CPU y todos los periféricos en un mismo chip. El programador se desentiende de una gran cantidad de dispositivos y se concentra en el programa de trabajo. Esta circunstancia da lugar a una gran pérdida de tiempo porque los datos tienen que ser retirados de la memoria y llevados a la CPU (Unidad de procesamiento central) y viceversa. Esto significa que la computadora dedica la mayor parte del tiempo al transporte de datos de ida o de vuelta, en lugar de usar este tiempo para trabajar sobre los datos. Por eso las actuales Pentium usan memoria de caché que está incrustado en el microprocesador.

Los PICs emplean un conjunto de instrucciones del tipo RISC (Reduced Instruction Set Computer). Con el RISC se suele ejecutar la mayoría de las instrucciones con un solo pulso del clock. Con las instrucciones que se usan en otros equipos del tipo CISC (Complex Instruction Set Computer), se logran instrucciones más poderosas, pero a costa de varios ciclos del clock. En el bien conocido procesador 68HC11 de Motorola se requieren a veces hasta 5 ciclos del clock para ejecutar una instrucción.

A los fines prácticos nos vamos a referir a los microcontroladores como bloques que poseen una memoria de programa que es el lugar donde deben alojarse los datos que le indiquen al chip qué es lo que debe hacer; una

memoria de datos donde ingresen las señales que debe procesar el programa, una unida aritmética y lógica donde se desarrollen todas las tareas, una unidad de control que se encargue de supervisar todos los procesos y puertos de entrada y salida para que el PIC tenga contacto con el exterior.

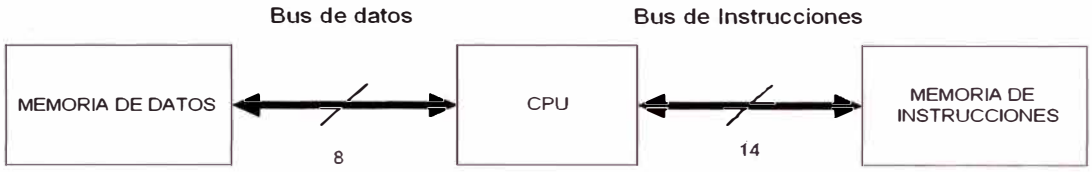
4.2 Arquitectura interna del microcontrolador

Microchip Incorporado en USA Arizona es fabricante de microcontroladores de 8 bits, ofreciendo una gama muy diversa de productos. Se pueden encontrar dispositivos con instrucciones de hasta 16 bits como los famosos dsPIC, 12 bits y encapsulados desde 8 pines hasta 68 pines.

La combinación de operación rápida con una característica de bajo consumo y bajo costo hace que la familia de estos microcontroladores sean productos muy populares en el mercado. A continuación se presentan las diferentes arquitecturas y modelos del microprocesador

- **Arquitectura del procesador modelo Harvard**

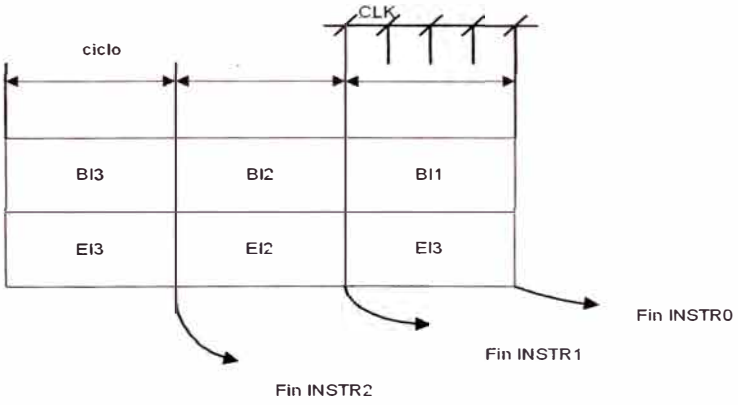
Aquí el CPU se conecta de forma independiente y con buses distintos con la memoria de instrucciones y con la de datos, permitiendo el acceso simultáneo a estas.



(Figura 4.2.1) Arquitectura Harvard

- **Técnica de segmentación ("pipe-line")**

En la ejecución de las instrucciones se le permite al procesador realizar al mismo tiempo la ejecución de una instrucción y la búsqueda del código de la siguiente. De esta forma se puede ejecutar cada instrucción en un ciclo (un ciclo de instrucción equivale a 4 ciclos de reloj)



(Figura 4.2.2) Técnica pipe-line

- **Procesador RISC**

Es un procesador con conjunto reducido de instrucciones. Los modelos de la gama baja disponen de un repertorio de 33 instrucciones, 35 los de la gama media y 58 los de la alta.

• Pila de hardware de 8 niveles de profundidad

• Modos de direccionamiento relativo, directo e indirecto

• Power-on Reset (POR)

• Power-up Timer (PWRT) y Oscillator Start-up Timer (OST)

• Watchdog Timer (WDT) con oscilador propio en-chip RC para confianza de operación

• Protección de código programable

• Modo SLEEP de ahorro de energía

• Opciones de oscilador seleccionable

• Tecnología CMOS FLASH/EEPROM de bajo consumo y alta velocidad

• Diseño estático completo

• Programación Serial In-Circuit (ICSP) vía 2 pines

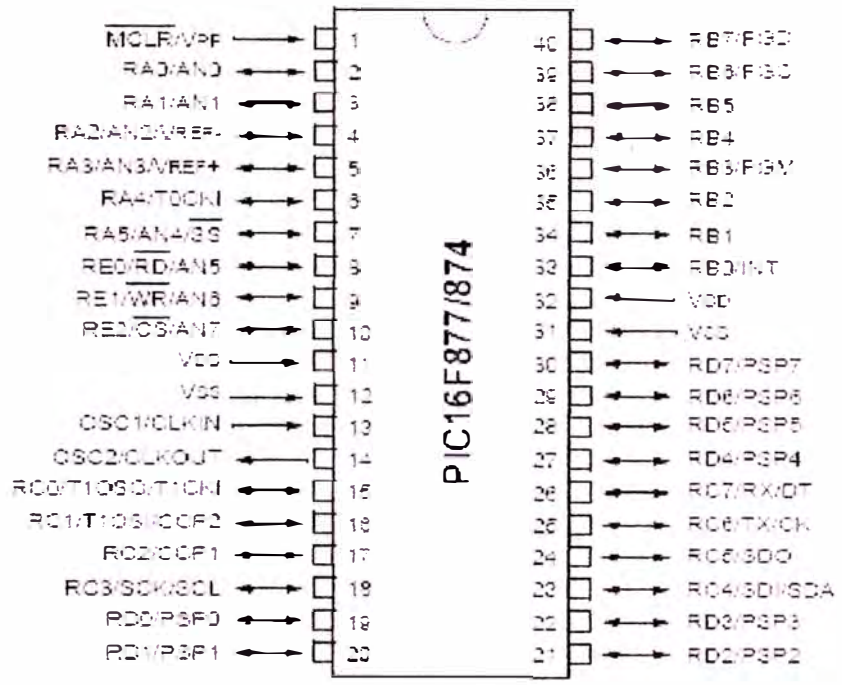
• Capacidad de Programación Serial In-Circuit de 5V

• Depuración In-Circuit vía 2 pines

• Microprocesador lee y escribe para acceder a memoria de programa

• Amplio rango de voltaje de operación: 2.0V a 5.5V

• Alta corriente de Fuente/Drenaje: 25 mA



(Figura 4.2.3) Diagrama de Pines

• Rango de temperatura Extendida, Comercial e Industrial

• Bajo consume de energía:

- Típico < 0.6 mA a 3V, 4 MHz

- Típico 20 µA a 3V, 32 kHz

- Típica corriente de standby < 1 µA

- **Características de los periféricos del Microcontrolador:**

• Timer0: Timer/contador de 8-bit con pre escalador de 8-bit

• Timer1: Timer/contador 16-bit con pre escalador, puede ser incrementado durante SLEEP vía externo cristal/clock

• Timer2: Timer/contador de 8-bit con pre escalador, post escalador y registro de periodo de 8-bit

• 2 Módulos de Captura, Comparador, PWM

- Captura es de 16-bit, resolución máx. de 12.5 ns

- Comparador es de 16-bit, resolución máx. de 200 ns

- PWM resolución máx. de 10-bit

• Conversor Analógico Digital de 10-bit multi-canal

• Puerto Serial Síncrono (SSP) con SPI (Modo Master) e I2C (Master/Esclavo)

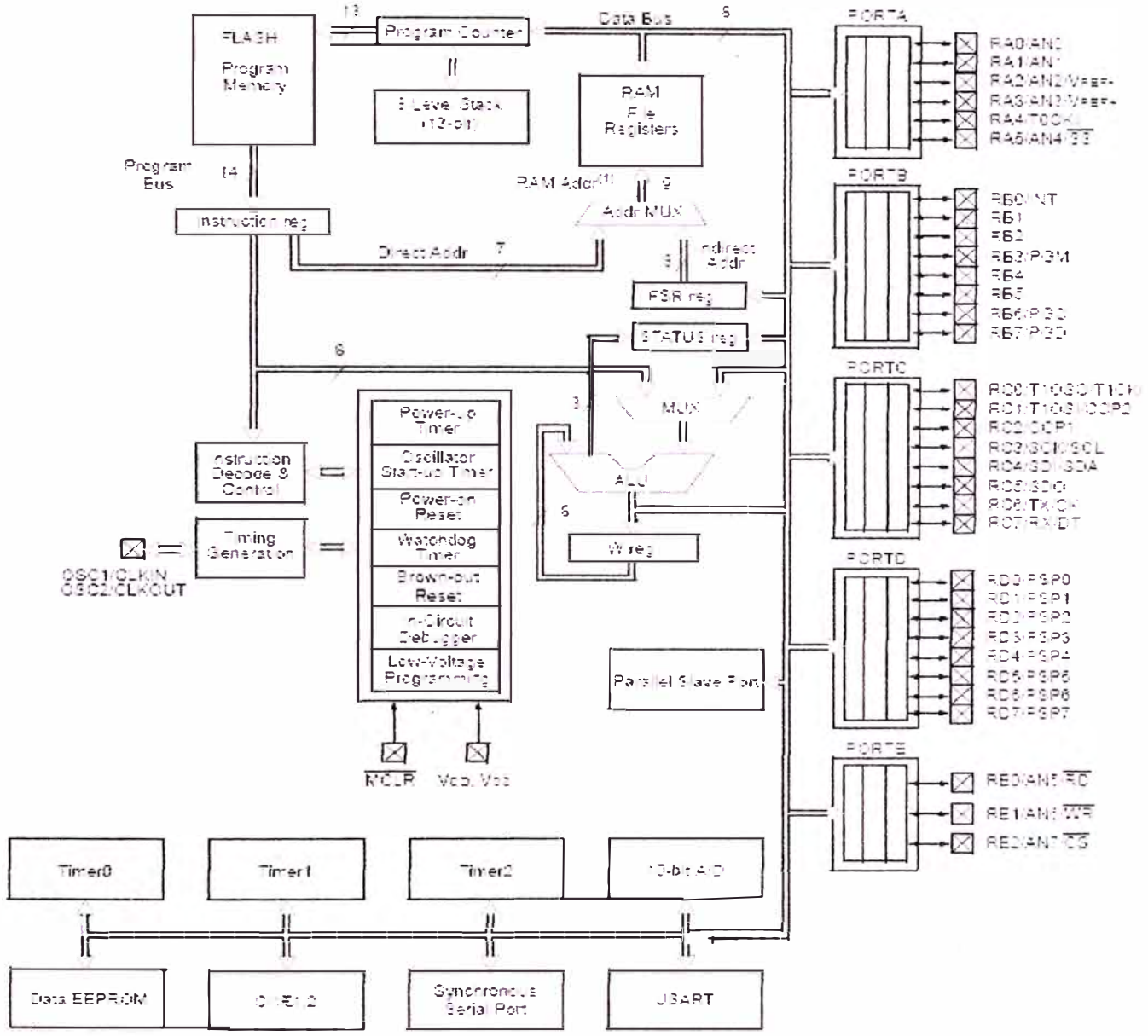
• Transmisor Receptor Asíncrono Síncrono Universal (USART/SCI) con 9-bit de detección de dirección.

• Puerto Paralelo Esclavo (PSP) de 8-bits de ancho, con RD externo, controles WR y CS

• Circuito de detección Brown-out para reset Brown-out (BOR)

- **Vista de los bloques internos del Microcontrolador:**

Diagrama de bloques del microcontrolador



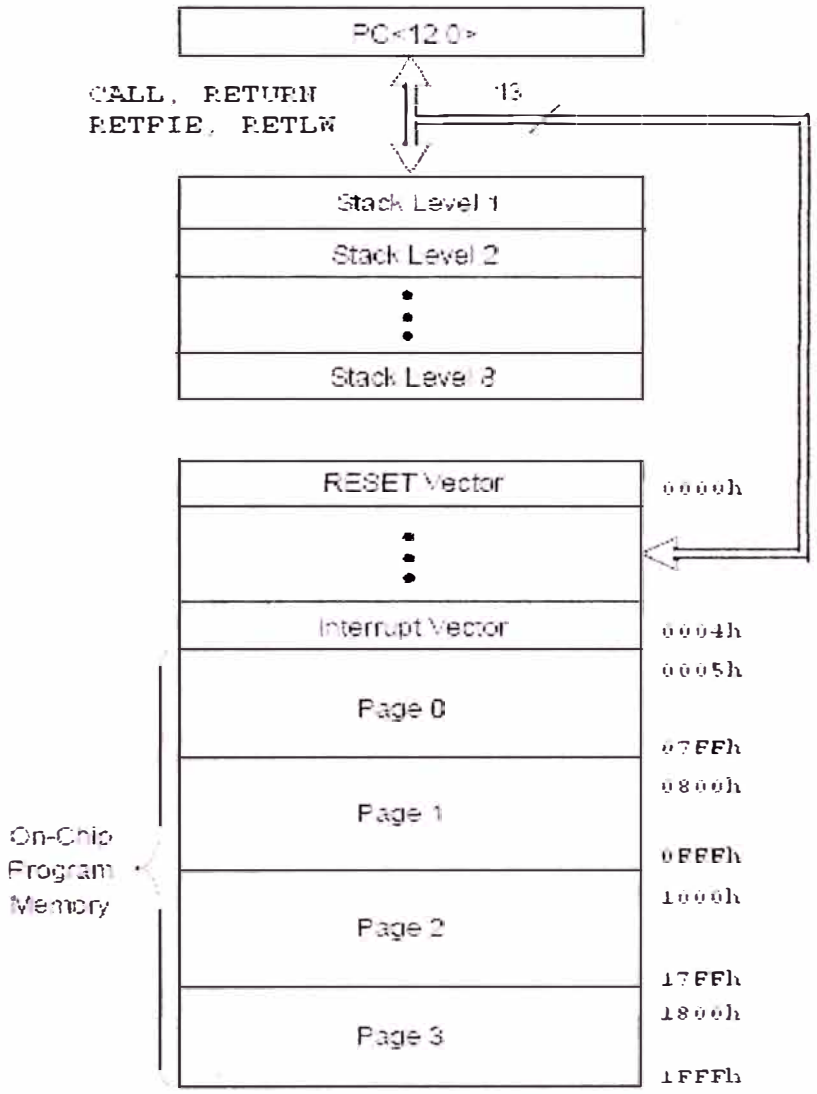
(Figura 4.2.3) Diagrama de bloques del PIC16F877A

4.3 Organización de la memoria

Hay 2 bloques de memoria: de programa, de datos.

- **Organización de memoria de programa**

Contador de programa de 13-bit con capacidad de direccionamiento de 8K x 14 palabras de memoria programa FLASH.

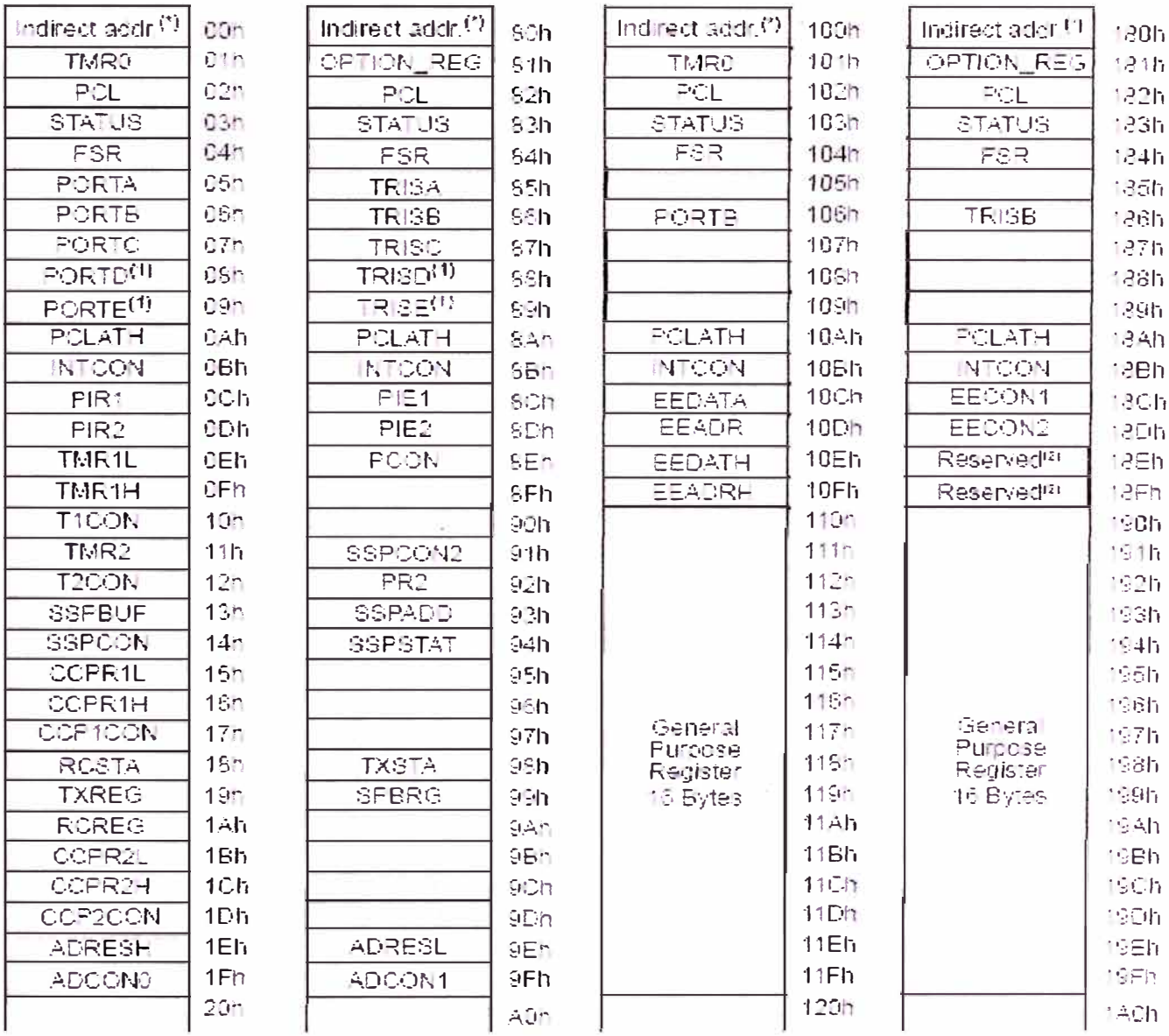


(Figura 4.3.1) Mapa y pila de memoria de programa del microcontrolador

- **Organización de memoria de datos**

Memoria de datos es particionado en 4 Bancos seleccionados con los Bits RP1 (STATUS<6>) y RP0 (STATUS<5>). Cada Banco se extiende hasta

7Fh (128 puertos). Las bajas localizaciones de cada Banco son reservadas para Registros de Funciones Especiales y los demás son registros de propósito general.



(Figura 4.3.1) Mapa de archivos de registros del microcontrolador

RP1:RP0	Bank
00	0
01	1
10	2
11	3

(Figura 4.3.2) Partición de memoria en bancos

- **Detalle de registros del microcontrolador**

80h(3)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000
81h	OPTION_REG	$\overline{RBPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111
82h(3)	PCL	Program Counter (PC) Least Significant Byte								0000 0000
83h(3)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx
84h(3)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx
85h	TRISA	—	—	PORTA Data Direction Register						--11 1111
86h	TRISB	PORTB Data Direction Register								1111 1111
87h	TRISC	PORTC Data Direction Register								1111 1111
88h(4)	TRISD	PORTD Data Direction Register								1111 1111
89h(4)	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction Bits			0000 -111
8Ah(1,3)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000
8Bh(3)	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x
8Ch	PIE1	PSPIE(2)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000
8Dh	PIE2	—	(5)	—	EEIE	BCLIE	—	—	CCP2IE	-r-0 0--0
8Eh	PCON	—	—	—	—	—	—	$\overline{POR}$	$\overline{BOR}$	---- --qq
8Fh	—	Unimplemented								—
90h	—	Unimplemented								—
91h	SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	0000 0000
92h	PR2	Timer2 Period Register								1111 1111
93h	SSPADD	Synchronous Serial Port (I²C mode) Address Register								0000 0000
94h	SSPSTAT	SMP	CKE	$\overline{D/A}$	P	S	$\overline{R/W}$	UA	BF	0000 0000
95h	—	Unimplemented								—
96h	—	Unimplemented								—
97h	—	Unimplemented								—
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010
99h	SPBRG	Baud Rate Generator Register								0000 0000
9Ah	—	Unimplemented								—
9Bh	—	Unimplemented								—
9Ch	—	Unimplemented								—
9Dh	—	Unimplemented								—
9Eh	ADRESL	A/D Result Register Low Byte								xxxx xxxx
9Fh	ADCON1	ADFM	—	—	—	PCFG3	PCFG2	PCFG1	PCFG0	0--- 0000

(Figura 4.3.3) Mapa del Banco 1

00h[3]	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								xxxx xxxx
01h	TMR0	Timer0 Module Register								xxxx xxxx
02h[3]	PCL	Program Counter (PC) Least Significant Byte								0000 0000
03h[3]	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx
04h[3]	FSR	Indirect Data Memory Address Pointer								xxxx xxxx
05h	PORTA	—	—	PORTA Data Latch when written: PORTA pins when read						--0x 0000
06h	PORTB	PORTB Data Latch when written: PORTB pins when read								xxxx xxxx
07h	PORTC	PORTC Data Latch when written: PORTC pins when read								xxxx xxxx
08h[4]	PORTD	PORTD Data Latch when written: PORTD pins when read								xxxx xxxx
09h[4]	PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -xxx
0Ah[1,3]	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000
0Bh[3]	INTCON	GIE	PEIE	TOIE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x
0Ch	PIR1	PSPIF[3]	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000
0Dh	PIR2	—	(5)	—	EEIF	BCLIF	—	—	CCP2IF	-r-0 0---0
0Eh	TMR1L	Holding register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx
0Fh	TMR1H	Holding register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	--00 0000
11h	TMR2	Timer2 Module Register								0000 0000
12h	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000
13h	SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register								xxxx xxxx
14h	SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000
15h	CCPR1L	Capture/Compare/PWM Register1 (LSB)								xxxx xxxx
16h	CCPR1H	Capture/Compare/PWM Register1 (MSB)								xxxx xxxx
17h	CCP1CON	—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x
19h	TXREG	USART Transmit Data Register								0000 0000
1Ah	RCREG	USART Receive Data Register								0000 0000
1Bh	CCPR2L	Capture/Compare/PWM Register2 (LSB)								xxxx xxxx
1Ch	CCPR2H	Capture/Compare/PWM Register2 (MSB)								xxxx xxxx
1Dh	CCP2CON	—	—	CCP2X	CCP2Y	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000
1Eh	ADRESH	A/D Result Register High Byte								xxxx xxxx
1Fh	ADCON0	ADCS1	ADCS0	CHS2	CHS1	CHS0	$\overline{GO}$/DONE	—	ADON	0000 00-0

(Figura 4.3.4) Mapa del Banco 0

4.4 Interrupciones y registros especiales del PIC16F877A

- **Registro status (dirección 03h, 83h, 103h, 183h)**

IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C

Bit 7 IRP: Selector de banco de registro (usado para direccionamiento indirecto)

1 = Banco 2, 3 (100h - 1FFh)

0 = Banco 0, 1 (00h - FFh)

Bit 6-5 RP1:RP0: Selector de banco de registro (usado para direccionamiento directo)

11 = Banco 3 (180h - 1FFh)

10 = Banco 2 (100h - 17Fh)

01 = Banco 1 (80h - FFh)

00 = Banco 0 (00h - 7Fh)

Bit 4 TO: Time-out (tiempo – fuera)

1 = Después de power-up, instrucción CLRWDT, o instrucción SLEEP

0 = Un WDT time-out ocurre

Bit 3 PD: Power-down (bajo voltage)

1 = Después power-up o por La CLRWDT instrucción

0 = Por ejecución de la instrucción SLEEP

Bit 2 Z: Cero bit

1 = Resultado de una operación aritmética o lógica es cero

0 = Resultado de una operación aritmética o lógica no es cero

Bit 1 DC: Digito llevada/prestada (ADDWF, ADDLW, SUBLW, SUBWF instrucciones) (Para prestada, La polaridad es inversa)

1 = Un carry-out desde el 4th bit bajo

0 = No carry-out desde el 4th bit bajo

Bit 0 C: Carry/borrow bit (ADDWF, ADDLW, SUBLW, SUBWF instrucciones)

1 = Un carry-out desde el bit mas significativo

0 = No carry-out desde el bit mas significativo

Not: Para prestada, la polaridad es inversa. Una resta es ejecutado por la adición de complemento a 2 del segundo operando

- **Registro option_reg (dirección 81h, 181h)**

RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0

Bit 6 INTEDG: Interrupcion selector de flanco

1 = Interrupcion en flanco de subida de RB0/INT

0 = Interrupcion en flanco de bajada de RB0/INT

Bit 5 T0CS: TMR0 Selector fuente de Clock

1 = Transición en RA4/T0CKI

0 = Clock con ciclo de instrucción interna (CLKOUT)

Bit 4 T0SE: TMR0 Selector de flanco de fuente

1 = Incrementa en flanco de bajada en RA4/T0CKI

0 = Incrementa en flanco de subida en RA4/T0CKI

Bit 3 PSA: Asigna pre escalador

1 = Prescaler es asignado a WDT

0 = Prescaler es asignado a Timer0

Bit 2-0 PS2:PS0: Velocidad de pre escalador de acuerdo a los 3 bits.

