UNIVERSIDAD ROVIRA I VIRGILI

PROYECTO DE FINAL DE CARRERA

Osciloscopio basado en Windows 95 utilizando el puerto serie

Realizado por Raúl Bartolomé Castro
Dirigido por Ernest Gil Dolcet
1999 / Septiembre

ÍNDICE

1 MEMORIA DESCRIPTIVA

1.1 OBJETIVOS

El objetivo de este proyecto es desarrollar una placa de adquisición de datos analógicos, gobernada por una computadora personal o PC sobre el sistema operativo Windows 95 mediante el puerto serie.

Debe presentar en pantalla e impresora las lecturas temporales adquiridas de la placa, así como la posibilidad de guardar los datos en memoria no volátil.

También se debe implementar los controles funcionales típicos de un osciloscopio.

1.2 DESCRIPCIÓN GENERAL

Para la realización del osciloscopio se utilizará un ordenador del tipo PC, aprovechando la capacidad de éste para procesar datos y poder representarlos de forma gráfica. Mediante una tarjeta se recogerán la información, se convertirán en digital y se transmitirá al ordenador para que éste los procese con un programa desarrollado en Visual C++ sobre el sistema operativo Windows 95.

El proyecto consta de tres partes bien diferenciadas:

1) Periférico: constituido por una tarjeta de adquisición de datos analógicos de cuatro canales, la cual está conectada al PC mediante el puerto serie. El "corazón" de esta tarjeta es un microcontrolador, que se encarga de realizar el control de la electrónica y la transmisión de la información por el canal serie al PC
2) Software de bajo nivel: éste es el "cerebro" del periférico, debe interpretar las órdenes recibidas mediante el canal serie para alterar la electrónica de control y transmitir la información adquirida por los canales.
3) Software de alto nivel: constituye el "interface" de usuario. También se encarga de procesar la información procedente del canal serie para presentarla como el usuario desea, así como transmitir al periférico el estado que éste debe adaptar.

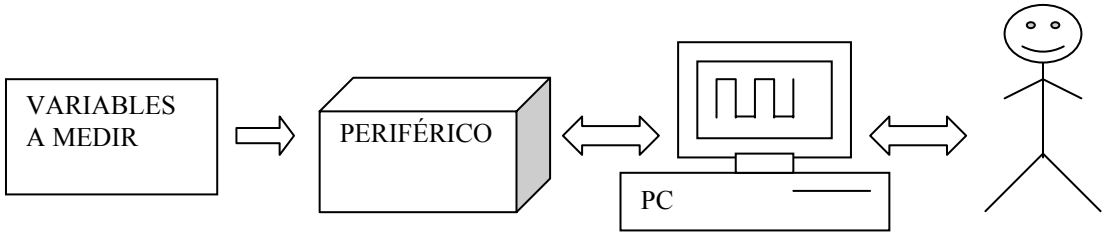

Ilustración 1.1: diagrama general

El periférico incluye diferentes bloques que le permiten realizar las funciones de un osciloscopio controlado mediante un PC:

- Cuatro etapas de entrada con la posibilidad de atenuación interna.
- Multiplexación de canales.
- Amplificación de ganancia variable.
- Filtrado banda pasante o filtro paso bajo.
- Sumador.
- Convertidor de analógico a digital.
- Microcontrolador.
- Memoria suplementaria.
- Interface serie.

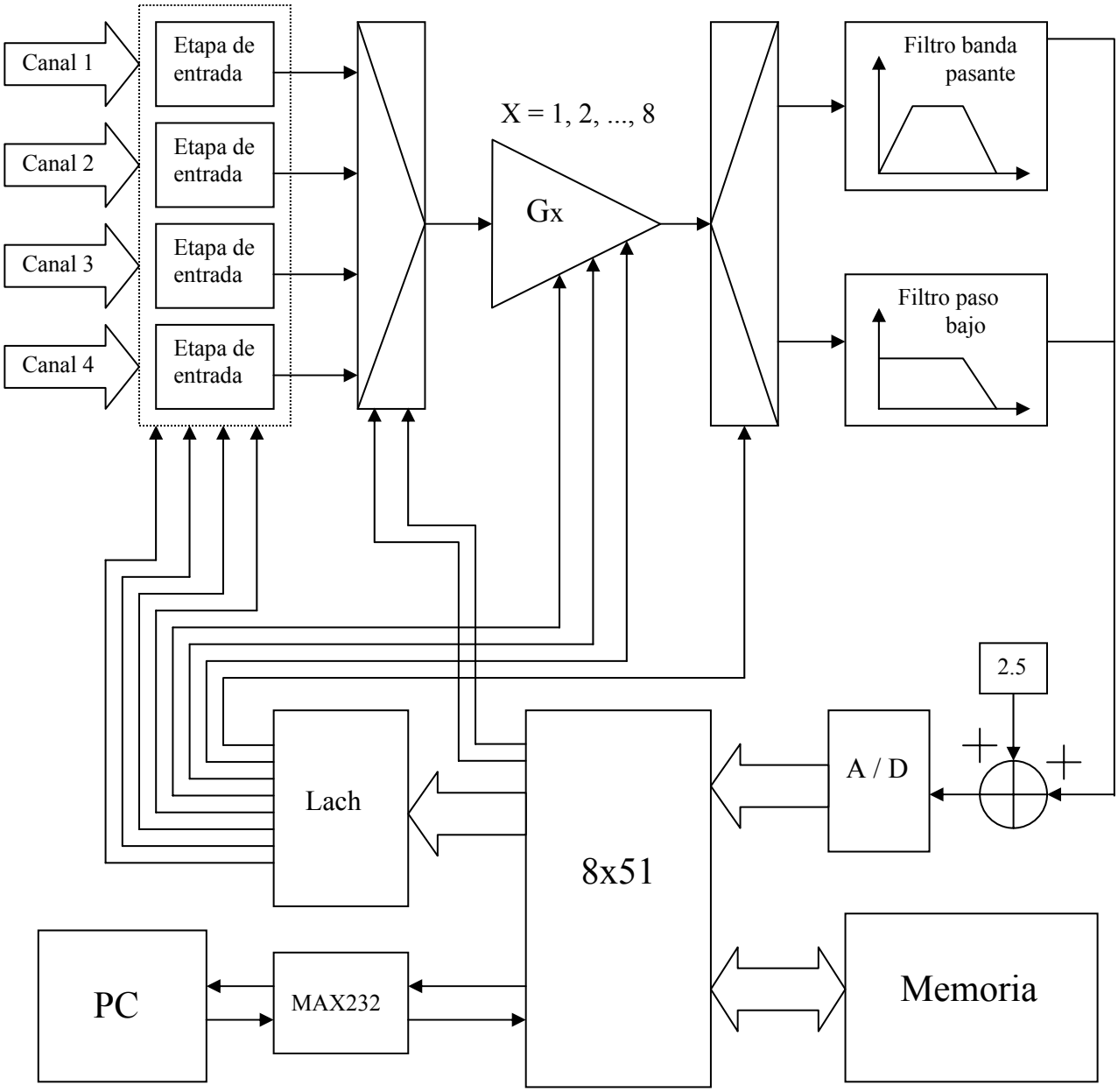

Ilustración 1.2: diagrama de bloques

El software de bajo nivel que está formado por código en ensablador interpretable por el microcontrolador 8x51 constituido por los siguientes conceptos:

- Inicialización del microcontrolador y electrónica externa.
- Muestreo de las señales analógicas a medir y guardar la información en memoria.
- Transmisión de las muestras adquiridas que están almacenadas en memoria.
- Gestión de las ordenes recibidas por el PC.