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

- **Registro intcon (dirección 0bh, 8bh, 10bh, 18bh)**

GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF

Bit 7 GIE: Activa interrupción global

1 = Activa toda las interrupciones

0 = Desactiva toda las interrupciones

Bit 6 PEIE: Activa interrupción de periféricos

1 = Activa toda las interrupciones de periféricos

0 = Desactiva toda las interrupciones de periféricos

Bit 4 INTE: RB0/INT Activa interrupción externa

1 = Activa interrupción externa RB0/INT

0 = Desactiva interrupción externa RB0/INT

Bit 1 INTF: Marca de interrupción externa RB0/INT

1 = La interrupción externa ocurre en RB0/INT (debería ser borrado en software)

0 = La interrupción externa no ocurre en RB0/INT

- **Registro pie1 (dirección 8ch)**

PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE

Bit 7 PSPIE (1): Activa interrupción de lectura/escritura en el puerto paralelo esclavo

1 = Activa interrupción de lectura/escritura en PSP

0 = Desactiva interrupción de lectura/escritura en PSP

Bit 3 SSPIE: Activa interrupción de puerto serial síncrono

1 = Activa interrupción de la SSP

0 = Desactiva interrupción de la SSP

Bit 1 TMR2IE: Activa interrupción de emparejamiento de TMR2 a PR2

1 = Activa interrupción de emparejamiento de TMR2 a PR2

0 = Desactiva interrupción de emparejamiento de TMR2 a PR2

Bit 0 TMR1IE: Activa interrupción de desbordamiento del TMR1

1 = Activa interrupción de desbordamiento del TMR1

0 = Desactiva interrupción de desbordamiento del TMR1

- **Registro pir1 (dirección 0ch)**

PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF

Bit 7 PSPIF (1): Marca de interrupción de lectura/escritura en puerto paralelo esclavo

1 = Ha sucedido una operación de lectura/escritura (debería borrarse en software)

0 = No ocurre operación de lectura/escritura

Bit 3 SSPIF: Marca de Interrupcion en puerto serial síncrono (SSP)

1 = La condición de interrupción SSP ha ocurrido (debería borrarse en software)

0 = No ocurre condición de interrupción SSP

Bit 1 TMR2IF: Marca de interrupción de emparejamiento de TMR2 a PR2

1 = Un emparejamiento de TMR2 a PR2 ha ocurrido (debería borrarse en software)

0 = No ocurre un emparejamiento de TMR2 a PR2

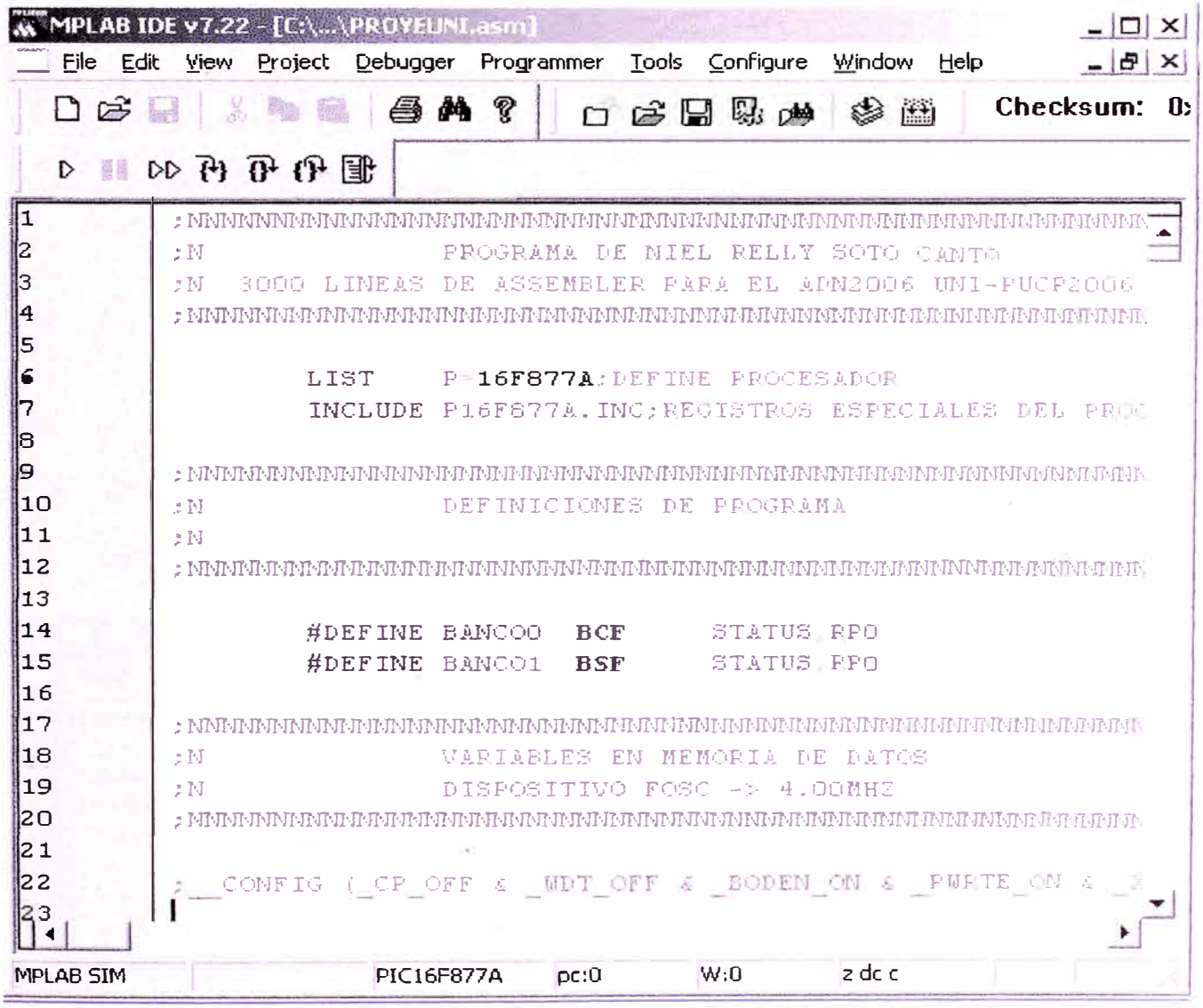
Bit 0 TMR1IF: Marca de interrupción de desbordamiento de TMR1

1 = Registro TMR1 ha desbordado (debería borrarse en software)

0 = Registro TMR1 no ha desbordado

4.5 Desarrollo y depuración de firmware

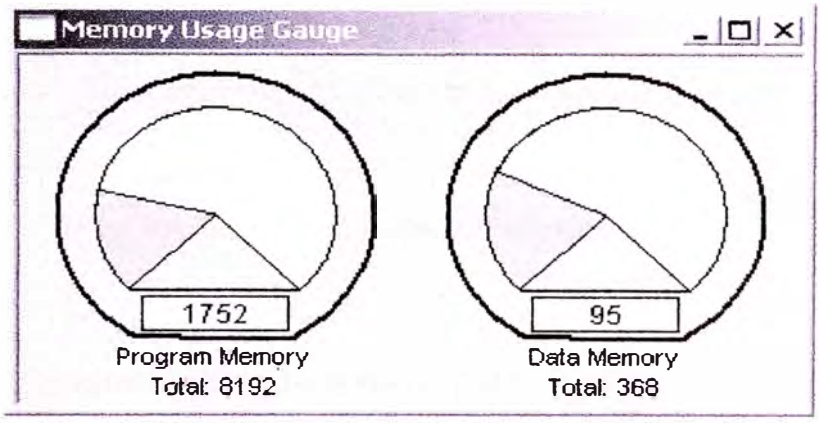
La empresa Microchip y otras que utilizan los PIC ponen a disposición de los usuarios numerosas herramientas para desarrollar hardware y software: programadores (PICSTART PLUS), simuladores software (MPSIM), emuladores en tiempo real (PIC MASTER), ensambladores (MPASM), compiladores C (MP-C), entornos de desarrollo integrados (MPLAB).



(Figura 4.5.1) Entorno del MPLAB

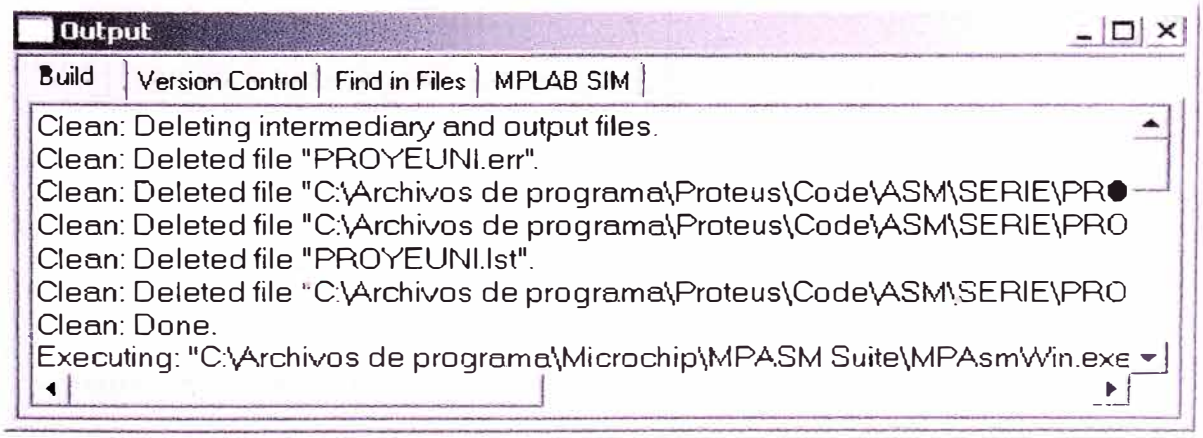
Filter-Lab, FuzzyLAB, MPLINK y MPLIB son también productos de Tecnología Microchip y su valiosa Hoja de Datos el DS30292C disponibles gratuitamente en su Web www.microchip.com. A continuación partes principales para el desarrollo y depuración del firmware.

- **Cuadro de uso de memoria:** Permite conocer la capacidad total de las memorias disponibles y también su actual uso.



(Figura 4.5.2) Uso de memoria

- **Cuadro de salida del compilador:** Muestra información del resultado de la compilación del firmware, como errores y archivos generados.



(Figura 4.5.3) Resultados de compilación

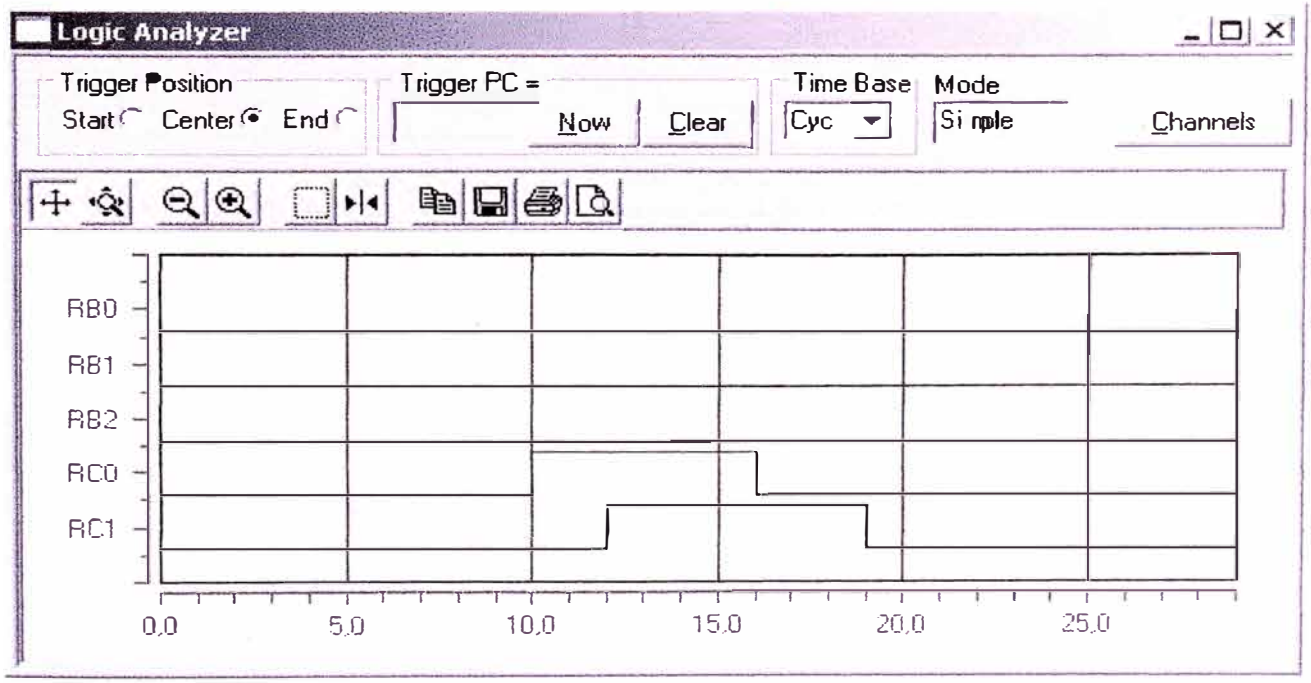
- **Cuadro de seguimiento de memoria:** Analiza con detalle la instrucción realizada en cada línea y la dirección asignada.

Trace

Line	Addr	Op	Label	Instruction	SA	SD	DA	DD	Cycles
-3949	0048	2842		GOTO 0x42	----	--	----	--	301005788
-3948	0049	0000		NOP	----	--	----	--	301005789
-3947	0042	0000	DENUEVO	NOP	----	--	----	--	301005790
-3946	0043	301F		MOVLW 0x1f	W	--	W	1F	301005791
-3945	0044	00A4		MOVWF 0x24	----	--	0024	1F	301005792
-3944	0045	0BA4	NOCERO	DECFSZ 0x24, F	0024	1F	0024	1E	301005793
-3943	0046	2845		GOTO 0x45	----	--	----	--	301005794
-3942	0047	0000		NOP	----	--	----	--	301005795
-3941	0045	0BA4	NOCERO	DECFSZ 0x24, F	0024	1E	0024	1D	301005796
-3940	0046	2845		GOTO 0x45	----	--	----	--	301005797
-3939	0047	0000		NOP	----	--	----	--	301005798

(Figura 4.5.4) Resultados de compilación

- **Cuadro de analizador de datos:** Este analizador lógico permite observar el resultado de un dato en los pines de entrada.

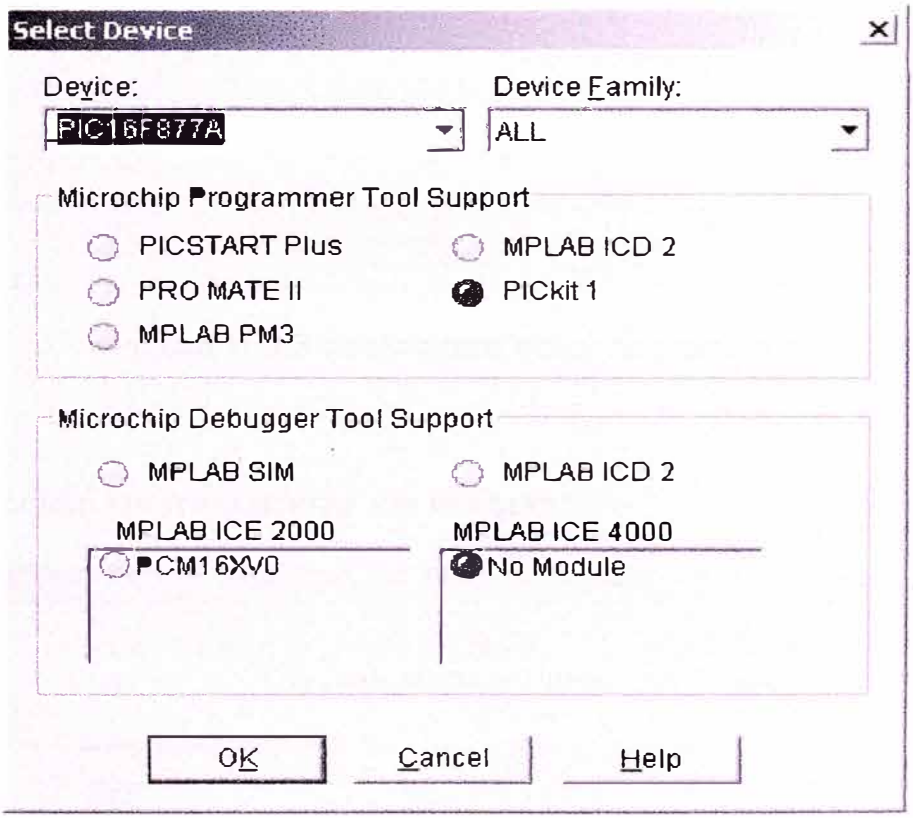


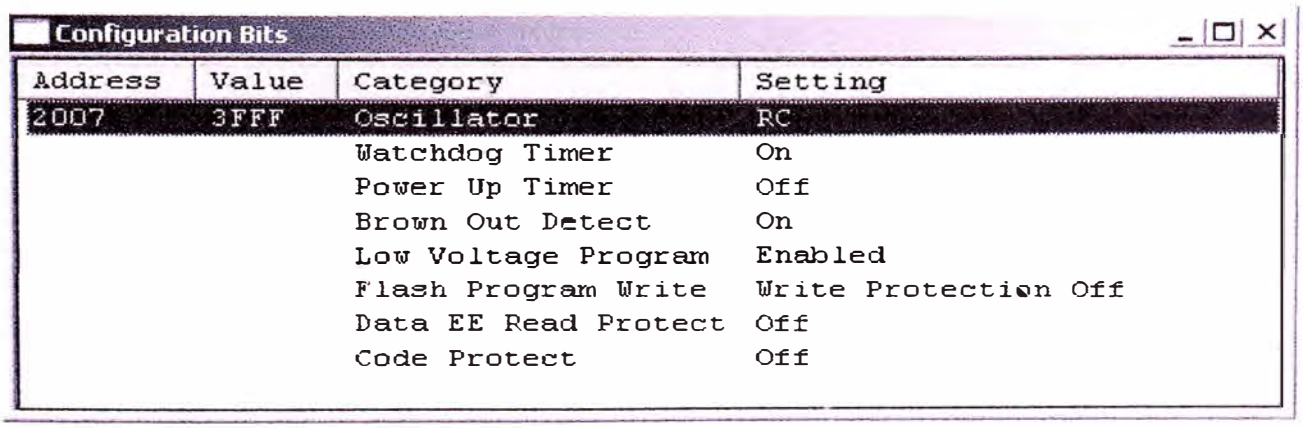
(Figura 4.5.5) Analizador lógico

4.6 Simulación avanzada en funcionamiento del firmware

Para la simulación del firmware se recomienda usar principalmente los emuladores de Hardware en los entornos del MPLAB y el Proteus.

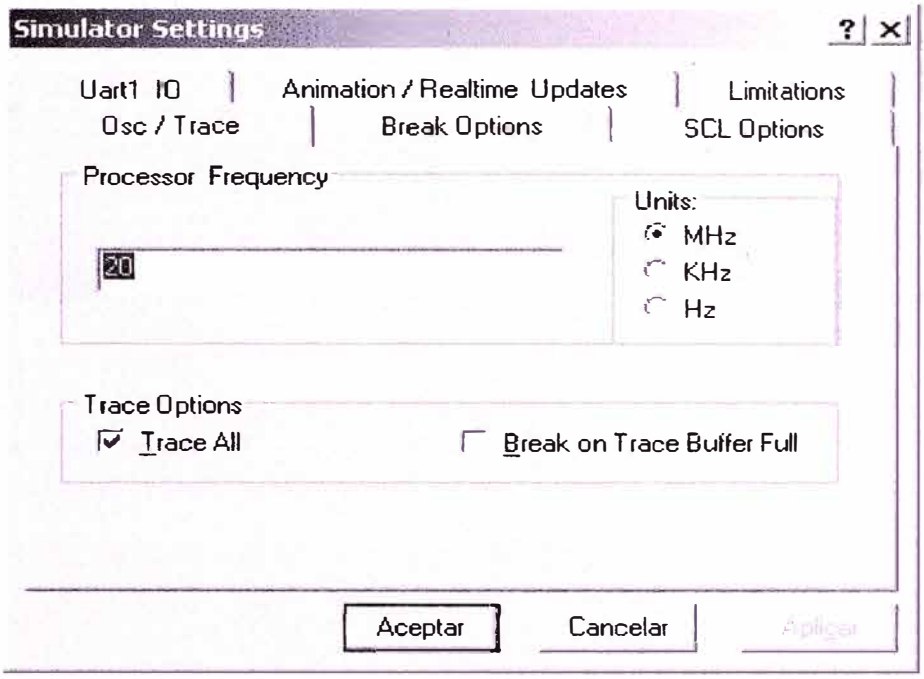
- **Elección del dispositivo y el oscilador**





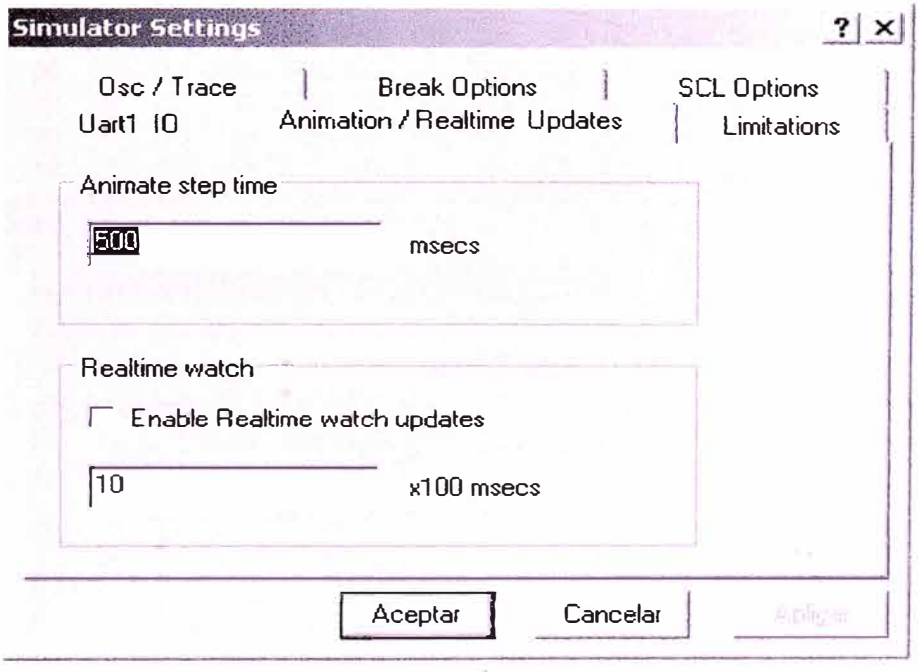
(Figura 4.6.1) Selección del PIC (arriba) y velocidad del oscilador (abajo)

- **Selección de frecuencia y trazado del oscilador**



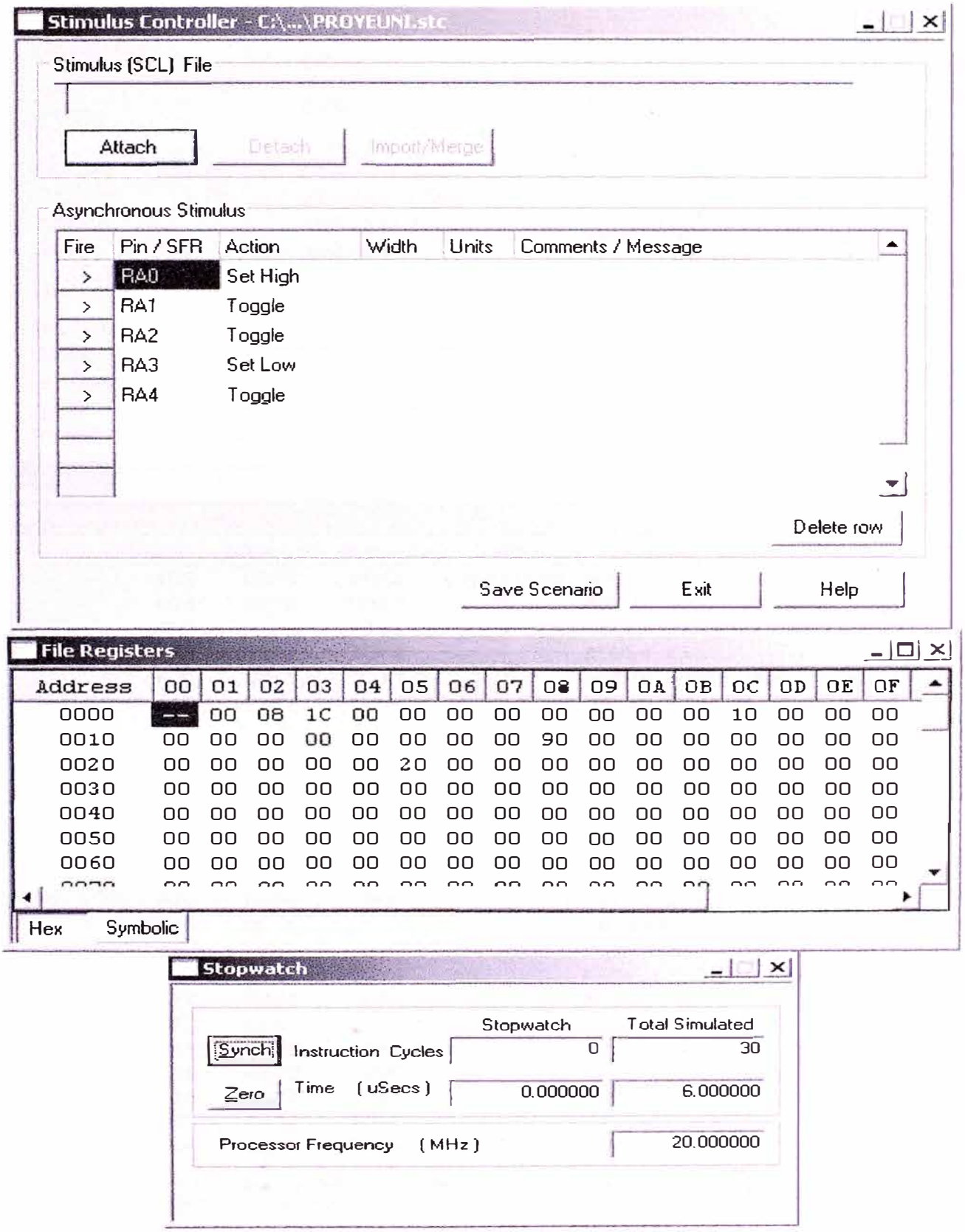
(Figura 4.6.2) Velocidad de procesamiento

- **Selección de frecuencia de animación**



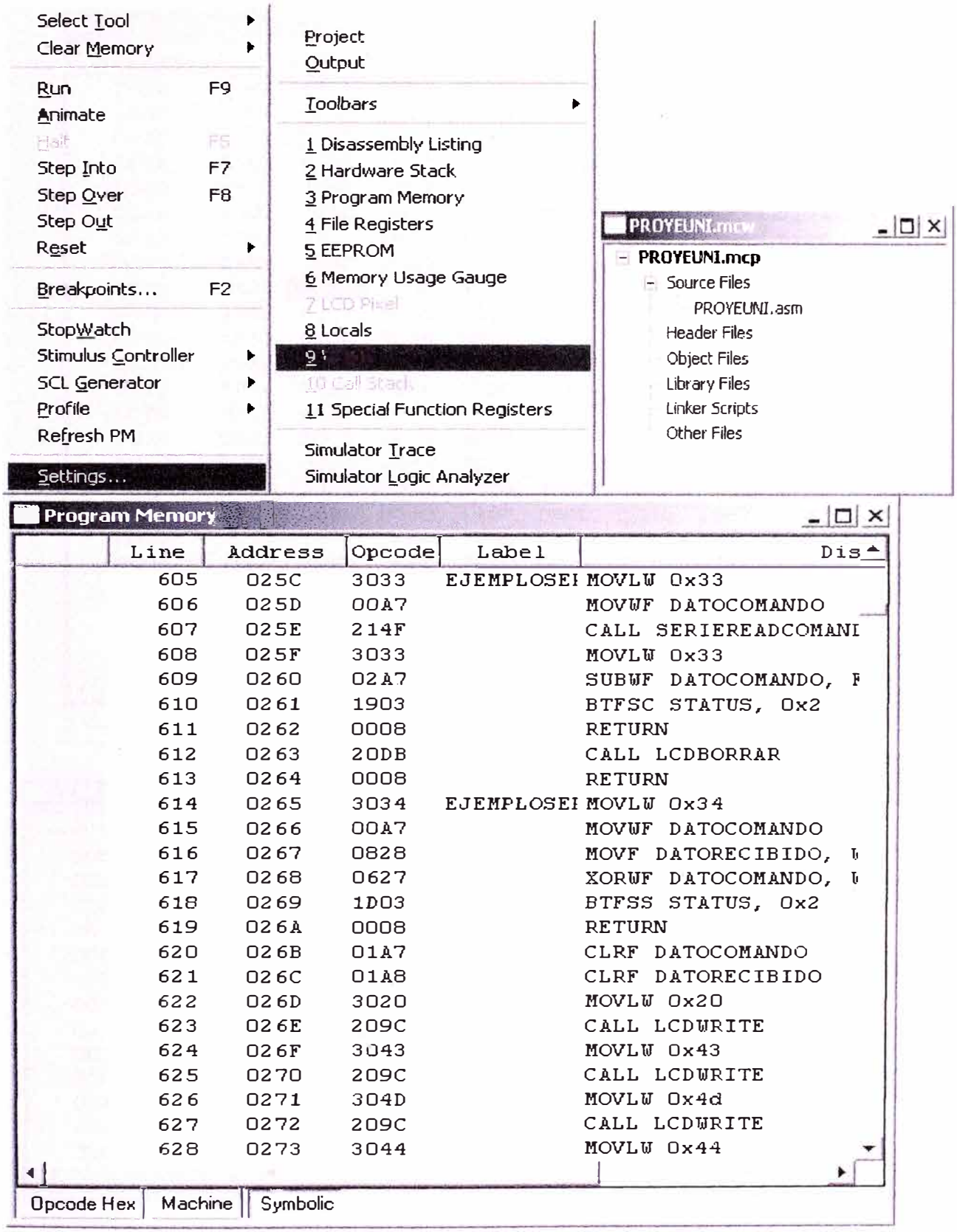
(Figura 4.6.3) Frecuencia de animación en tiempo real

- **Cuadros de entrada y salida con un tiempo de simulación dado**



(Figura 4.6.4) Entrada y Salida de datos binarios con tiempo de simulación

- **Menú de configuraciones de proyecto y memoria de programa**



(Figura 4.6.5) Menú y funcionamiento de memoria de programa

- **Entrada y salida de datos en forma binaria en memoria**

Program Memory

Address									
0000	018A	2805	3FFF	3FFF	2829	2015	2068	2113	.
0008	2026	2156	21E8	0000	2228	2240	225C	2265	&
0010	21FD	21A7	2278	2289	280B	1683	3080	0087	.
0018	0188	019F	30FF	0085	30FF	0086	1605	1208	.
0020	1688	1283	0185	0187	0188	0008	178B	170B	.
0028	0008	00A0	0803	0183	00A1	080A	00A2	018A	.
0030	1683	1E8C	283A	1283	1E8C	283A	128C	2128	.
0038	2134	209C	0183	0822	008A	0821	0083	0EA0	4
0040	0E20	0009	0000	301F	00A4	0BA4	2845	0BA3	
0048	2842	0008	300A	00A3	2042	0008	3032	00A3	B
0050	2042	0008	3064	00A3	2042	0008	2052	2052	B
0058	2052	2052	2052	0008	2056	2056	0008	205C	R
0060	205C	205C	205C	205C	0008	205F	205F	0008	\
0068	0188	0000	1007	1087	3078	00A3	2042	3022	.
0070	00A3	2042	1007	1087	3003	0088	0000	1487	.
0078	0000	0000	1087	3032	00A3	2042	3003	0088	.
0080	0000	1487	0000	0000	1087	3005	00A3	2042	.
0088	3003	0088	0000	1487	0000	0000	1087	3002	.
0090	00A3	2042	3002	0088	0000	1487	0000	0000	.
0098	1087	20ED	210F	0008	00A6	3014	00A3	2042	.
00A0	0E26	390F	0088	0000	1407	3032	00A3	2042	&
00A8	0000	1487	0000	0000	1087	0000	0000	0826	.
00B0	390F	0088	0000	1407	0000	0000	1487	0000	.
00B8	0000	1087	0000	0000	1007	0000	0000	0008	.

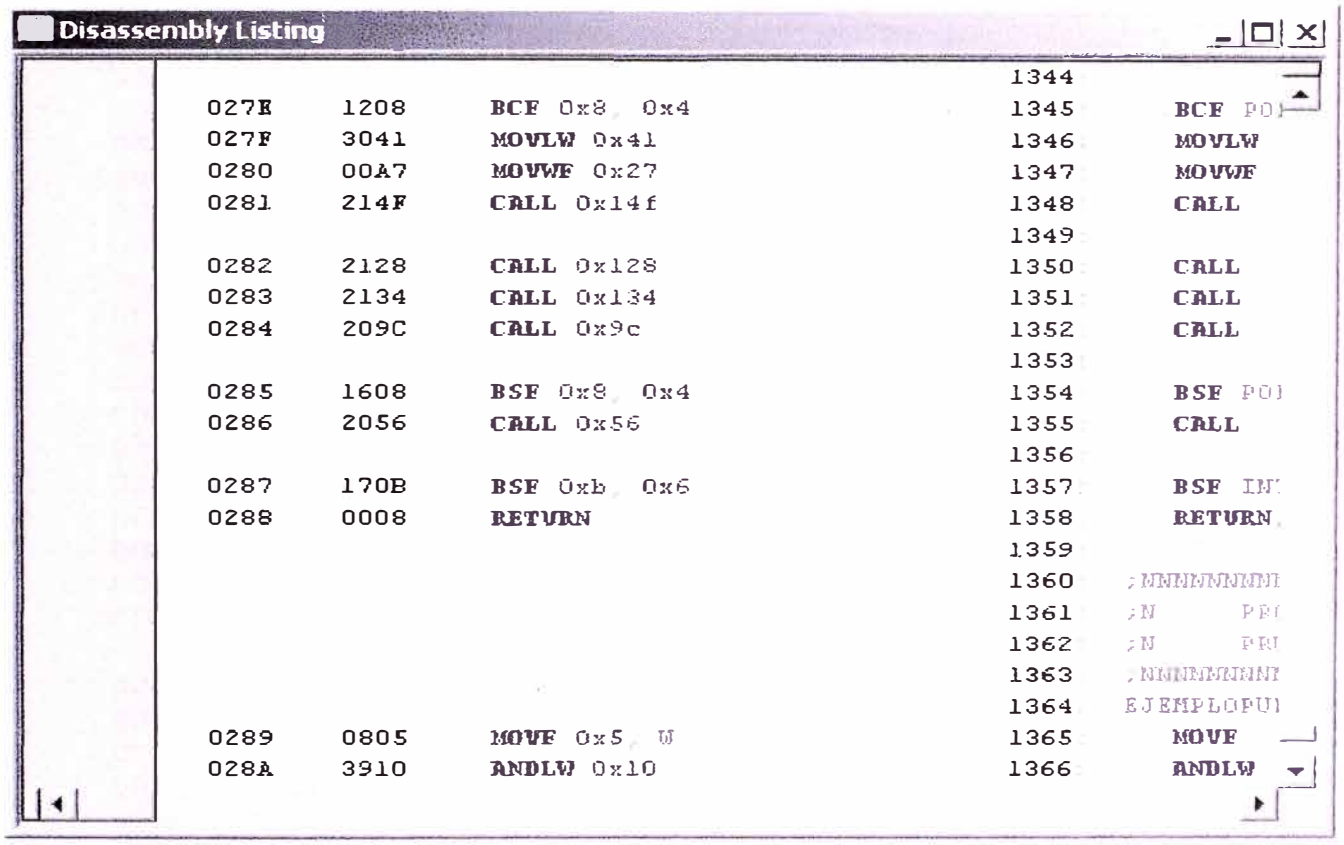
Opcode Hex | Machine | Symbolic

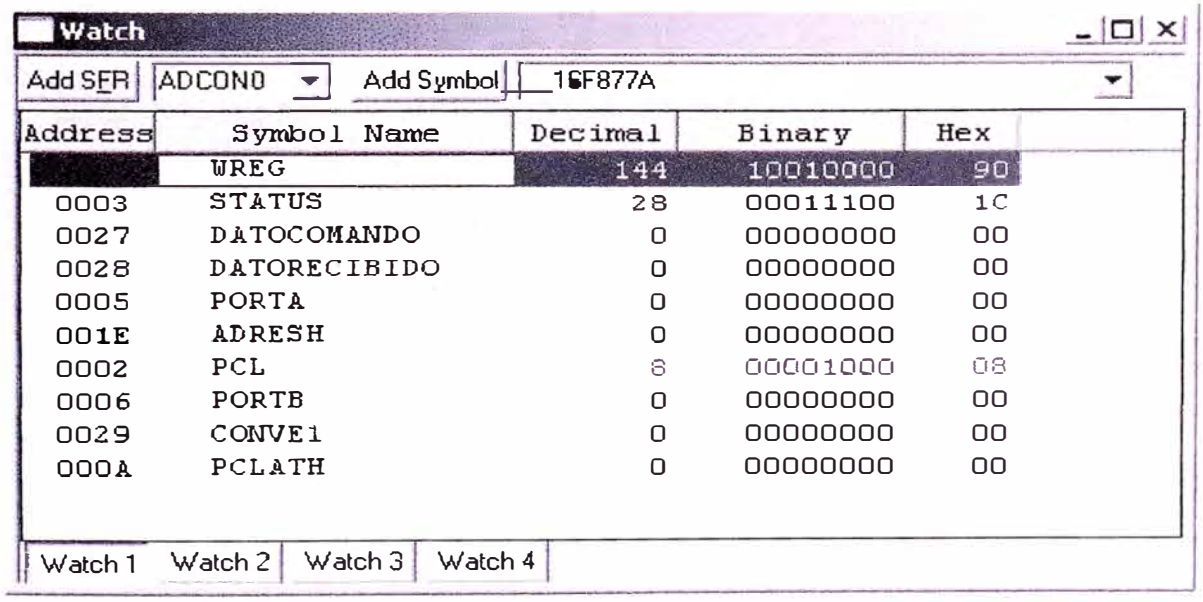
EEPROM

Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
0000	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0010	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0020	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0030	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0040	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0050	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0060	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0070	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0080	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0090	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
00A0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
00B0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
00C0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF

(Figura 4.6.6) Salida de memoria de programa y memoria EEPROM

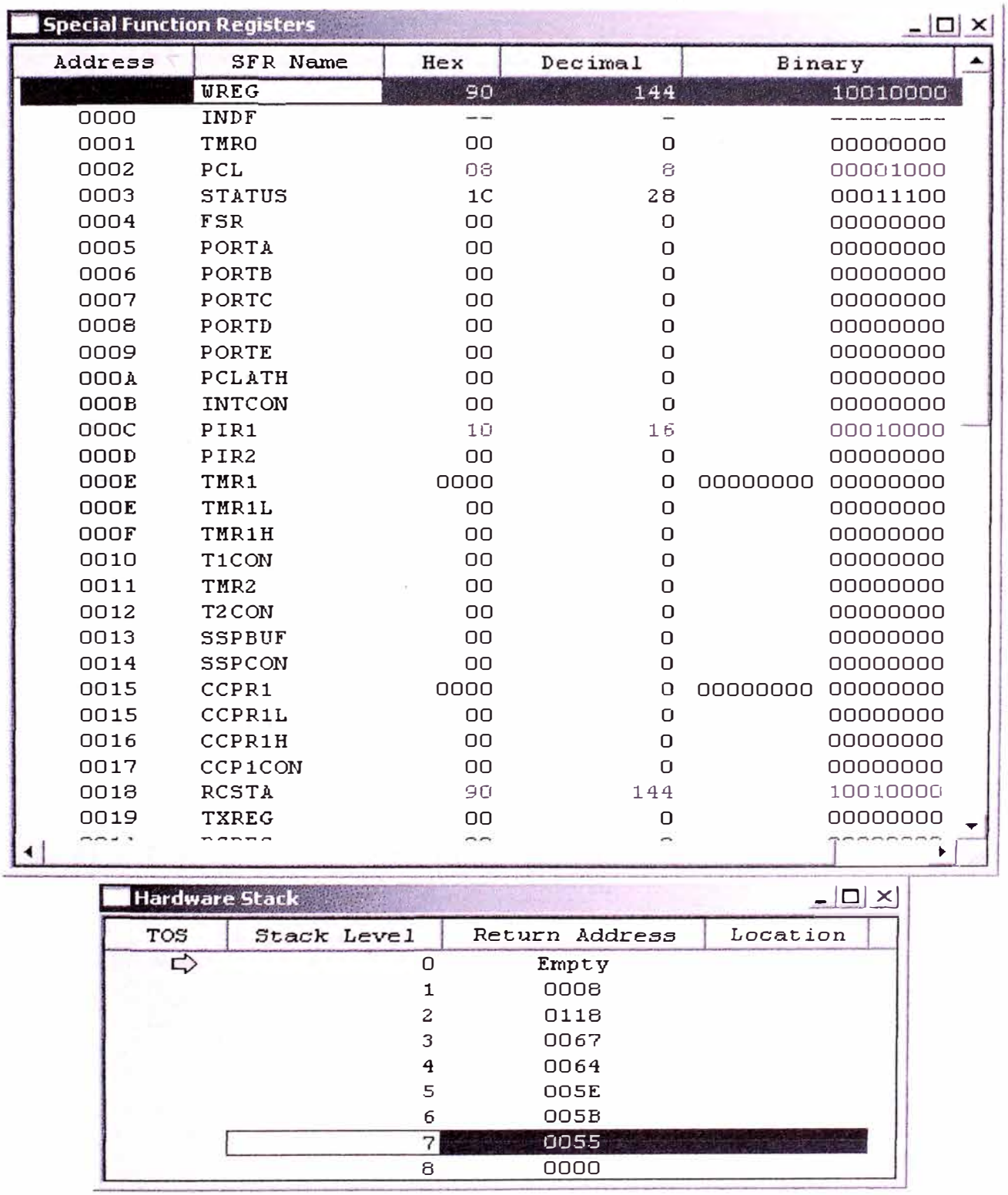
- **Programa desensamblado y variables visualizadas:** Dissassembly nos permite observar al detalle la ejecución del programa, mientras que Watch nos permite ver como cambian las variables.





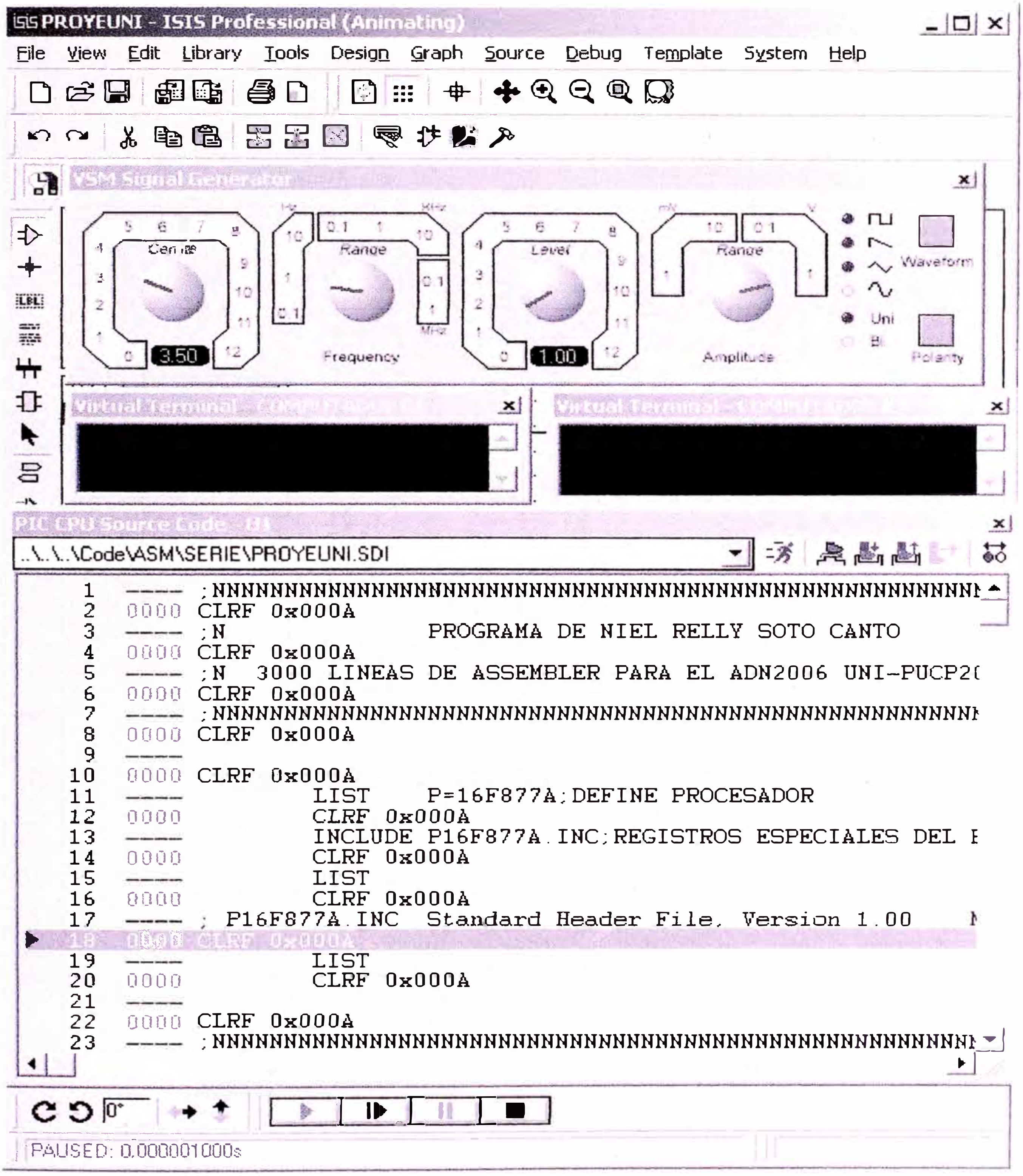
(Figura 4.6.7) Visualización de programa y variables

- **Entrada y salida de datos en forma binaria en memoria:** Visualiza solo los registros principales y el nivel de anidamiento del programa.



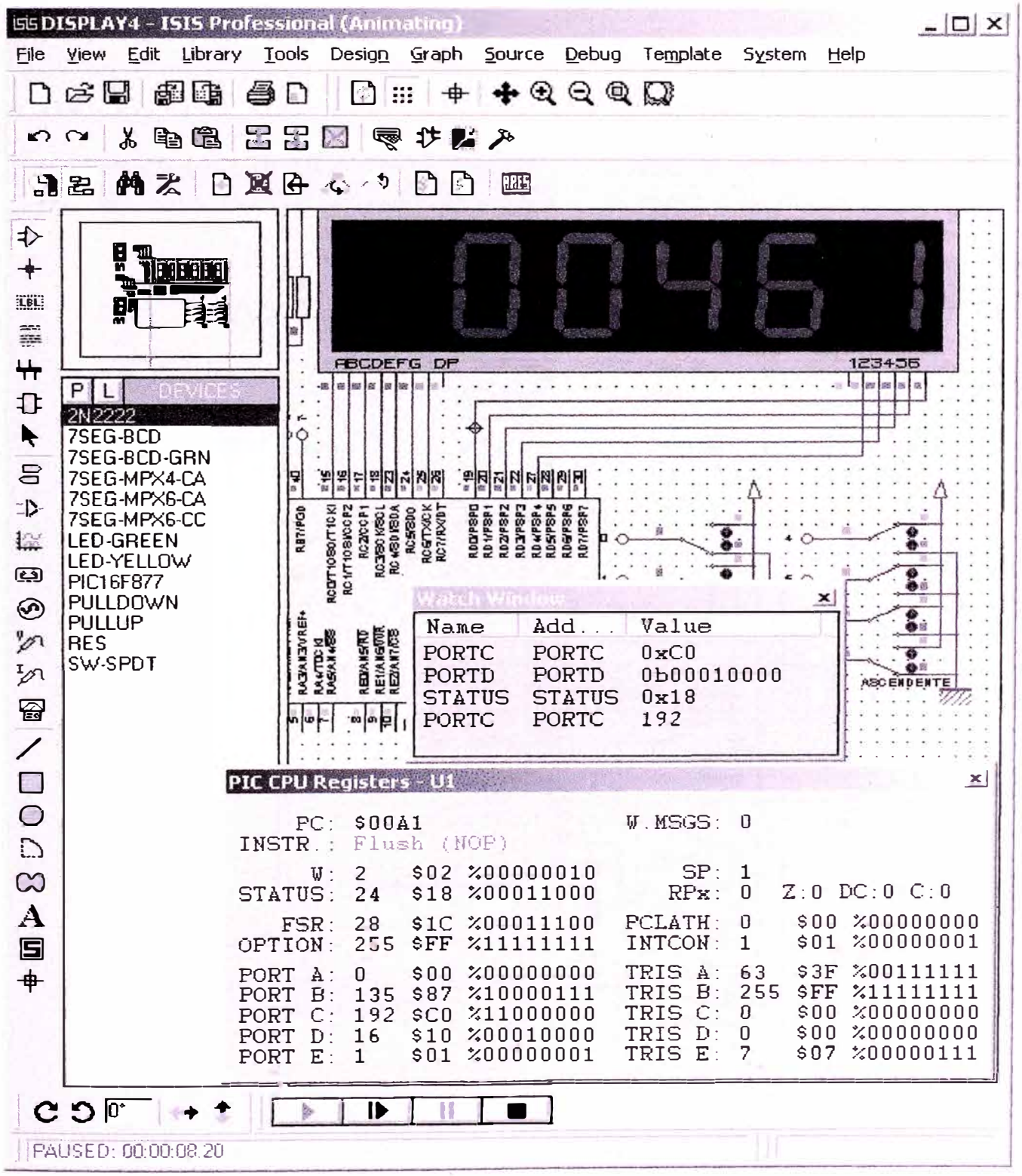
(Figura 4.6.8) Visualización de registros especiales y nivel de pila

- **Simulación en proteus:** Animación de la adquisición de datos a través de un dispositivo, con señales generadas virtualmente.



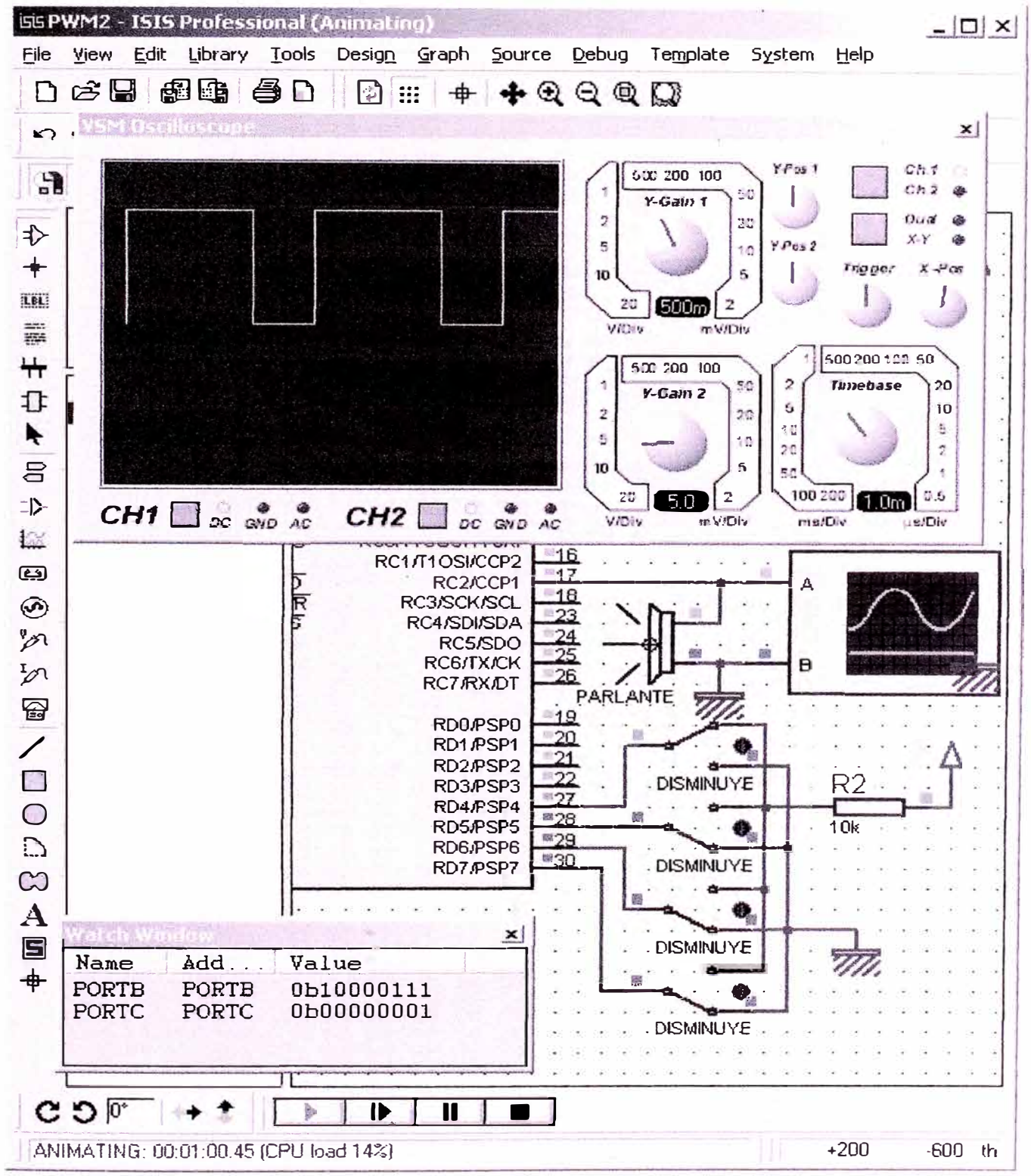
(Figura 4.6.9) Simulación del generador con línea de código ejecutado

En la siguiente figura podemos ver la simulación de entrega de resultados de un proceso de temporizado, útiles en relojes PWM, frecuencímetros, etc.



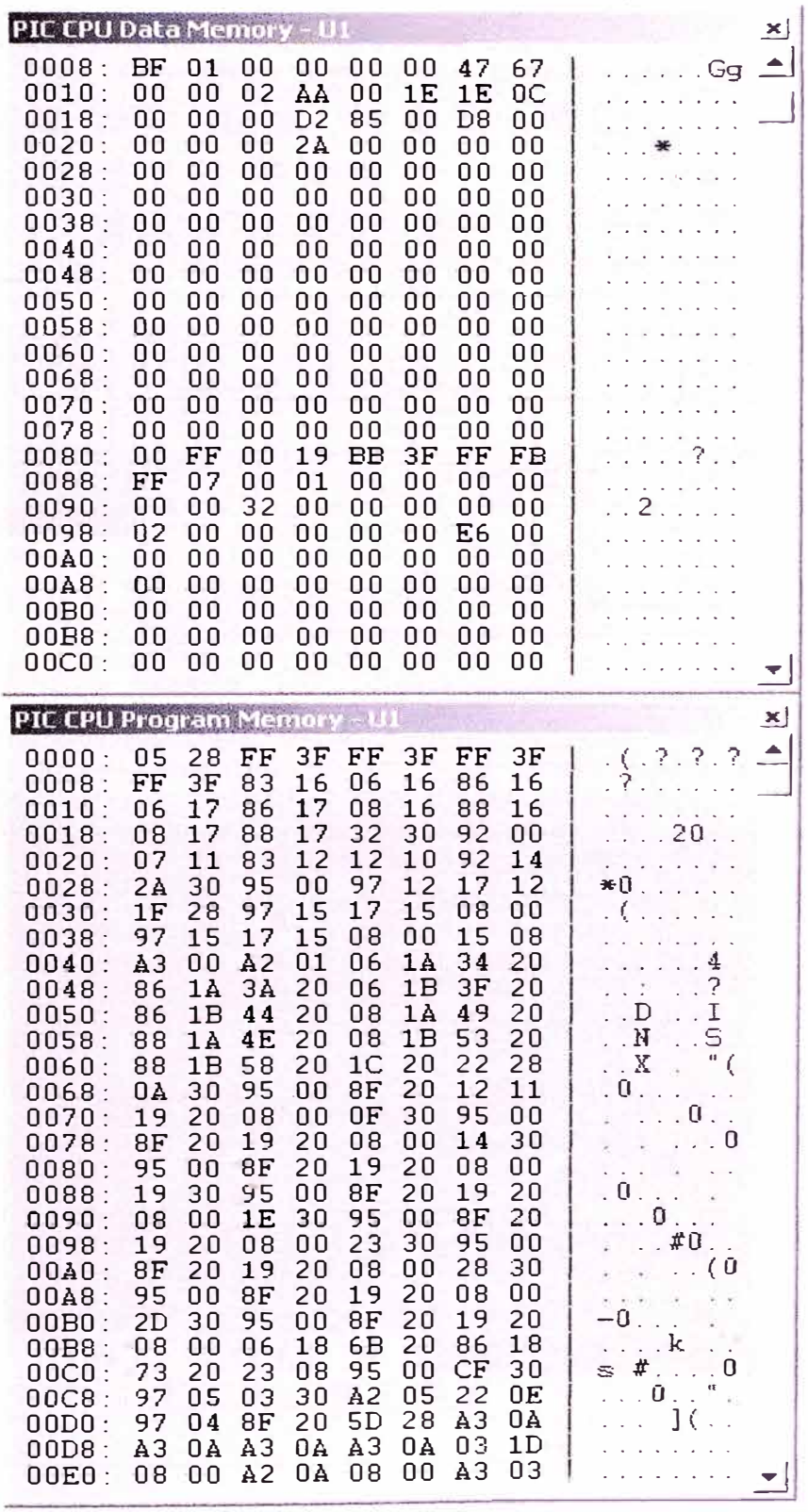
(Figura 4.6.10) Simulación de salida de datos a un temporizador

Acá mostramos la simulación de un piano, donde se requiere entrada de datos a través de pulsadores y salida de datos por el puerto de PWM.



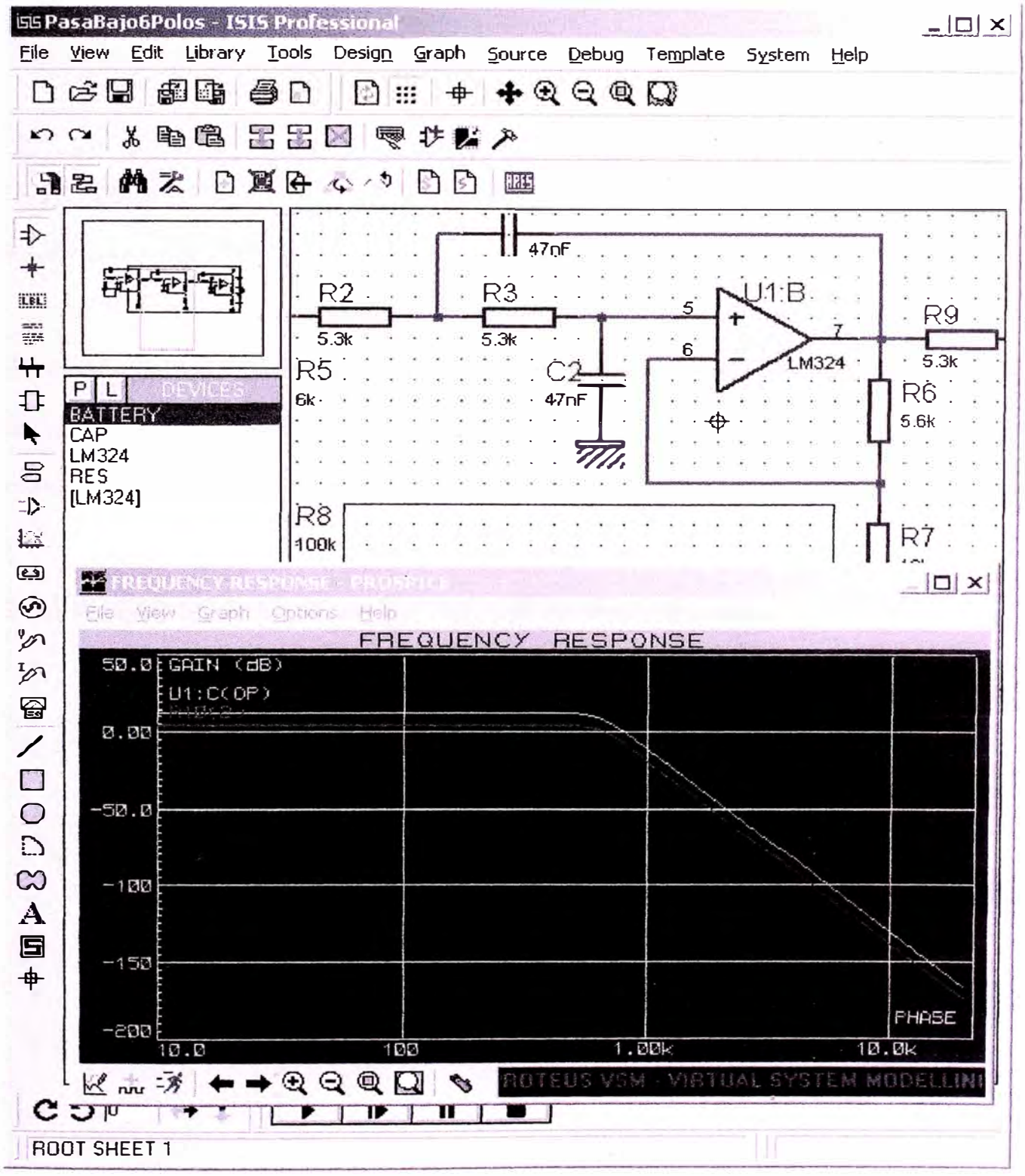
(Figura 4.6.10) Simulación de entrada de datos para un piano

Como todo firmware tiene resultados binarios en sus puertos, acá se puede ver los datos y resultados en forma hexadecimal que sucede en memoria.



(Figura 4.6.11) Resultados de simulación en forma binaria

Adicionalmente se puede simular el filtro que se coloca antes de la adquisición de datos sin errores de ruido y otras perturbaciones.



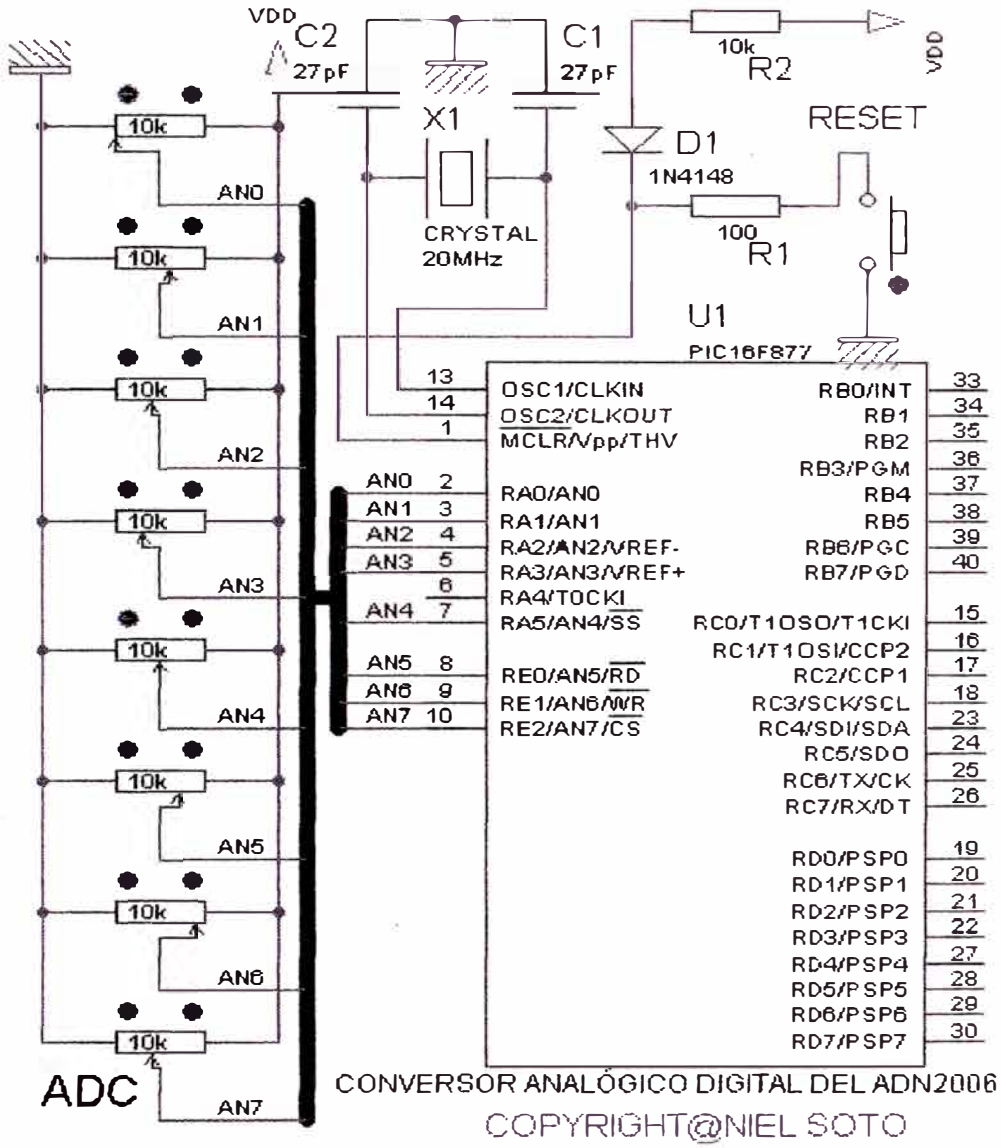
(Figura 4.6.11) Simulación de filtro aplicado antes de la adquisición

CAPÍTULO V

MODULOS DE PROCESAMIENTO DE DATOS CON TECNOLOGÍA WIRE

En este apartado se explicará la configuración del dispositivo a diseñar con la arquitectura de micro escogida anteriormente. También se verá la conexión eléctrica de los módulos.

5.1 Convertidor análogo digital para datos de sensores



(Figura 5.1) Circuito de conexión del ADC

- **Bits de control del ADC**

Se usará el bit 6 del registro PIE1 de la dirección 8Ch.

Bit 6 ADIE: Activa Interrupción de Conversión A/D

1 = Activa interrupción Conversión A/D

0 = Desactiva interrupción Conversión A/D

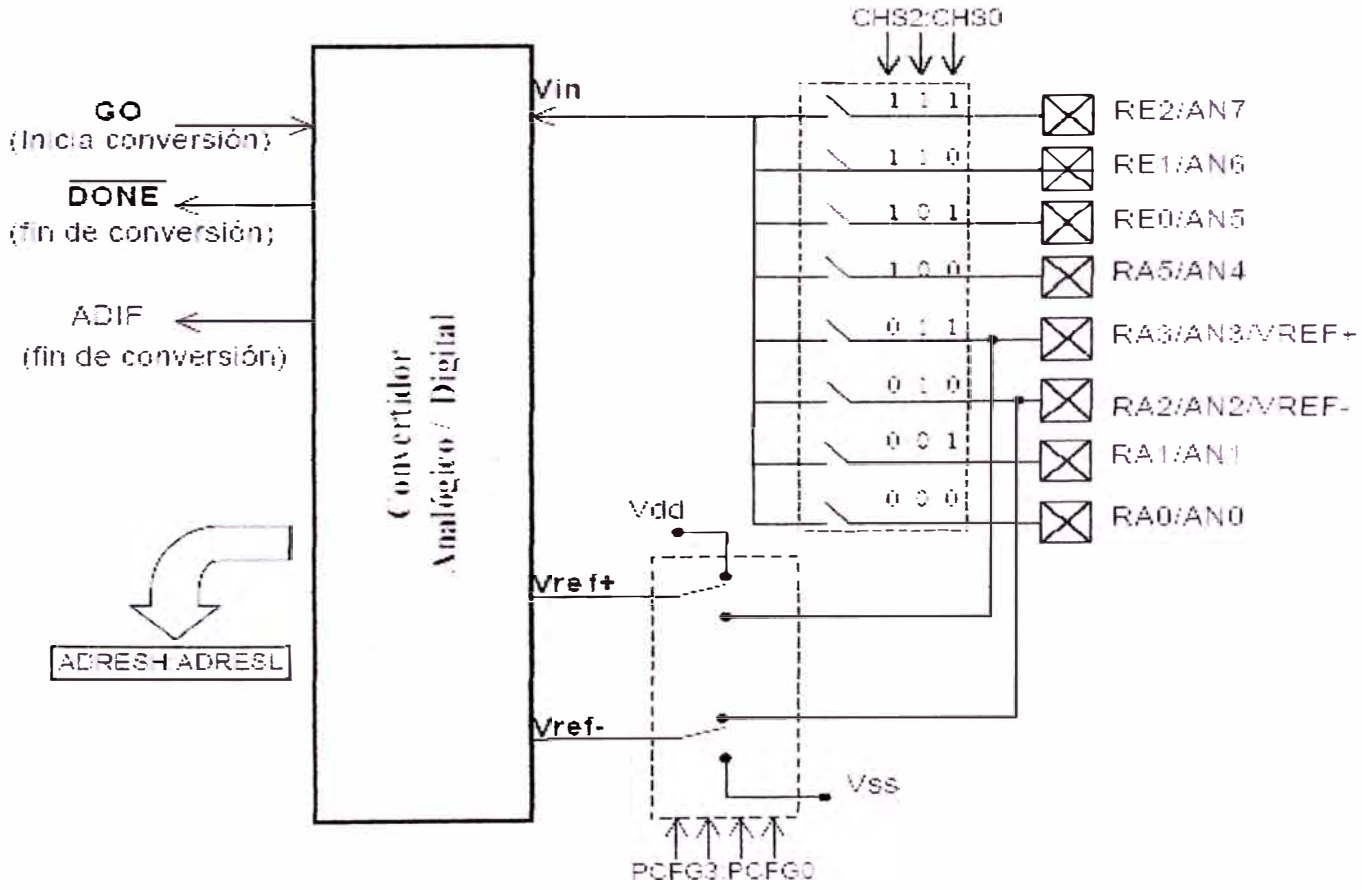
También se usará el bit 6 del registro PIR1 de la dirección 0Ch.

Bit 6 ADIF: Marca de Interrupción de Conversión A/D

1 = La conversión A/D se ha completado

0 = La conversión A/D no es completa

En resumen, para gobernar el funcionamiento del ADC se utilizan dos registros: ADCON0 (0x1F) y ADCON1 (0x9F). El primero, selecciona la velocidad con los bits ADCS1 y ADCS0; también selecciona el canal a convertir con los bits CHS2, CHS1 y CHS0; si se activa el ADC con (ADON=1) entonces se puede iniciar la conversión fijando el bit GO/DONE a 1, que automáticamente se pone a 0 al término de la conversión

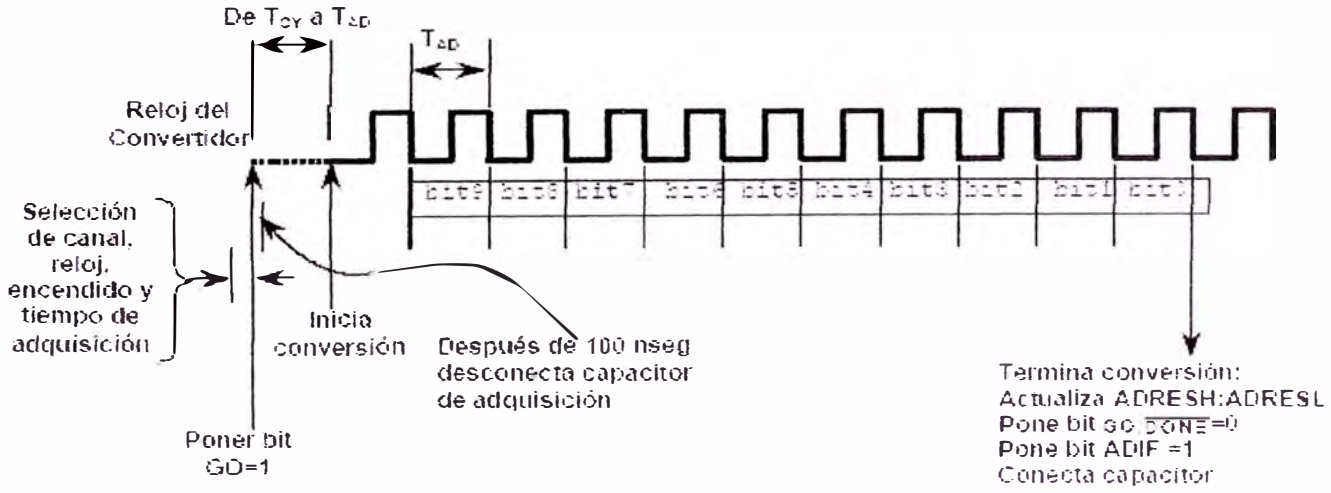


(Figura 5.2) Bits de conversión del ADC

El registro ADCON1 con los bits PCFG3, PCFG2, PCFG1 y PCFG0 establece las entradas que son digitales y analógicas, así como el tipo de tensión de referencia (interna o externa), además se configura el registro ADRES con el bit ADFM en alineación izquierda (ADFM =0 típico para modo 8 bits), así los 6 bits menos significativos de ADRESL son leídos como '0', leeremos el registro ADRESH que contendrá 8 bits perdiendo 2 bits de resolución y en el modo de alineación derecha (ADFM =1 típico para modo 10 bits) los 6 bits más significativos de ADRESH son leídos como '0', leeremos los 2 registros que contendrá los 10 bits del ADC .

- **Tiempo de conversión del ADC**

El tiempo de conversión del ADC por bit es de 1TAD. Siendo la mínima espera de 2TAD entre conversión y conversión. Es decir que para convertir los 10 bits se necesitan 12TAD que en el caso de un cristal de 16MHz se tiene: 32xTOSC = 32x1/(16M) = 2 us.



(Figura 5.3) Tiempo de conversión del ADC

Por lo tanto para convertir un dato se necesitan 12TAD = 24 us y esperar un tiempo de adquisición o de muestreo y carga del condensador de mantenimiento de 19,72 us. Todos estos tiempo sumados nos dan 42,12 us lo que es aproximadamente 50 us o frecuencia de 20 000 Hz (20 Khz.). Esta frecuencia es muy importante ya que limita el ancho de banda de la señal a muestrear a 10 Khz. por el teorema de Nyquist (muestreo).

- **Características del conversor**.-

Las siguientes son algunas de las especificaciones más importantes, y son válidas para los siguientes microcontroladores PIC16F87X-04, PIC16F87X-10, PIC16F87X-20.Esto quiere decir que la tarjeta ADN2006 se puede cambiar de microcontrolador sin tener repercusiones pero eso si con el mismo código que se le entrega. Incluso está la opción para dsPIC pero se debería cambiar algunos pines.

Característica		mínimo	típico	máximo
V_{REF+}-V_{REF-}		2v	-	V_{DD}+0.3v
V_{REF+}		V_{DD}-2.5v	-	V_{DD}+0.3v
V_{REF-}		V_{SS}-0.3v	-	V_{REF+}-2v
Voltaje analógico V_{AIN}		V_{SS}-0.3v	-	V_{REF+}+0.3v
Impedancia de la fuente de señal externa Z_{AIN}		-	-	10 KΩ
Corriente promedio consumida por el convertidor I_{AD}	Estándar	-	220µA	-
	Extendido	-	90µA	-

(Figura 5.4) Características eléctricas

Los PIC 16F877A traen un ADC (conversor análogo-digital) de 10 bits con una tensión de referencia positiva que puede ser interna (VDD) o externa (entra por el pin AN3/Vref+) y también con una referencia negativa que puede ser interna (VSS) o externa (entra por el pin AN2/Vref-).

En cada instante la conversión se realiza con el canal seleccionado (uno de los 8 que trae), depositando el resultado en el registro ADRESH y ADRESL; activando el señalizador ADIF, que provoca una interrupción al término de la conversión si el bit de permiso ADIE está activado. Además, al termino de la conversión el bit GO/DONE# se pone a 0.

5.2 Habilitación del teclado matricial

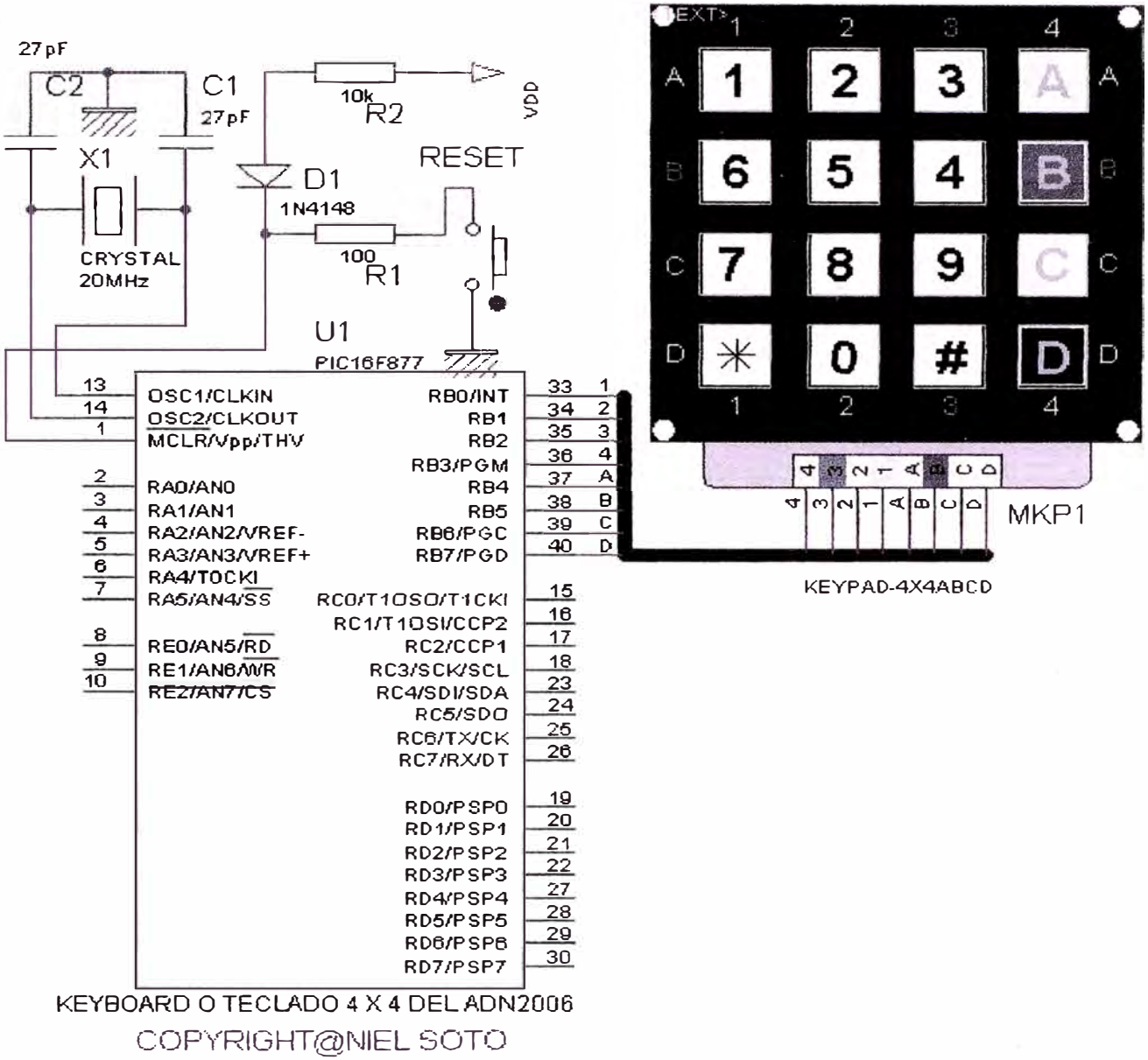
- **Bits de Manejo del teclado**

Para este módulo se usará el bit 7 del OPTION_REG de la dirección 81h.

Bit 7 RBPU: PUERTOB Pull-up Activo

1 = PUERTOB pull-ups son desactivados

0 = PUERTOB pull-ups son activado por individual puerto



(Figura 5.5) Circuito de conexión del teclado

También se usará los bits 3 y 0 del registro **INTCON** de la dirección 0Bh.

Bit 3 RBIE: Activa Interrupción de Cambio en Puerto RB

1 = Activa interrupción de Cambio en Puerto RB

0 = Desactiva interrupción de Cambio en Puerto RB

Bit 0 RBIF: Marca de Interrupción de Cambio en Puerto RB

1 = En cualquier cambio de estado de RB7:RB4; continuara en uno hasta leer el PUERTOB, esto finalizará la incongruencia (debería ser borrado en software).

0 = Ninguno de los pines RB7:RB4 ha cambiado de estado

- **Fases para controlar el teclado**

a) Definir en un instante determinado tan sólo un pin de salida con el valor '0' cargado a una fila del teclado, mientras que los otros tres pines están en modo entrada con Pull-Up activa.

b) Leer el estado de los cuatro bits y compararlo con el valor de salida anterior.

c) Si son iguales entonces no se ha producido ninguna pulsación de tecla por tanto se debe actualizar la fila y regresar al paso a), en el otro caso ir al paso siguiente.

d) En este caso la información de fila y de columna son diferentes por tanto se ha producido la pulsación de una tecla.

Identificar la tecla pulsada mediante la búsqueda del código en una tabla

Determinar cual ha sido la tecla pulsada y ejecutar una acción debida. El PORTB es cargado con el valor 0x00 y cambiando el registro TRISB haciendo que un bit sea de salida por vez, se fuerza a una salida en estado

bajo. Las entradas tienen habilitadas la opción PULL-UP internas. Se utiliza la interrupción del Temporizador TMR0 para realizar el escaneo de las diferentes filas del teclado.

- **Códigos de identificación de tecla.** salida (RB7:4) ingreso (RB3:0).

RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0		
1	1	1	**0**	1	1	0	0	'1'	EC
				1	0	0	0	'2'	E8
				1	0	1	0	'3'	EA
				0	1	1	0	'A'	E6
1	1	**0**	1	1	1	**0**	0	'4'	DC
				1	0	**0**	1	'5'	D9
				1	0	**0**	0	'6'	D8
				0	1	**0**	0	'B'	D4
1	**0**	1	1	1	**0**	0	0	'7'	B8
				1	**0**	0	1	'8'	B9
				1	**0**	1	0	'9'	BA
				0	**0**	1	0	'C'	B2
0	1	1	1	**0**	1	0	0	'D'	74
				0	0	0	1	'0'	71
				0	0	1	0	'*'	72
				0	1	1	0	'#'	76

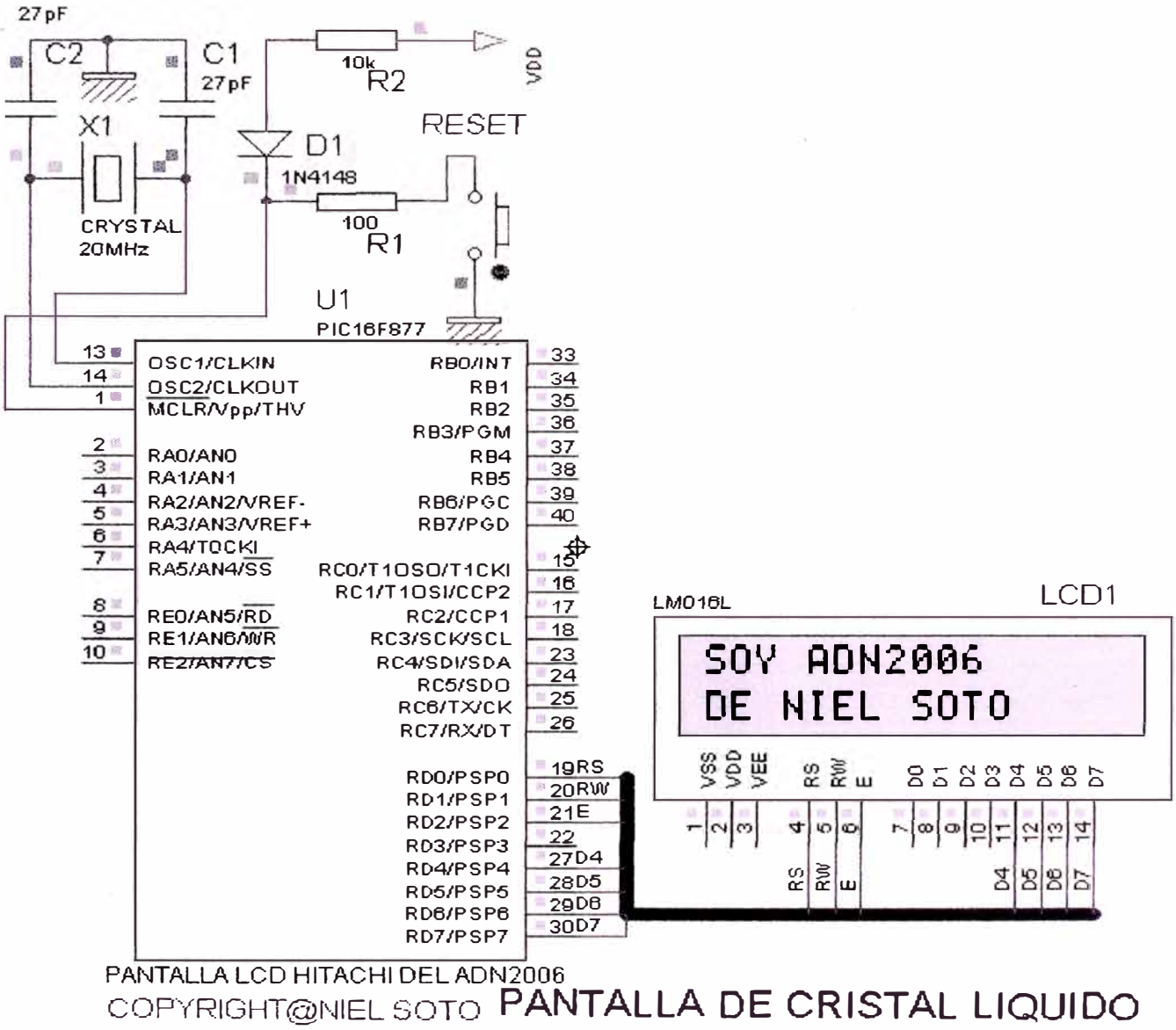
(Figura 5.6) Códigos de tecla

5.3 Visualización en la pantalla LCD

- **Bits de activación del LCD**

Para el LCD se usará el puerto D cuyos pines serán representados de la siguiente forma: RS, EN, WR, D7, D6, D5, D4 serán RD0, RD1, RD2, RD3, RD7, RD6, RD5, RD4.

El algoritmo de uso será:
LCD Configurar (PORTD, 1, 2, 0, 3, 5, 4, 6);



(Figura 5.7) Circuito de conexión del LCD

- **Características del Controlador HD44870**

El controlador HD 44870 es un manejador de pantalla de cristal líquido en formato de matriz de puntos con la posibilidad de mostrar símbolos y caracteres alfanuméricos. Se puede conectar directamente al control de un microprocesador o microcontrolador en formato de 4 bits u 8 bits. Todas las funciones requeridas para el control de la pantalla de cristal líquido se encuentran incorporadas dentro del chip.

Dentro de las características más importantes de este controlador se pueden mencionar:

-Capacidad de interfaz con uP de 4 u 8 bits
-RAM de datos para la pantalla 80 x 8 bits (80 caracteres máx.)
-ROM generador de caracteres
-Fuente de letra: 5 x 7 puntos: 160 caracteres
-Fuente del letra: 5 x 10 puntos: 32 caracteres
-Ambos bancos de memoria pueden ser leídos por el microcontrolador
-Amplia gama de funciones: Aclarado de pantalla, prender/ apagar el LCD, prender/ apagar cursor, desplazamiento del cursor.
-Reset interno automático durante el encendido (Internal Reset Circuit)

(Figura 5.8) LCD con el controlador HD44870

- **Inicialización del LCD**

La manera de inicializar el módulo LCD basado en el controlador HD44870 es automático al energizar el circuito, otra forma es a través de la ejecución de un programa.
El controlador HD 44870 se inicia automáticamente cuando se enciende la alimentación pro ello se le llama inicializar por Circuito de Reset Interno. Durante este proceso se realizan las siguientes instrucciones. La marca de ocupado (BF) se mantiene en estado ocupado hasta que la inicialización finaliza. (BF=1)

1) Aclarado de la pantalla
2) Fijado de la función:

DL = 1: Interfaz de datos de 8 bits

N = 0: LCD de una línea

F = 0: Fuente de letra de 5 x 7 puntos

3) Control del Display ON/OFF:

D = 1: Display ON

C = 1: Cursor ON

B = 1: Blink ON

4) Fijado del modo de entrada:

I/D = 1: incrementa (+1)

S = 0: No desplazamiento

Si no se consiguen de manera correcta las condiciones para la fuente de alimentación, se requiere de la inicialización por instrucciones de tal forma que el PIC le envía la data tal y como se hace en la forma anterior.

5.4 Interconexión para el procesamiento de datos del USART

- **Bits usados en el USART**

En este módulo se usarán los bits 5 y 4 del registro PIE1 de la dirección 8Ch.

Bit 5 RCIE: Activa Interrupción de Recepción USART

1 = Activa Interrupción de Recepción USART

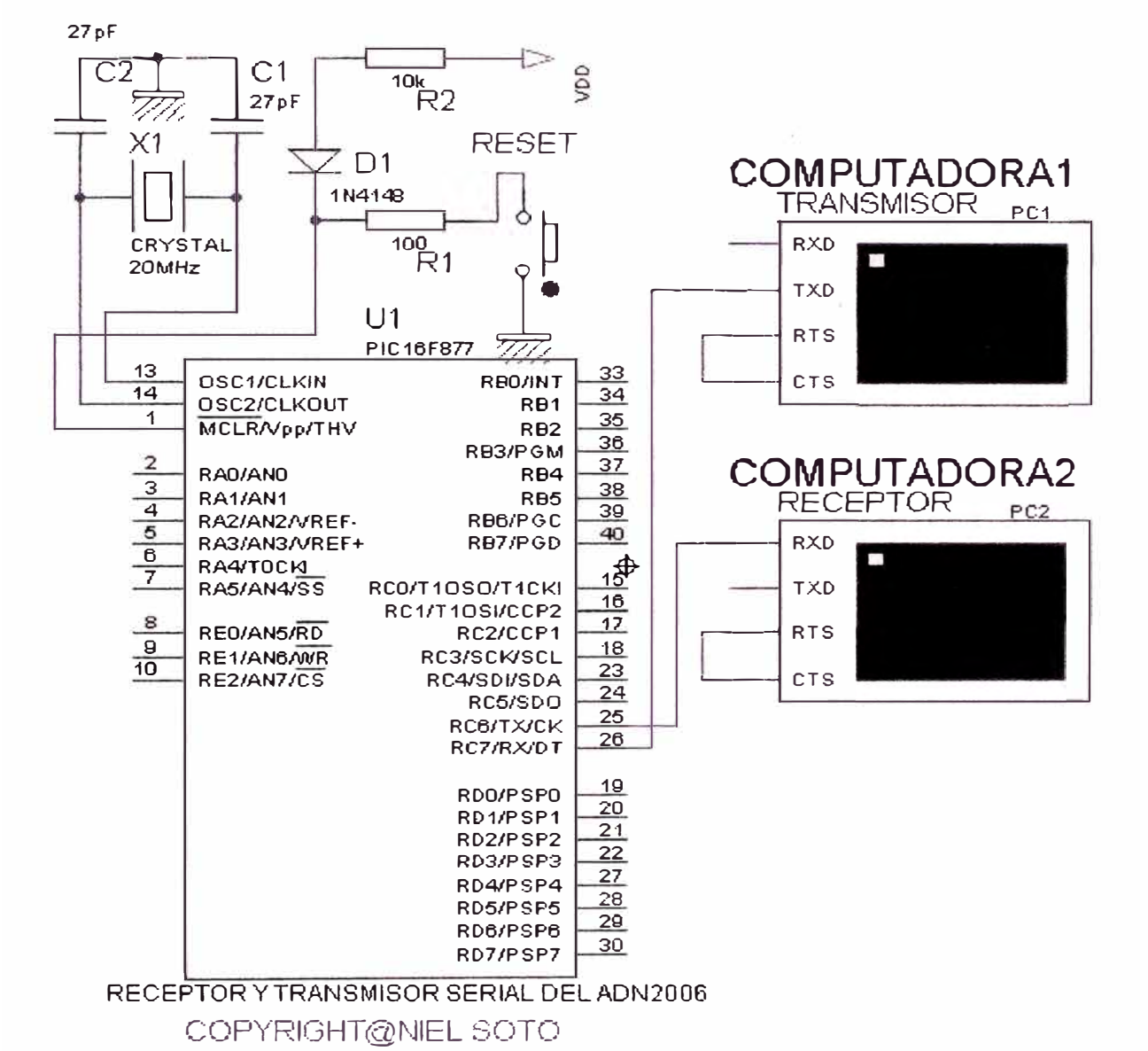
0 = Desactiva Interrupción de Recepción USART

Bit 4 TXIE: Activa Interrupción de Transmisión USART

1 = Activa Interrupción de Transmisión La USART

0 = Desactiva Interrupción de Transmisión La USART

También se usarán los bits 8 y 4 del registro PIR1 de la dirección 0Ch.



(Figura 5.9) Circuito de conexión serial

Bit 5 RCIF: Marca de Interrupción de Recepción USART

1 = El buffer de Recepción USART esta lleno

0 = El buffer de Recepción USART esta vacío

Bit 4 TXIF: Marca de Interrupción de transmisión USART

1 = El buffer de transmisión USART esta vacío

0 = El buffer de transmisión USART esta lleno

- **Control del módulo USART**

El USART del PIC puede ser configurado para operar en tres modos:

Modo Asíncrono (full duplex (transmisión y recepción simultáneas)),

Modo Síncrono – Maestro (half duplex)

Modo Síncrono – Esclavo (half duplex)

En el modo asíncrono el USART usa un formato estándar NRZ asíncrono, el cual para la sincronización usa: 1 bit de inicio (I), 8 o 9 bits de datos y 1 bit de paro (P). Mientras no se están transmitiendo datos el USART envía continuamente un bit de marca. Este modo se selecciona limpiando el bit SYNC del registro TXSTA (98H) y es deshabilitado durante el modo SLEEP.

Cada dato es transmitido y recibido comenzando por el LSB. El hardware no maneja bit de Paridad, pero el noveno bit puede ser usado para este fin y manejado por software.

- **Partes del módulo USART**

El módulo Asíncrono de la USART consta de 4 módulos fundamentales:

-El circuito de muestreo

-El generador de frecuencia de transmisión (Baud Rate)

-El transmisor asíncrono

-El receptor asíncrono.

El circuito de muestreo: El dato en la patita de recepción (RC7/RX/DT) es muestreado tres veces para poder decidir mediante un circuito de mayoría, si se trata de un nivel alto o un nivel bajo.

El Generador de Baud Rate (BRG): Este generador sirve tanto para el modo síncrono como el asíncrono y consiste de un contador/divisor de frecuencia de 8 bits controlado por el registro SPBRG (99H). De tal manera que la frecuencia de transmisión se calcula de acuerdo a la siguiente tabla:

SYNC	BRGH=0 (baja velocidad)	BRGH=1 (Alta velocidad)
0 (modo asíncrono)	Baud rate=Fosc/(64(X+1))	Baud rate=Fosc/(16(X+1))
1 (modo síncrono)	Baud rate=Fosc/(4(X+1))	-

(Figura 5.10) Selector de velocidad

Debido a que el divisor es de 8 bits, no se puede tener cualquier velocidad de transmisión deseada, ya que X se deberá redondear al entero más cercano.

El transmisor asíncrono: Es usado en la transmisión y estos son sus bits de control:

CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D

Receptor asíncrono: Es usado en la recepción y estos son sus bits de control:

SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D

En las tablas de velocidad se muestran algunos valores baudio estándares, el divisor necesario (X=SPBRG) bajo diferentes frecuencias Fosc y el error producido en porcentaje

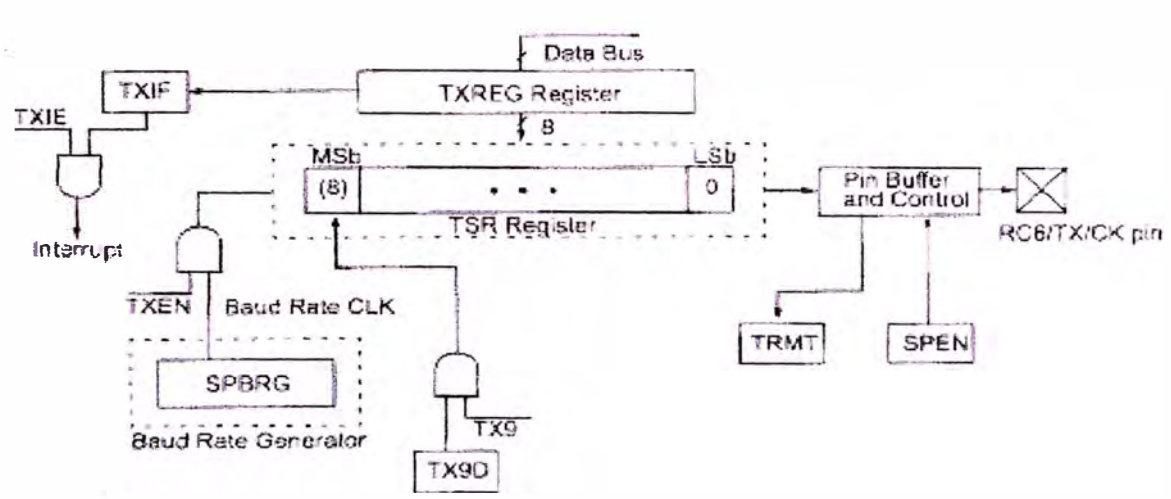
BAUD RATE (K)	Fosc = 20 MHz			Fosc = 16 MHz			Fosc = 10 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	-	-	-	-	-	-	-	-	-
1.2	-	-	-	-	-	-	-	-	-
2.4	-	-	-	-	-	-	2.441	1.71	255
9.6	9.615	0.16	129	9.615	0.16	103	9.615	0.16	64
19.2	19.231	0.16	64	19.231	0.16	51	19.531	1.72	31
28.8	29.070	0.94	42	29.412	2.13	33	28.409	1.36	21
33.6	33.784	0.55	36	33.333	0.79	29	32.895	2.10	18
57.6	59.524	3.34	20	58.824	2.13	16	56.818	1.36	10
HIGH	4.883	-	255	3.906	-	255	2.441	-	255
LOW	1250.000	-	0	1000.000		0	625.000	-	0

(Figura 5.11) Velocidad con una determinada Fosc.

- **Conexión e Interfaz serial**

Transmisor asíncrono

En la siguiente figura se muestra el diagrama de bloques del transmisor de la USART.



(Figura 5.12) Diagrama de bloques del transmisor USART.

El corazón de este módulo es el registro de corrimiento (transmit shift register, TSR). La única manera de acceder al registro TSR es a través del registro TXREG (19H).

Para transmitir un dato, el programa deberá ponerlo primero en el registro TXREG. En cuanto el TSR termina de enviar el dato que tenía (en cuanto transmite el bit de paro) lee el dato contenido en TXREG (si hay alguno) esto ocurre en un ciclo TCY. En cuanto el dato de TXREG es transferido al TSR el TXREG queda vacío esta condición es indicada mediante el bit bandera TXIF (que es el bit 4 del registro PIR1 (0Ch)), el cual se pone en alto. Este bit NO puede ser limpiado por software, sólo dura un instante en bajo cuando se escribe un nuevo dato a TXREG. Si se escribe un dato seguido de otro (back to back) a TXREG el primero se transfiere inmediatamente a TSR y el otro tiene que esperar hasta que el TSR termine de enviar el bit de Stop del primero. Durante esta espera TXIF permanece en bajo.

Existe otro bit, llamado TRMT (TXSTA<1>), el cual muestra el estado del TSR. TRMT se pone en alto cuando TSR está vacío, y en bajo cuando TSR está transmitiendo un dato. Mientras que TXIF puede generar una interrupción TRMT no lo puede hacer, TRMT está pensado para ser consultado por "poleo" (sin usar interrupciones).

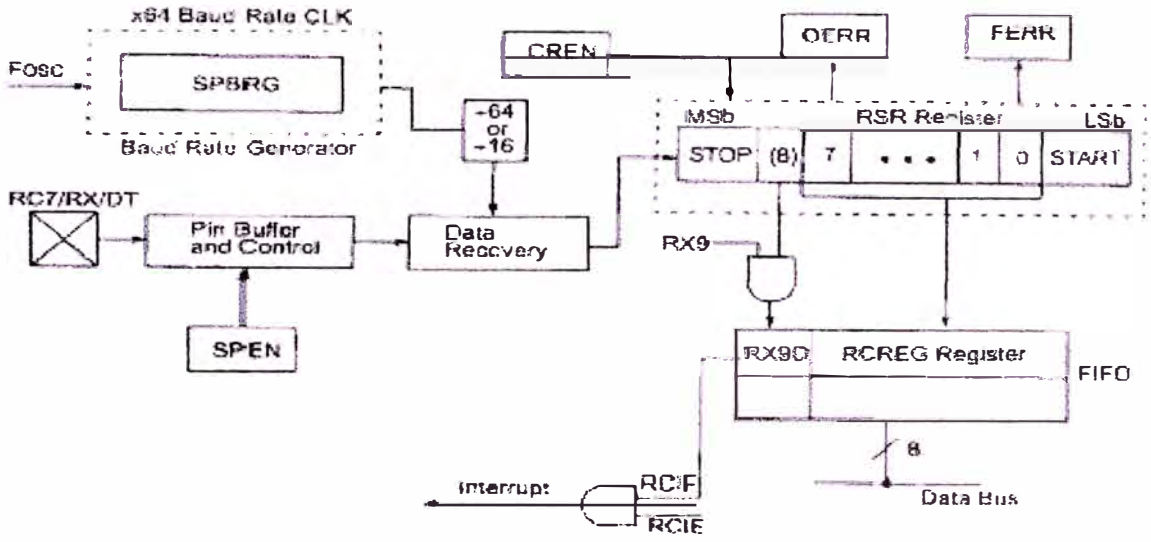
Para habilitar el módulo de transmisión es necesario poner en alto el bit TXEN (TXSTA<5>), mientras no se habilite el módulo, la patita de transmisión (RC6/TX/CK) se mantiene en alta impedancia. Si TXEN es deshabilitada a la mitad de una transmisión, está será abortada y el transmisor será reseteado.

Si se está usando un noveno bit TX9 (TXSTA<6>), éste deberá ser escrito antes de escribir los 8 bits restantes a TXREG, ya que en cuanto se escribe un dato a este registro inmediatamente es transferido a TSR (si éste está vacío).

Receptor asíncrono

El módulo de recepción es similar al de transmisión, en la siguiente figura se muestran los bloques que lo constituyen. Una vez que se ha seleccionado el modo asíncrono, la recepción se habilita poniendo en alto el bit CREN

(RCSTA<4>) El dato es recibido mediante la línea RC7/RX/DT, la cual maneja un registro de corrimiento de alta velocidad (16 veces el Baud rate).



(Figura 5.13) Diagrama de bloques del receptor USART.

El corazón del receptor es el registro de corrimiento RSR. Este registro no es accesible por software, pero, cuando el dato recibido se ha completado (se ha recibido el bit de Stop) el dato de RSR es transferido automáticamente al registro RCREG (1Ah) si éste está vacío y al mismo tiempo es puesto en alto la bandera de recepción RCIF (PIR1<5>).

La única forma de limpiar la bandera RCIF es leyendo el los datos del registro RCREG. El registro RCREG puede contener hasta dos datos, ya que es un buffer doble que funciona como una cola de dos posiciones.

Si las dos posiciones del registro RCREG están llenas (no han sido leídas) y se detecta el bit de Stop de un tercer dato de recepción, lo cual ocasiona un transferencia automática del dato recibido a RCREG, esto destruirá el primer

dato recibido y activará el indicador de sobre escritura OERR (RCSTA<1>). Para evitar esto, se deberán leer los dos datos en RSREG haciendo dos lecturas consecutivas.

La única manera de limpiar el bit OERR una vez que ha sido activado es reseteando el módulo de recepción (limpiando CREN y volviéndolo a poner), si no se limpia OERR se bloquea la transferencia de datos de RSR a RCREG y no puede haber más recepción de datos.

Si se detecta un bit nivel bajo en la posición del bit de stop se pone el indicador de error de encuadre (frame error) FERR RCSTA<2>. Tanto este indicador como el noveno bit RX9D de los datos están en una cola de dos posiciones al igual que los datos recibidos, de manera que al leer RCREG se actualizan FERR y RX9D con nuevos valores, por lo cual estos bits deberán ser leídos antes de leer RCREG para no perder su información.

- **Configuración en el puerto serie**

La inicialización del módulo de transmisión consiste en los siguientes pasos:

1. Inicializar baud rate escribiendo al registro SPBRG el divisor adecuado y opcionalmente al bit BRGH.

2. Habilitar comunicación asíncrona limpiando el bit SYNC y poniendo el bit SPEN.

3. Si se van a usar interrupciones, poner el bit TXIE (PIE<4>).

4. Poner el bit TX9 si se desea transmitir datos de 9 bits

5. Habilitar transmisión poniendo el bit TXEN, lo cual pondrá el bit TXIF.

6. Colocar el noveno bit del dato en TX9D si se están usando datos de 9 bits.

7. Cargar el dato al registro TXREG (inicia la transmisión).

La inicialización del módulo de recepción es como sigue:

1. Inicializar el baud rate escribiendo al registro SPBRG el divisor adecuado y opcionalmente al bit BRGH.

2. Habilitar el puerto serie asíncrono limpiando el bit SYNC y poniendo el bit SPEN.

3. Si se van a usar interrupciones, poner el bit RCIE (PIE<5>).

4. Si se desea recepción de datos de 9 bits se deberá poner el bit RX9 (RCSTA<0>).

5. Habilitar la recepción poniendo el bit CREN (RCSTA<4>)

6. El bit RCIF se pondrá cuando la recepción de un dato se complete y se generará una interrupción si RCIE está puesto.

7. Leer el registro RCSTA para obtener el noveno bit (si se están recibiendo datos de 9 bits) o para determinar si ha ocurrido un error de recepción.

8. Leer los 8 bits del dato recibido leyendo el registro RCREG.

9. Si ocurrió algún error este se limpia al limpiar el bit CREN, el cual deberá volver a ponerse si se desea continuar la recepción.

5.5 Temporizadores en dispositivos de adquisición de datos

- **Bits de configuración del temporizador**

Para el temporizador se usan los bits 5 y 2 del INTCON de la dirección 0Bh.

Bit 5 T0IE: TMR0 Activa Interrupción de Desbordamiento

1 = Activa la interrupción TMR0

0 = Desactiva la interrupción TMR0

Bit 2 T0IF: Marca de Interrupción de Desbordamiento TMR0

1 = Registro TMR0 ha desbordado (debería ser borrado en software)

0 = Registro TMR0 no ha desbordado

Para el PWM se usará el bit 2 del PIE1 y PIR1 de la dirección 8Ch y 0Ch.

Bit 2 CCP1IE: Activa Interrupción de modulo CCP1

1 = Activa Interrupción de modulo CCP1

0 = Desactiva Interrupción de modulo CCP1

Bit 2 CCP1IF: Marca de Interrupción en CCP1

En modo captura:

1 = La captura de Registro TMR1 ha ocurrido (debería borrarse en software)

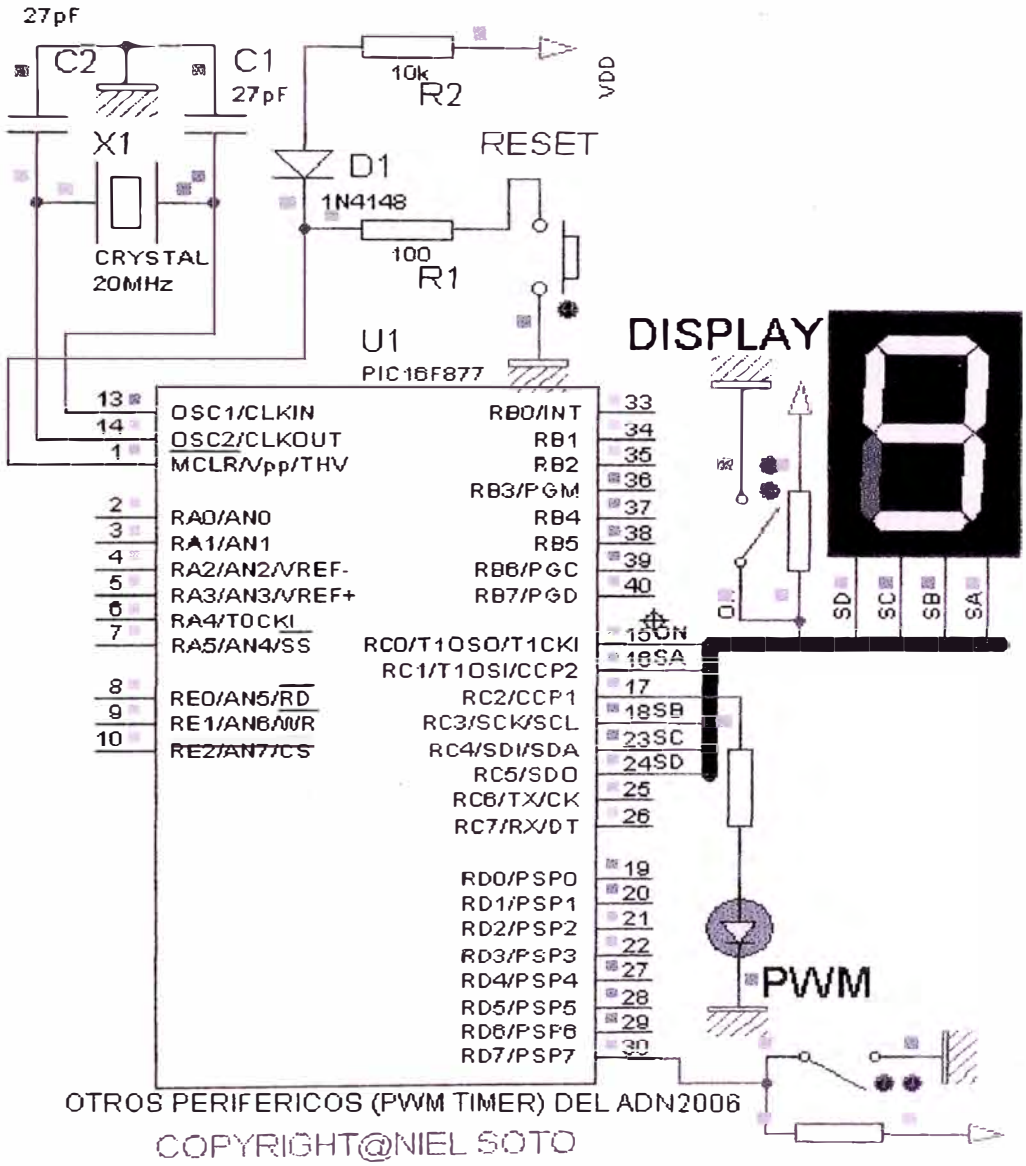
0 = No ocurre una captura de Registro TMR1

En modo Compare:

1 = Un emparejamiento de Registro TMR1 ha ocurrido (se borra en software)

0 = No ocurre un emparejamiento de Registro TMR1

En modo PWM: Nunca usado en este modo

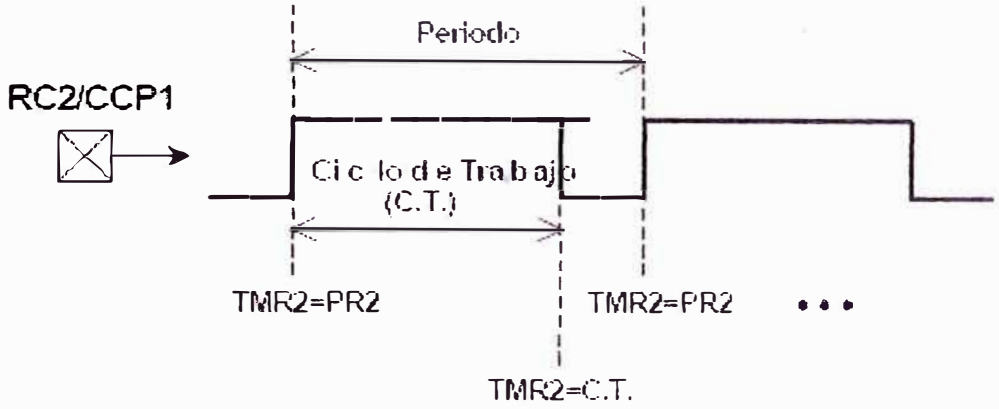


(Figura 5.14) Circuito de conexión PWM y Timer

- **Utilización del PWM**

En este modo se puede producir una salida de frecuencia fija seleccionable modulada en ancho de pulso (o ciclo de trabajo) con una resolución de 10

bits, a través de la patita RC2/CCP1, como se muestra en la figura siguiente Debido a que la patita CCP1 está multiplexado con RC2, este bit del puerto C deberá ser configurado como salida (TRISC<2>=0) para poder usar la salida CCP1.



(Figura 5.15) Diagrama de tiempo del PWM

- **Control del Periodo del PWM**

Para especificar el periodo del PWM se usa el registro PR2, de manera que el valor del periodo será:

$$\text{Periodo PWM} = (PR2+1)*4*TOSC*M$$

Donde 1/M es el valor del PRE-escalador del Timer 2.

Cuando el valor en TMR2 alcanza el valor PR2 los siguientes tres eventos ocurren en el siguiente ciclo (Ver figura anterior):

-El registro TMR2 es limpiado

-La patita CCP1 es puesta en alto (Excepto si el ciclo de Trabajo del PWM vale cero).
-El Ciclo de Trabajo es cargado de CCPR1L (15h) a CCPR1H (16h).

De esta manera, de acuerdo a la figura anterior, el siguiente valor de comparación para TMR2 en el comparador de 10 bits es el Ciclo de Trabajo, el cual al alcanzarse limpiará la patita CCP1.

- **Control del Ciclo de Trabajo del PWM**

El ciclo de Trabajo se especifica escribiendo un valor de 10 bits al registro CCPR1L (los 8 bits más significativos (MSB)) y los dos bits menos significativos (LSB) a CCP1CON<5:4> este valor de 10 bits lo representaremos como CT=CCPR1L:CCP1CON<5:4>. El valor de tiempo que dura el ciclo de trabajo para un valor del PRE-escalador de 1/M, se calcula de acuerdo a la siguiente ecuación: TPWM = CT*TOSC*M.

El valor que determina la duración de C.T. del PWM no es el cargado en CT (CCPR1L), sino en CCPR1H, el cual sólo se actualiza copiando el valor de CT en el momento en que TMR2 alcanza el valor de PR2 (es decir, cada vez que se completa un periodo). Por ello, aunque CCPR1L puede ser escrito en cualquier momento, el Ciclo de Trabajo solo se actualiza hasta que termina el periodo que está en transcurso.

TPWM = (power (2, n))* TOSC*M

Sin embargo, dependiendo del valor de Ciclo de trabajo máximo (TPWM) deseado, no será posible realizar las 2r divisiones y por lo tanto no se podrán usar los r bits de resolución. O al revés, si se elige una resolución deseada r no será posible tener cualquier Ciclo de Trabajo máximo (TPWM) Deseado.

- **Secuencia de configuración del PWM**

Pasos para realizar la configuración inicial del PWM:

1. Establecer el periodo del PWM escribiendo al registro PR2.
2. Establecer el Ciclo de Trabajo del PWM escribiendo al registro CCPR1L y a los bits CCP1CON<5:4>.
3. Configurar como salida la patita CCP1, limpiando el bit TRISC<2>.
4. Configurar el PRE-escalador del Timer 2 y habilitar el Timer 2, escribiendo al registro T2CON.
5. Configurar el módulo CCP1 para operación PWM. Poniendo en alto los bits CCP1CON <2:3>.

CAPÍTULO VI

VISUALNIEL2 COMO UTILIZACIÓN DE TECNOLOGÍA WIRE

En este capítulo se explica de manera detallada el desarrollo y funcionamiento del software VisualNiel2 luego de obtener la data de un dispositivo DAQ que a su vez obtuvo las señales ECG debidamente acondicionadas (amplificadas y filtradas) las cuales deberán ser convertidas a formato digital (conversión analógico/ digital) para su posterior transmisión serial.

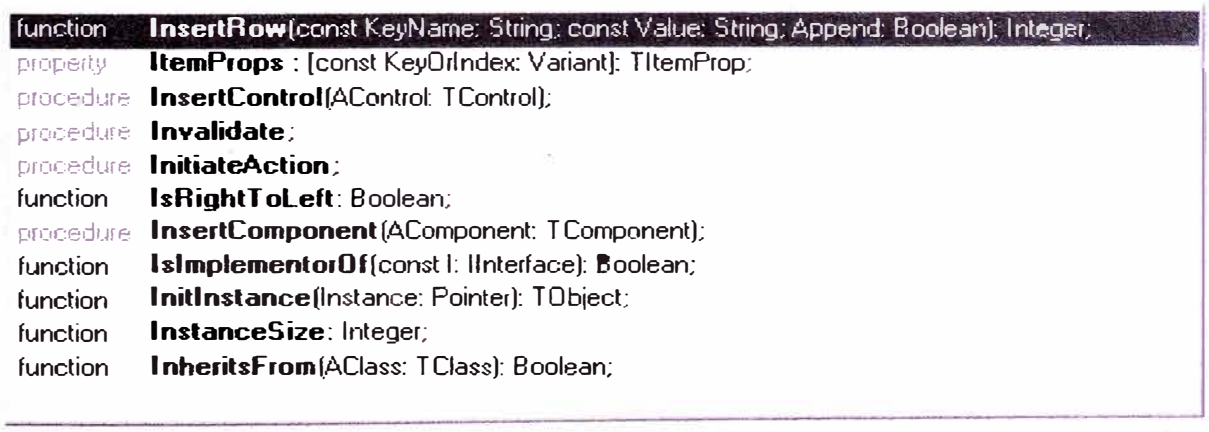
6.1 Criterios a tomar en cuenta para VISUALNIEL2

6.1.1 Parámetros para elección del entorno de desarrollo

Para la elección del software se tuvo en cuenta criterios de facilidad y rapidez tanto en la hora de diseño como en el momento de funcionamiento del programa. Es decir un programa que no tenga redundancia de código, se considerará un programa óptimo, por ende el usuario ejecutara un software

liviano y de esa manera el microprocesador de la computadora responderá más rápido sin posibilidades de colgarse.

El IDE (interfaz gráfico de usuario) elegido es el de la compañía Borland por la manera eficaz de división modular de código con opciones de Ayuda en tiempo de ejecución. Por ejemplo si uno escribe un comando, con solo poner la primera letra es suficiente para que la ayuda contextual aparezca y con la simple presión de la tecla enter se escoge el comando deseado. Esto ahorra el tiempo de escritura en el código, de escribir 20 letras a 1.

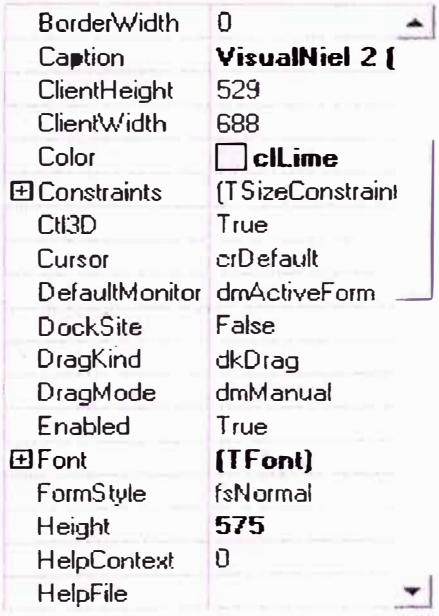


(Figura 6.1.1) Ayuda de escritura automática de código

Otro de los parámetros para la elección del software fue la rapidez con que el programador puede re-usar código teniendo a la vista la gran cantidad de componentes en una paleta separada en páginas para mayor orden. (Estilo único de los productos de Borland).

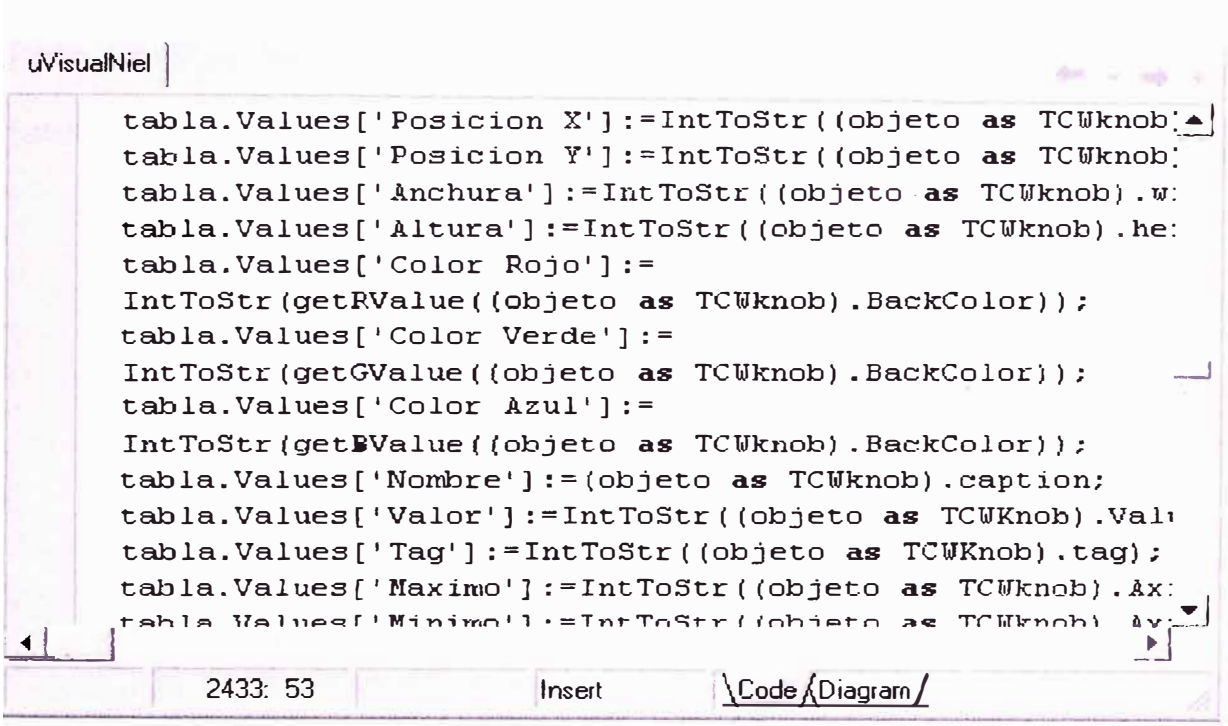
(Figura 6.1.2) Paleta de componentes

También es necesario mencionar el inspector de propiedades de un objeto o componente el cual permite modificar rápidamente las características del componente, tales como color, tamaño, posición, nombre, valor, etc.



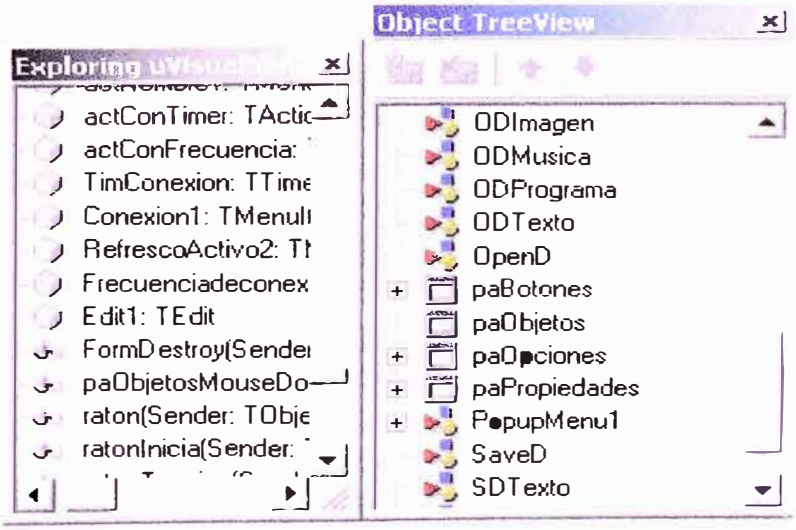
(Figura 6.1.3) Inspector de propiedades

La ventana de código es un editor de código avanzado, con las variantes de colores necesarios para la distinción de propiedades y comandos.



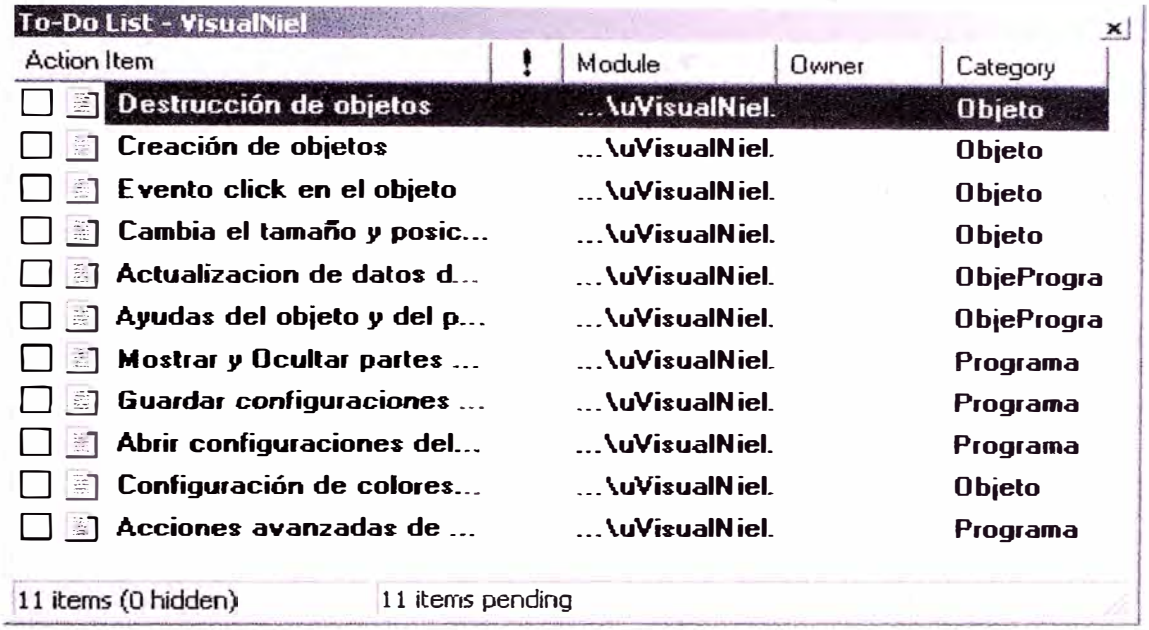
(Figura 6.1.4) Ventana de código

Además el IDE de Borland tiene otras ventanas para la ubicación rápida de clases, componentes, eventos, procedimientos, funciones. De esta manera uno se ubica de manera fácil en determinado procedimiento de código.



(Figura 6.1.5) Ventanas de exploración de procedimientos y objetos

Para finalizar se resalta una de las últimas novedades del programa, el listado de tareas, que nos sirve para ubicar rápidamente la línea de código.



(Figura 6.1.6) Ventanas de tareas

Esta ventana me permitió dividir las secciones de código más importantes, de esta manera tener una estructura general de lo que tenía que hacer. La gran ventaja fue cuando mi código estaba entre las líneas 4000 a 5000, en este caso uno se perdía con la sabana de código, pero felizmente la ventana de tareas me facilito bastante en la localización de las partes.

Se debe tener en cuenta que un programador trate en lo posible que el código sea legible, y que cuando uno vuelve de un cierto tiempo no se de con la sorpresa de estar perdido en la sábana de código. Por ello es necesario que el código esté debidamente comentado.

6.1.2 Entornos de desarrollo integrado

Dentro del mundo de programación encontramos lenguajes de bajo nivel como ensamblador, de alto nivel como C++, Pascal e interpretadores.

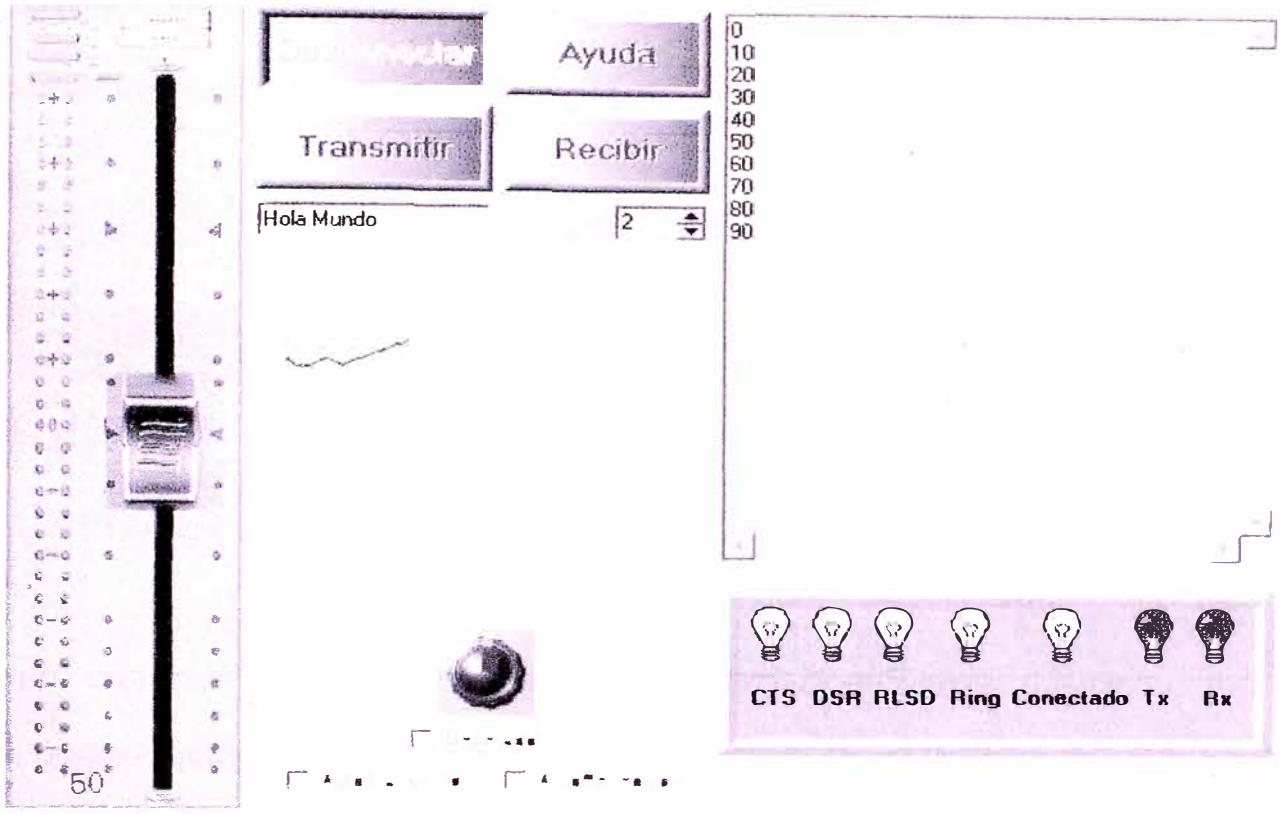
Borland comparte una filosofía con el interpretador Visual Basic (de ahora en adelante VB), que es programar en un entorno totalmente visual. Pero es un único punto que tiene en común con VB. Las principales diferencias están en el lenguaje en el que se apoya cada uno. Borland lo hace en C++ y *Object Pascal* que son lenguajes orientados a objetos, y VB como su nombre indica se basa en Basic. El Basic por mucho que lo disfrace con palabras como visual, orientado a objetos, nunca dejará de ser basic, en cambio C++ ya es un lenguaje compilado extremadamente rápido.

6.1.3 Envió y Presentación de señales

Para la recepción y transmisión de señales puede usar directamente VisualNiel2 con el menú serie o paralelo que se explicará luego. También puede usar el menú programas para usar el programa extra llamado Serie.

(Figura 6.1.7) Menú programas/serie

De este modo podemos usar los botones de transmitir y recibir del programa.



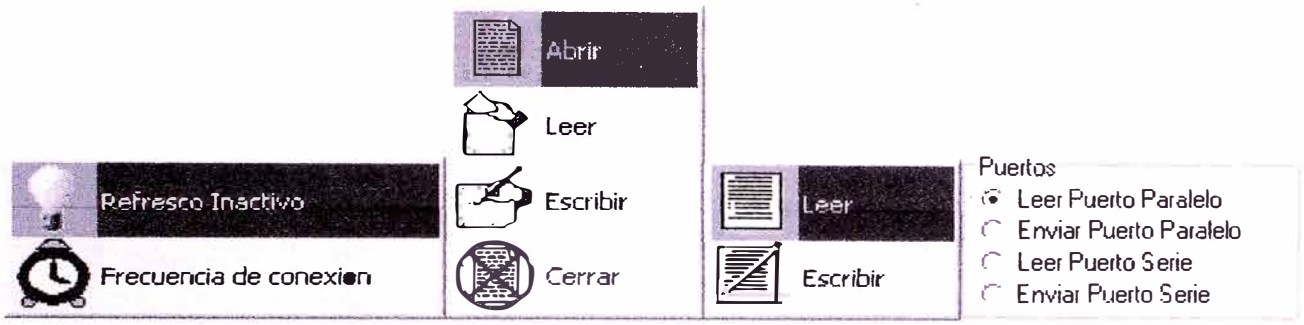
(Figura 6.1.8) Ventana Serie

Para el uso de esta ventana podemos activar la opción de conectar, para luego poder recibir datos del ADN2006 o cualquier dispositivo de transmisión serial que este conectado al puerto COM o serie con conector DB9 o ingrese la data por el pin 2 de dicho puerto.

Cuando el botón de conectar esté activo, éste ventana puede también transmitir datos seriales a otra PC o dispositivo que posea recepción serial, en todo caso la data que circula será en formato ascqui. El pin de transmisión serial es el 3 en el DB9 o puerto COM.

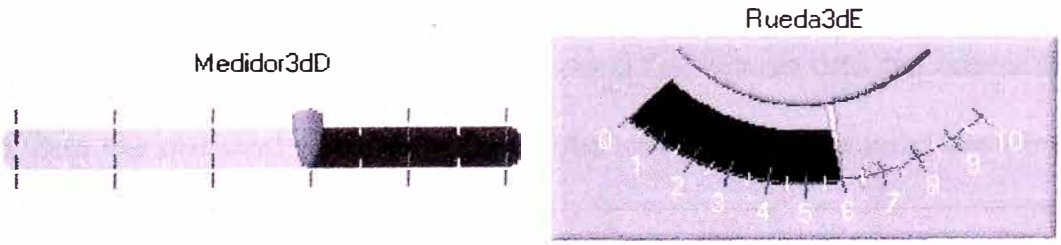
Para la presentación de señales se pueden usar los controles o componentes que vienen con VisualNiel2, para ello se procede como la

ventana anterior, es decir conectar para luego recibir datos ya sea por el puerto paralelo o serie.



(Figura 6.1.9) Menús conexión, serie, paralelo y opciones de puertos

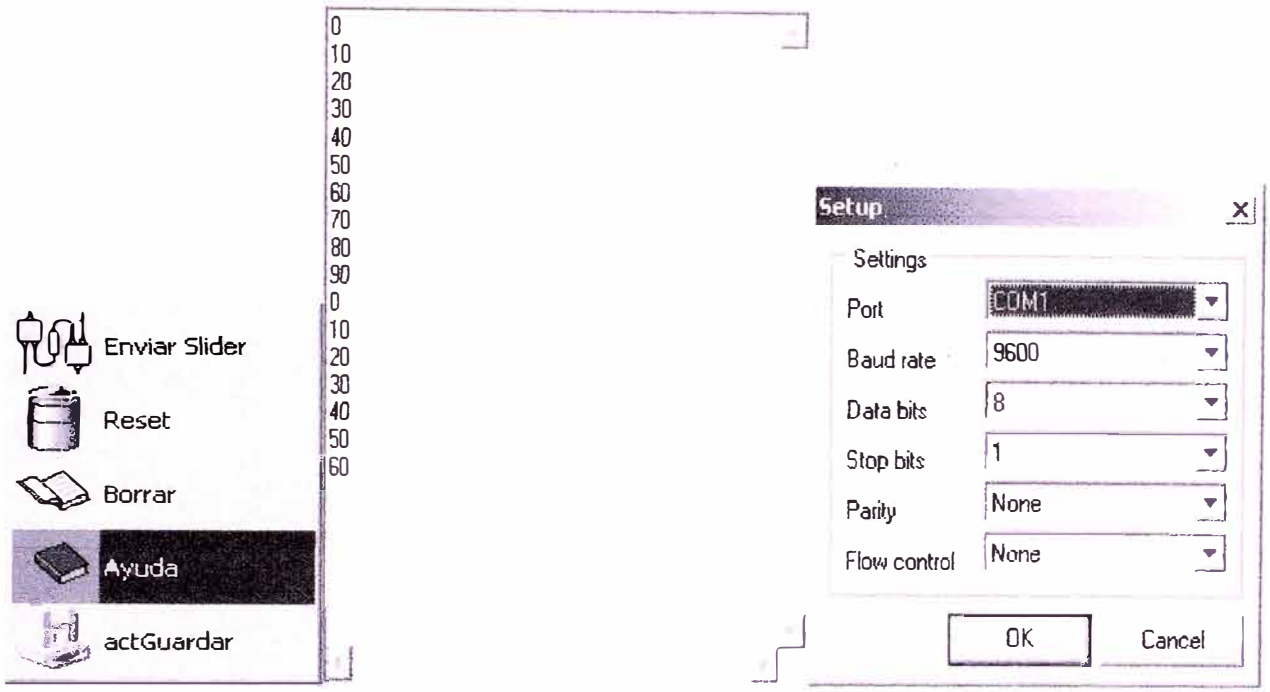
Para conectar al puerto serial se tiene que activar el foco de refresco y abrir el puerto serie, también se puede configurar la frecuencia de conexión, luego se procede a leer la data para visualizarlo en un potenciómetro por ejemplo.



(Figura 6.1.10) Medidor envía data y Rueda recibe

6.1.4 Almacenamiento de datos

Para guardar los datos recibidos a través del puerto serial use la ventana serie, y en el cuadro de texto dar clic derecho para abrir el menú contextual de guardar data. También puede usar Borrar para limpiar los datos entrados.



(Figura 6.1.11) Menú guardar data y ayuda en la ventana serie.

Puede hacer clic en Ayuda para configurar la velocidad en baudios del puerto seleccionado, también están los parámetros de bits de datos de entrada, bits de paridad. La velocidad estándar en la conexión serial es de 4800, 9600 ó 19200 baudios.

6.1.5 Abrir y guardar configuraciones

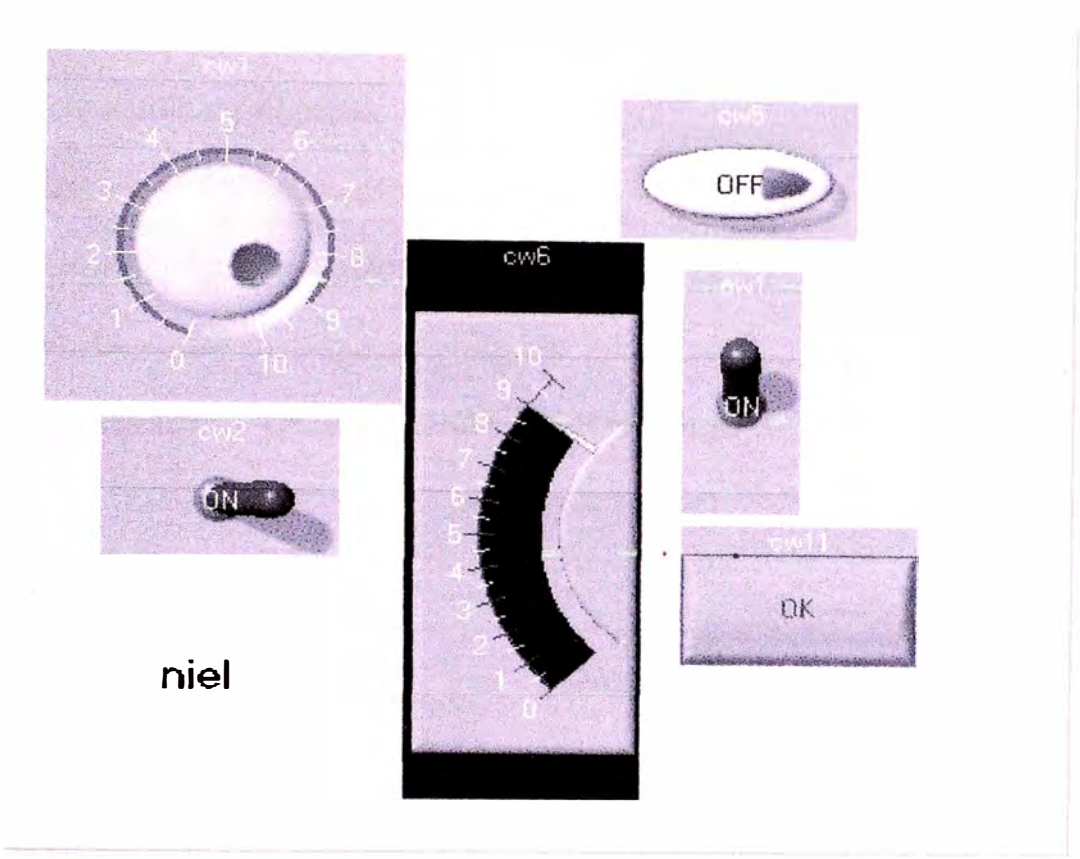
La programación tradicional para el entorno Windows ha tenido siempre como base al lenguaje C, desde ahí viene la tradición de re-usar código. Un programa no tendría sentido si no guardara sus configuraciones para luego volverlas abrir, y encontrar la configuración donde la dejamos.

(Figura 6.1.12) Botones de guardar y abrir archivo.

Por ello es indispensable que las configuraciones se guarden en archivos que para nuestro caso tendrá la extensión .nie, estos archivos guardan las propiedades y características de los componentes.

(Figura 6.1.13) Archivo guardado en formato nie

Es decir cada componente que insertamos en visualNiel2, ya sea un potenciómetro, un botón, un deslizador; tienen un conjunto de propiedades dentro del programa, y para recuperarlos tenemos que guardarlo en un archivo. Este archivo automáticamente ejecuta un conjunto de comandos cuando es abierto para llegar hasta la forma en que la dejamos.



(Figura 6.1.14) Archivo de Configuración abierta.

6.2 Procesamiento de datos adquiridos en PC

6.2.1 Estructura de la codificación

El código de VisualNiel2 fue seccionada para mantener cierto orden al momento de la actualización. A continuación se menciona las partes.

-Programa: Abrir configuraciones del programa

-Programa: Acciones avanzadas de adquisición de datos

-Objeto y Programa: Actualización de datos de la tabla

-Objeto y Programa: Ayudas del objeto y del programa

-Objeto: Cambia el tamaño y posición del objeto

-Objeto: Configuración de colores de los objetos

-Objeto: Creación de objetos

-Objeto: Destrucción de objetos

-Objeto: Evento clic en el objeto

-Programa: Guardar configuraciones del programa

-Programa: Mostrar y Ocultar partes del Programa

6.2.2 Descripción de módulos

- **Creación de objetos**

Creamos todos los objetos haciendo clic en la paleta de componentes. Acá se mencionan los nombres por defecto de alguno de ellos

lab:Tlabel; edi:Tedit; mem:TMemo; bot:Tbutton; che:TcheckBox;
rad:TradioButton; lis:TlistBox; com:TcomboBox; bar:TscrollBar;
gru:TgroupBox; gra:TradioGroup; pan:Tpanel;

botA,botB,botC,botD,botE,botF,botG,botH,botI,botJ,
botK,botL,botM,botN:TCWbutton;

rueA,rueB,rueC,rueD,rueE,rueF,rueG,rueH,rueI,rueJ:TCWslide;

- **Destrucción de objetos**

Esto es para liberar la memoria que se asigno a los objetos y ahora se detalla el algoritmo.

procedure TForm2.FormDestroy(Sender: TObject);// función de destrucción

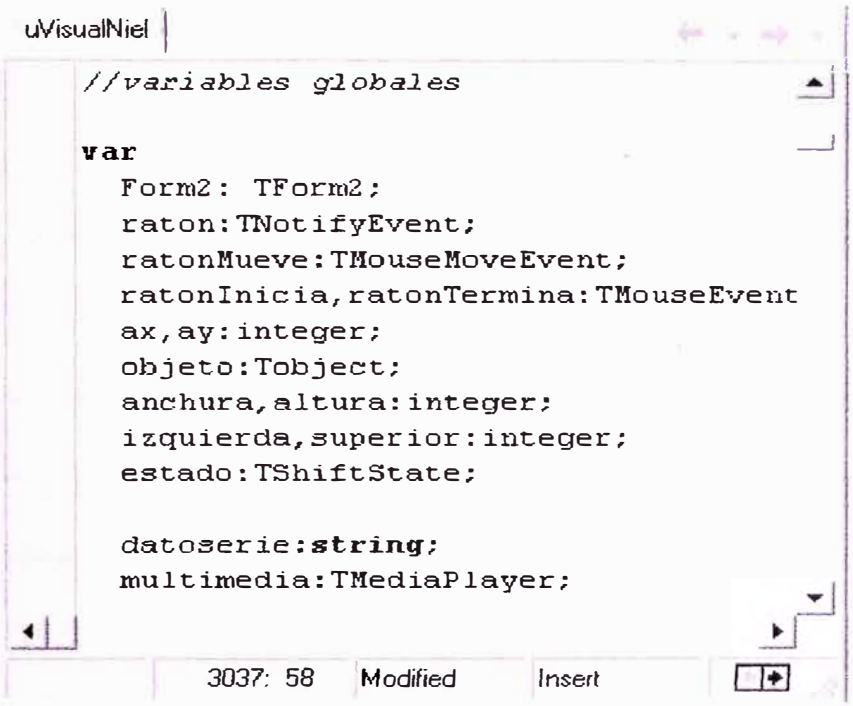
```
var n:integer; //variable para contar objetos
{
for n:=paObjetos.ComponentCount-1 downto 0 do //cuenta los componentes
paObjetos.Components[n].Free;//libera los componentes
};
```

- **Otras funciones de los módulos**

-Pulsando la tecla enter en la tabla actualizamos los datos del objeto.
-Con el objeto seleccionado cambiamos el tamaño con el evento OnMouseMove del propio objeto.
-Haciendo clic en el objeto obtenemos el elemento seleccionado y actualizamos los datos de la tabla.
- Mostrar y Ocultar partes del Programa haciendo clic derecho en el programa.

6.2.3 Variables generales y específicas

Las variables generales o globales son usadas en todo el programa, tiene la ventaja de que se le puede llamar desde cualquier parte del programa. Pero el programa se vuelve un poco lento debido a que estas variables tienen que estar almacenado en la memoria RAM todo el tiempo, incluso si es que no se le necesita.



(Figura 6.2.1) Variables globales.

A diferencia de las variables globales, las variables específicas o de módulo sólo se usan temporalmente en el interior de una función luego son destruidas. Por lo tanto significan un considerable ahorro de memoria, pero no se puede usar en todos los módulos del programa; se necesitaría volverlo a crear y esto es demanda de tiempo para el programador.

Por lo tanto el desarrollador tiene que tener en cuenta las 2 posiciones, y es recomendable usar variables globales sólo cuando sea estrictamente necesario a menos que no existan problemas de velocidad de ejecución.

Normalmente los contadores de bucles internos son variables específicas.

6.2.4 Algoritmos de Creación de objetos en RunTime

VisualNiel2 en su intención de ser un programa genérico que sea capaz de crear aplicaciones donde las propiedades y objetos no se conocen de antemano, es imprescindible la creación de objetos en RunTime, es decir en tiempo de ejecución; donde el usuario es el agente que creará o no algún objeto que desee. Por ello casi todos los lenguajes de programación usan esta técnica para desarrollar componentes, ya sean Activex de Microsoft o cualquier componente DLL.

Para crear un objeto en tiempo de ejecución se necesita un panel padre que albergue a dicho componente, para ello procedemos de tener listo el ancestro y si alguien agrega un componente, inmediatamente se asignan propiedades por defecto.

El algoritmo de jerarquía mínima de creación de objeto es la siguiente.

```
if tbBoton3dA.Down then// si alguien presiona crear botón
 {
 bot3A:=TCWbutton.Create(paObjetos);// al objeto se le asigna un padre
 with bot3A do // con el objeto creado hacer lo siguiente:
  {
   Parent:=paObjetos; // el padre es un panel
   name:='bot3A'+IntToStr(Random(1000));// se le da un nombre
```

```
ImportarEstilo('boton3d01.niel');// se le asigna un cuerpo
caption:=toolbar5.Buttons[1].Caption;// se le da un apellido
onMouseDown:=ratonInicia;// se le asigna el evento down
onMouseMove:=ratonMueve;// se le asigan el evento arrstrar
onClick:=raton;// se le asigna el evento clic
Left:=x; // se le da una posición horizontal
Top:=y; // se le da una posición vertical
};
}
```

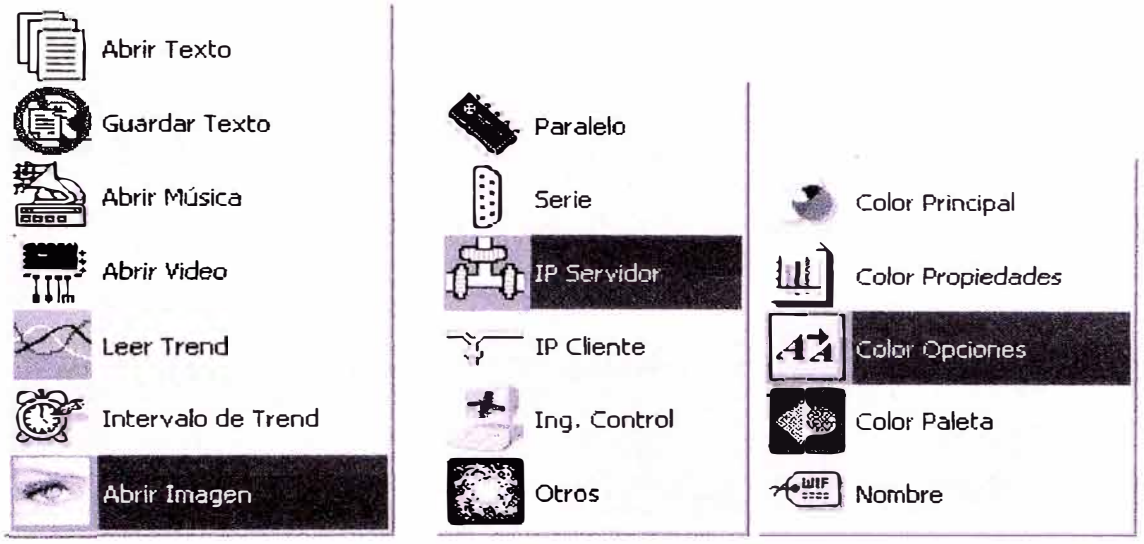
6.3 Funcionamiento general de VISUALNIEL2

6.3.1 Partes del programa

- **Menú :** Se tienen casi todas las características y opciones del programa y son los siguientes:

Archivo, Edición, Refresco, Color IDE, Proyectos, Serie, Paralelo, Ayuda, Programas, Conexión. Estos a su vez despliegan ítems y son:

(Figura 6.3.1) Menús refresco, conexión, paralelo, ayuda, archivo, serie

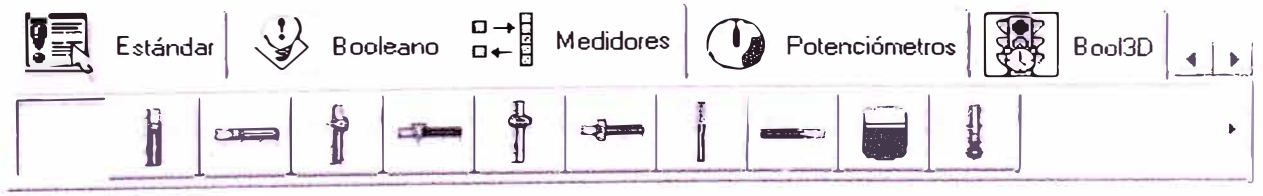


(Figura 6.3.2) Menús proyectos, programas, color IDE

- **Controles :** La paleta de componentes alberga a un gran número de objetos para desarrollar aplicaciones y tiene estas distribuciones: Estándar, Booleano, Medidores, Potenciómetros, Bool3d, Medi3d, Poten3d, Proyectos. Estos a su vez contienen íconos y son:

(Figura 6.3.3)Paleta Estándar

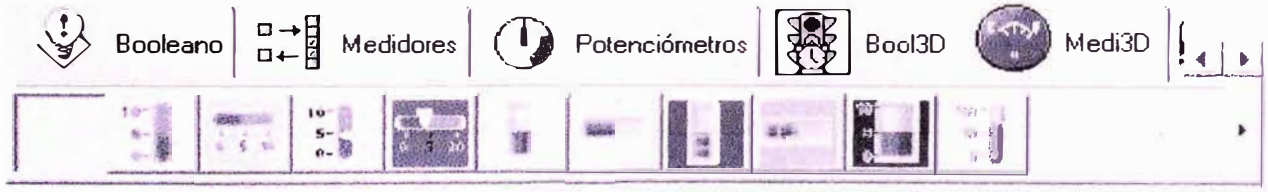
(Figura 6.3.4)Paleta Booleano



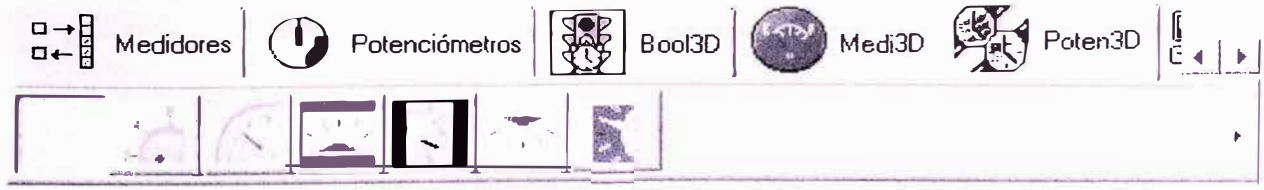
(Figura 6.3.5)Paleta Medidores

(Figura 6.3.6) Paleta Potenciómetros

(Figura 6.3.7) Paleta Bool3d



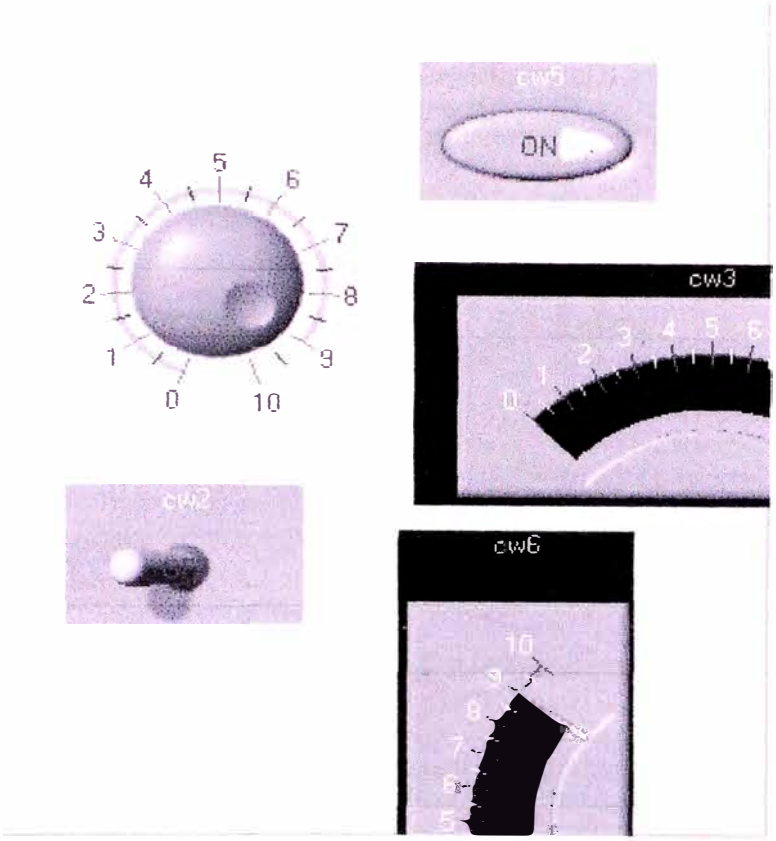
(Figura 6.3.8) Paleta Medi3d



(Figura 6.3.9) Paleta Poten3d

(Figura 6.3.10) Paleta Proyectos

- **Panel principal:** Es el espacio mayor del programa donde se colocan los componentes, para posicionarlos y dimensionarlos.

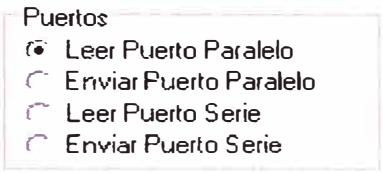


(Figura 6.3.11) Panel principal

- **Propiedades:** Es el inspector de propiedades de los componentes, desde acá se pueden cambiar sus características tales como color, posición, valor, etc.

Propiedades	Valores
Posicion X	219
Posicion Y	315
Anchura	100
Altura	266
Color Rojo	0
Color Verde	0
Color Azul	0
Nombre	cw6
Valor	10
Tag	0
Maximo	10
Minimo	0

(Figura 6.3.12) Inspector de propiedades

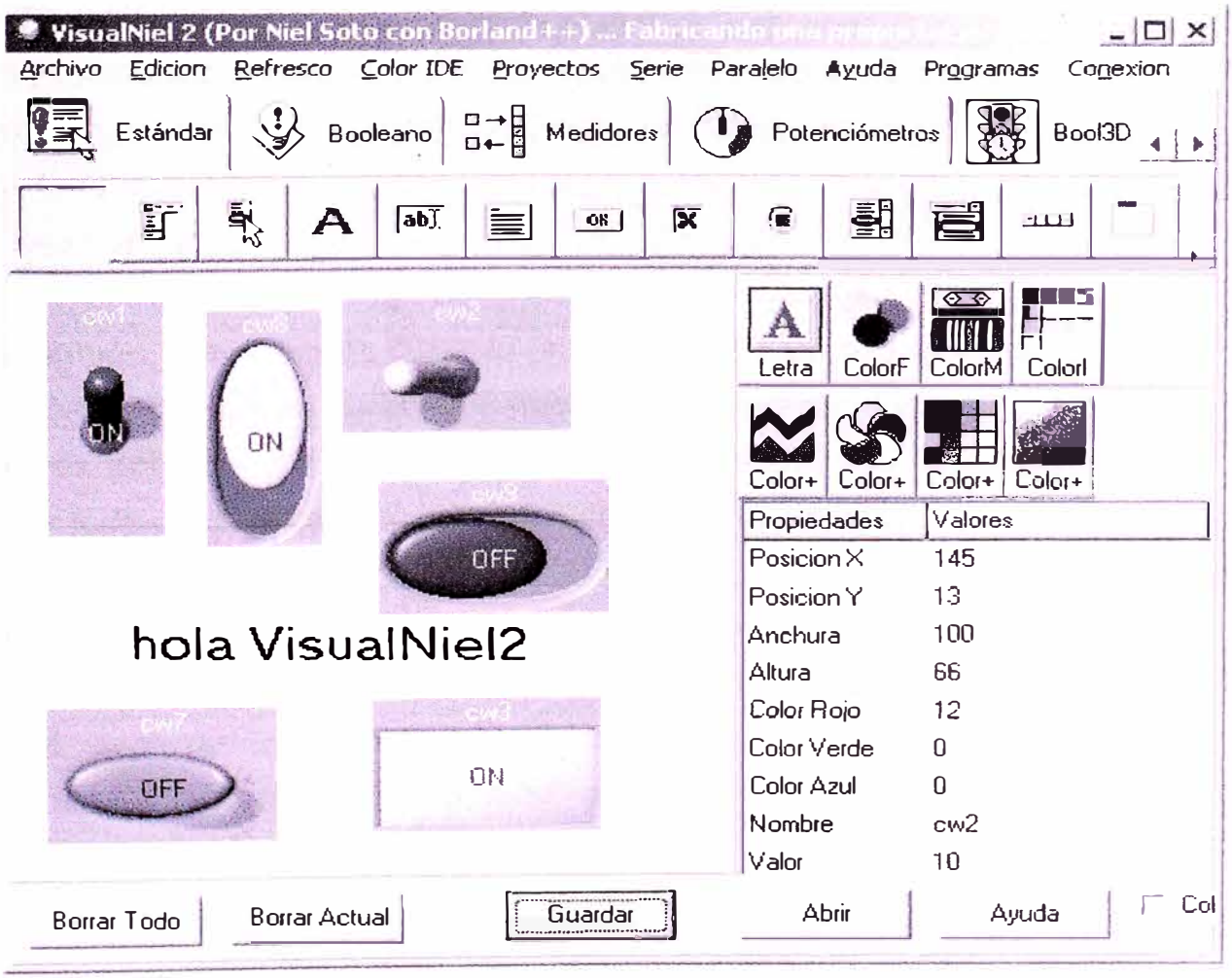


(Figura 6.3.13) Opciones en el Inspector de propiedades

- **Opciones:** Es el panel de opciones adicionales, tales como eliminar objeto, abrir y guardar archivos.

(Figura 6.3.14) panel de opciones

A continuación mostramos el programa VisualNiel2 de manera íntegra



(Figura 6.3.15) VisualNiel2 completo

6.3.2 Funcionamiento del programa

Una vez conocida las partes del programa, está listo para empezar a desarrollar una aplicación.

-De la paleta de componentes seleccione cualquier icono y hacer clic en ello.
-Luego hacer clic en alguna posición del panel principal para situar el objeto.
-Puede cambiar la sección de la paleta para sacar los objetos que desea.

Ahora que tiene la aplicación, es muy probable que se desee mover o dimensionar el objeto. Entonces haga lo siguiente:

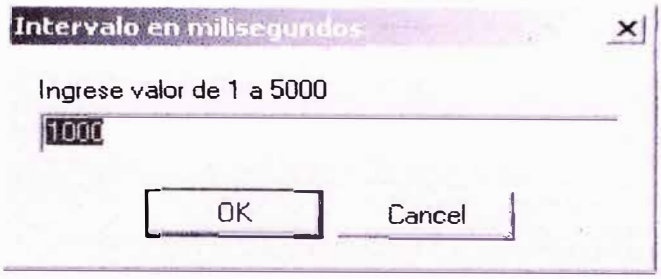
-Puede usar las teclas claves Shift y Ctrl dependiendo de lo siguiente.
-Shift y moviendo el mouse cambias la posición del Objeto
-Ctrl y moviendo el mouse cambias las dimensiones del Objeto
-También puedes usar el inspector de propiedades para modificar características del objeto seleccionado.
-La selección de un objeto es haciendo clic en el objeto.

En resumen, básicamente creas tu propio software en segundos, sólo es cuestión de que lleves los objetos hacia el panel principal y luego cuando terminas, sólo das clic derecho en el panel principal y activas la opción ocultar todo para de esta manera fácil obtengas tu software.

Tener en cuenta que este Software es sumamente rápido ya que fue hecho en Borland++, combinando la eficiencia de c++ y algunas ventajas del Pascal, para de esa manera tener un software eficaz y también código optimizado para tener un tamaño pequeño (2.2xMb) que cuenta con alrededor de 5000 líneas de código.

Para las funciones avanzadas tales como la adquisición de datos, se debe activar los timers con los menús refresco y conexión.

Se tiene que configurar la velocidad del timer, para determinar la frecuencia de visualización



(Figura 6.3.16) Intervalo del timer

Se pueden usar programas extras del menú programas para comunicación serial, paralela e Internet.

Si se desea una configuración de colores diferente del programa en general debe ir al menú Color IDE (Ambiente de desarrollo integrado).

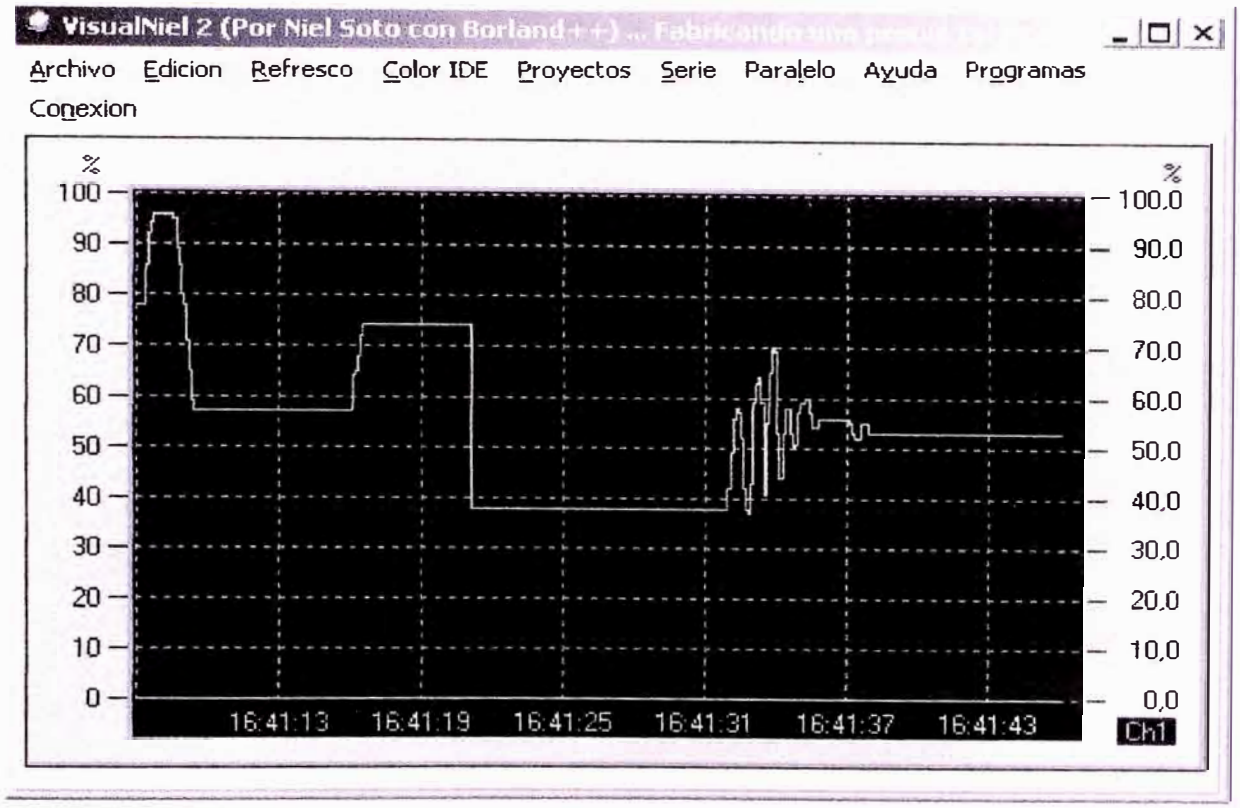
El menú de proyectos brinda posibilidades visualizar imágenes, reproducir música y video, además de tener una graficadora en tiempo real.

6.3.3 Pruebas y resultados obtenidos

a) Entre las pruebas realizadas está la de la ejecución de 2 versiones de VisualNiel2 ejecutándose simultáneamente donde uno de ellos es el transmisor y el toro es el receptor. Tener en cuenta que para cualquier tipo de transmisión ya sea paralela, serial o Internet se necesita la activación del timer y que es necesario sincronizar dichos relojes; por ejemplo a un intervalo de100 milisegundos.

El programa transmisor enviará la data moviendo el potenciómetro mientras que el programa receptor mostrará la data de entrada en una graficadora que usualmente se le llama Trend.

Cuanto menor sea el intervalo del reloj más rápido será la actualización.

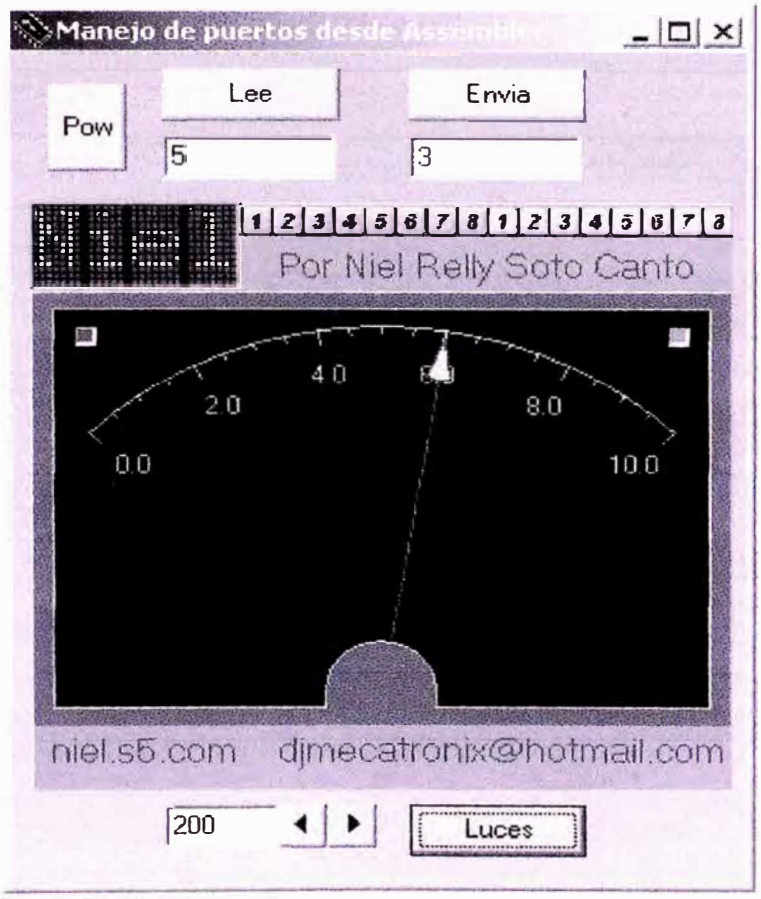


(Figura 6.3.17) Programa receptor

(Figura 6.3.18) Programa transmisor

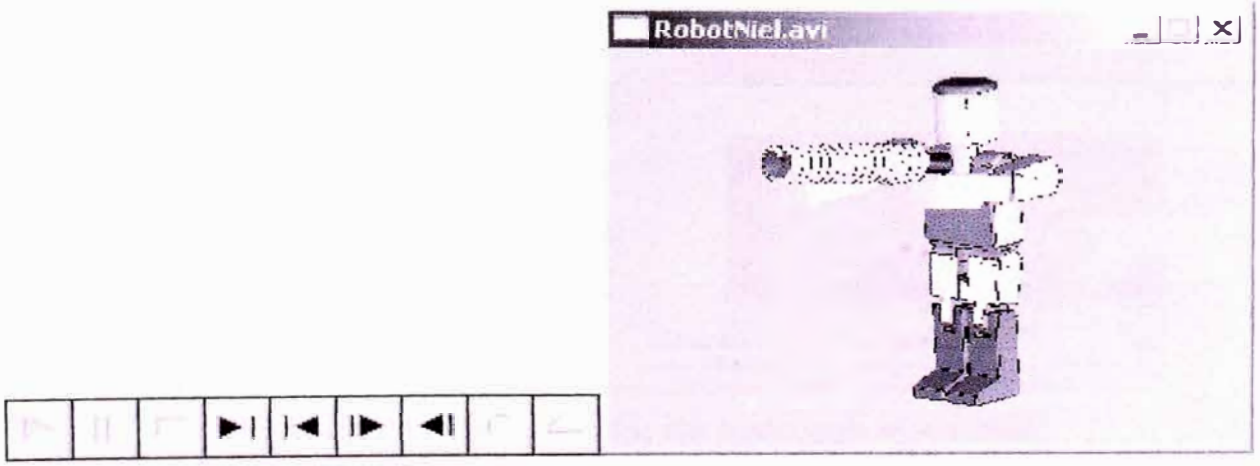
A los 2 programas anteriores se les ha desactivado la visualización de algunas de sus partes, por ello no aparecen sus barras de propiedades.

b) Otra de las pruebas realizadas es la de usar el menú programas/paralelo para enviar datos paralelo internamente, y recibir con el programa VisualNiel2 y de esa manera tener una simulación de movimiento con el botón luces.



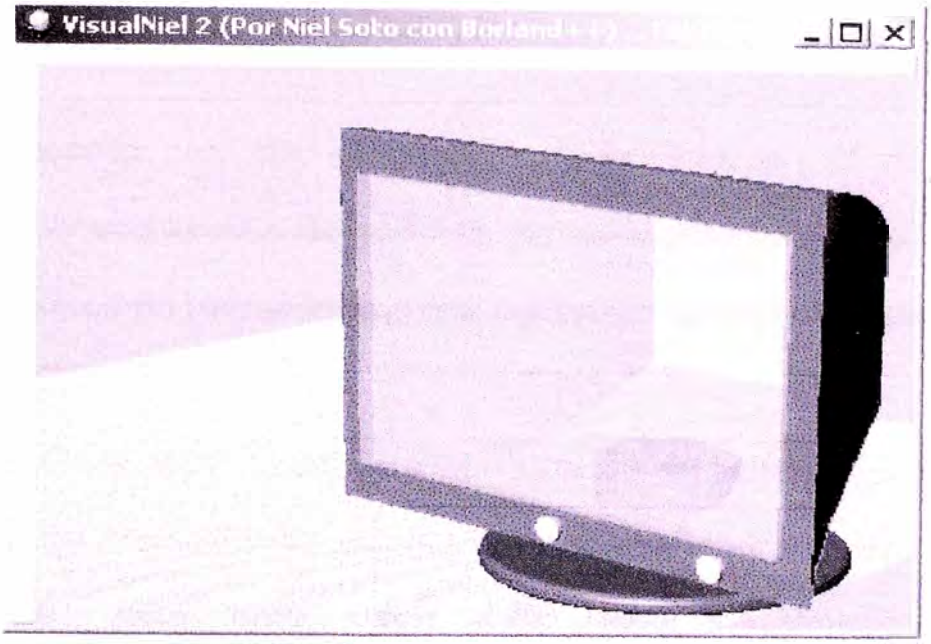
(Figura 6.3.19) Programa transmisor paralelo

c) La reproducción de audio y video son accesorios con proyectos/video.

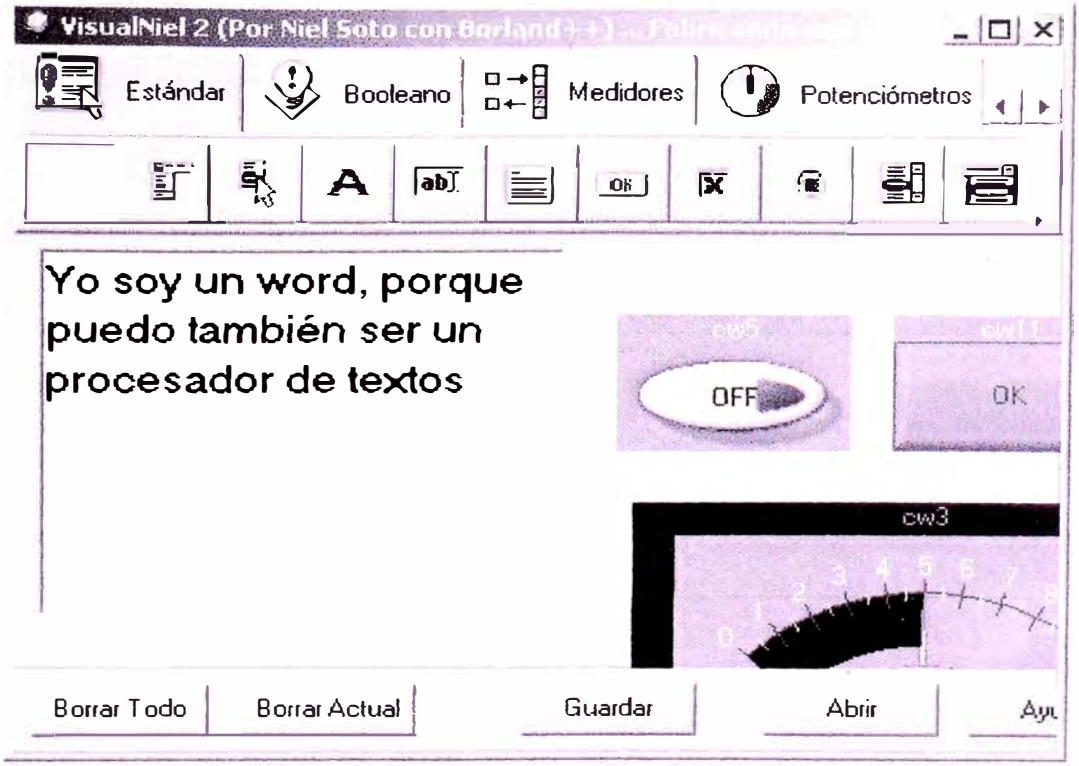


(Figura 6.3.20) Reproductor de video con visualNiel2

d) Entre las pruebas también está la de configuración del IDE, es decir algunos como graficadores sin nada de opciones, o en otro caso emular un visor gráfico o talvez un procesador de texto.



(Figura 6.3.21) visualizador de gráfico con visualNiel2



(Figura 6.3.22) Procesador de texto con visualNiel2

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES:

- De acuerdo con las experiencias adquiridas en el desarrollo de dispositivos de adquisición de señales, se ha logrado usar los sitemas de tecnología wire como herramientas que agilizarán aplicaciones de ingeniería.

- La DAQ de MCC y Softwire son una ayuda valiosa para el uso en aplicaciones de medición de señales tales como temperatura , niveles de llenado de tanques ,obtención de data de encoders y almacenamiento de cualquier tipo de data analógica y digital que pueden luego ser procesados y retransmitidos vía interfaces serial, paralela e Internet, que pueden ser complementados con otros software ya que la interfaz es abierta.

- Es muy importante saber que Softwire y VisuaNiel2 no solo están diseñados para algún tipo de aplicación, con un poco de imaginación se puede mezclar y obtener resultados inesperados para el autor, pero cabe destacar que puede funcionar como Word , Reproductor de Audio y Video , Visor de imágenes, Creador de diagrama de bloques, testeo de puertos paralelo serial e Internet.

- Opcionalmente se agregó herramientas extras a VisualNiel2 tales como la obtención de la ecuación en diferencias de Plantas de primer y

Segundo orden, luego de haber realizado el debido algoritmo para los cálculos de llevar Transformaciones del plano S a Z en la discretización de señal (método de TUSTIN).

- El proceso de transmitir cualquier tipo de señal tiene el inconveniente de perder data cuando la distancia se incrementa, para ello las DAQ tienen parámetros de corrección de errores mediante lectura de retransmisión y la velocidad de transmisión recomendada es de 9600 bps y 19200 bps para aplicaciones exigentes.

- En este proyecto solo se hizo la adquisición y procesamiento de señales de sensores, pero puede usarse para otras aplicaciones como para transmisión y recepción serial, reproductores multimedia, comunicación por Internet a partir de lo mencionado.

- El filtro Notch (de 60 Hz) simulado en Proteus de la PC elimina de manera adecuada el ruido de línea sin deformar la señal. Esto prueba la utilidad de algoritmos de procesamiento de señales implantadas en las DAQ en el acondicionamiento y tratamiento de diversos tipos de ondas.

- Para el uso de equipos DAQs deben tomarse en cuenta todo tipo de consideraciones técnicas, sin embargo los equipos que deberían ser usados en universidades por el bajo coste y alto rendimiento para el procesamiento

de datos son las DAQ de MCC y deben tomarse precauciones de acuerdo al tipo de aplicación que tendrá el equipo.

- El programa aplicativo esta estructurado con bloques que se conectan con Wires de tal manera que facilita la correcta transmisión y recepción de Datos desde un ETD (Equipo Terminal de Datos) a un ECD (Equipo de Comunicación de Datos). Los Datos llegan a los puertos de forma asíncrona, de tal manera tiene que procesarse inmediatamente para poder adquirir nuevos datos sin que se ilene el búfer.

- Los módulos de tecnología wire permiten el manejo del puerto serial, así como de detalles gráficos que permiten la visualización de las señales de sensores ya sea analógico o digital. Estos módulos de periféricos se conectan en la forma mas sencilla posible que permiten tenerlos de ejemplo y guía para futuras aplicaciones con otro tipo de sensores.

- Softwire cumple con diversas tareas en la adquisición de muestras de señales, transferencia de información de computadora a computadora. Se requiere únicamente que el puerto de comunicaciones este bien configurado. Para conseguir una conexión entre ambos módems el programa enviará un comando de atención al módem que esta conectado al equipo.

- El programa Softwire es adaptable a cualquier computadora que cuente con un puerto serial y que tenga instalado el sistema operativo Windows 98 o superior.

- Se puede desarrollar cambios en los bloques para aplicaciones distintas dependiendo del interesado. Partiendo de los principios y rutinas que se ha considerado.

- Los bloques de filtros y acondicionadores de señal se describen de una manera clara y sencilla, por ello es muy conveniente tener de guía estos pasos en bloques

- Es importante recoger experiencias sobre adquisición de datos anteriores de proyectos similares al que uno esta emprendiendo para tener una guía y un camino mas claro.

- El uso de los microcontroladores y DSP es de uso común en los dispositivos DAQ, pero debe indicarse que los microcontroladores PIC vienen teniendo mayor aceptación debido a la variedad de herramientas y a la simplificación de algunas funciones (funciones incorporadas como: comunicación serial USART, I2C, PSD, etc.), además de su bajo costo.

- Los programas en Softwire están hechas en forma modular (un módulo de programa maneja un periférico) cuyos bloques son independientes y se puede utilizar para muchas otras aplicaciones.

- El uso de Internet facilitó en gran medida el estudio de la tecnología Wire y de cómo poder hacer el procesamiento de las señales. También fue muy importante el contacto directo con los especialistas quienes colaboraron mucho en este punto.

RECOMENDACIONES:

- Para establecer conexión con la Computadora y el dispositivo de adquisición de datos (DAQ), el puerto de comunicaciones serie debe estar habilitado y configurado. Se debe tener cuidado de revisar los puertos hábiles para evitar conflictos.

- Las características comunes que tienen todos los equipos a ser usados en adquisición de señales de sensores con tecnología Wire son los siguientes:

-Dar un resultado positivo para los fines que fue creado, proveyendo alguna innovación y ahorro de dinero.
-Deben ser seguros y fáciles de usar (con relación a Softwire y uso de sus componentes).

-Deben ser amigables al usuario, de manera que pueda ser usado con mínima capacitación.

- Debe haber un esquema en bloques para el uso del equipó al momento de la adquisición de datos.

- Futuros estudios deberían centrase en realizar diferentes aplicaciones de procesamiento de señales de sensores: de iluminación, cardíacas, respiración y latidos, compresión de voz, códigos de transmisión para detección y corrección de errores, protocolo de transmisión.

- Se debe usar diagramas que se conecten en forma de Wires en la presentación (visualización) del proyecto a trabajar, esto ayuda bastante en la aceptación de este.

- Se debe incentivar a la población al uso de tecnologías Wire para que las personas pueden realizar la adquisición de cualquier tipo de séanles y en el procesamiento tratar de automatizarlas para el beneficio de la humanidad. Ejemplos de ello son la iluminación autorregulada por ejemplo si el día va oscureciendo la luz del hogar se enciende gradualmente, siempre en cuando se detecte presencia humana despierta. Las computadoras se apagan, si el límite superior del sensor es sobrepasado.

- Usar Internet como una fuente de información constante e inagotable.

BIBLIOGRAFIA

(1) BILSTEIN, Paul; Filtros activos; Marcombo; 1977

(2) CEBALLOS, Javier; Enciclopedia Microsoft Visual Basic; Ra-Ma; 1994

(3) GUREWICH, Nathan; Visual C++ in 21 days; Sams Premier; 1997

(4) HALVORSON, Michael; Aprenda proteus ya; Mc GrawHill; 1997

(5) JOHNSON, Johnson; Handbook of DAQ filters; Prentice Hall

(6) WATERS, Allan; Active filter design; Mc GrawHill; 1991

DIRECCIONES ELECTRÓNICAS:

1)http://sunsite.tut.fi/hwb/ (Configuración de Filtros y comunicación)

2)http://www.microchip.com (MICROCHIP)

3)http://www.analog.com (ANALOG DEVICES AND DAQS)

4)http://www.softwire.com (TECNOLOGÍA WIRE)

5)http://www.ni.com (NATIONAL INSTRUMENTS)

6) http://www.measurementcomputing.com/dasylab.html (DAQS)

7)http://www.ti.com (TEXAS INSTRUMENTS)

8)http://www.avnet.com (Distribuidor de componentes electrónicos)

9)www.mathworks.com

Buscadores utilizados:

http://www.google.com

http://www.altavista.com

APENDICE

ww1.microchip.com/downloads/en/DeviceDoc/39582b.pdf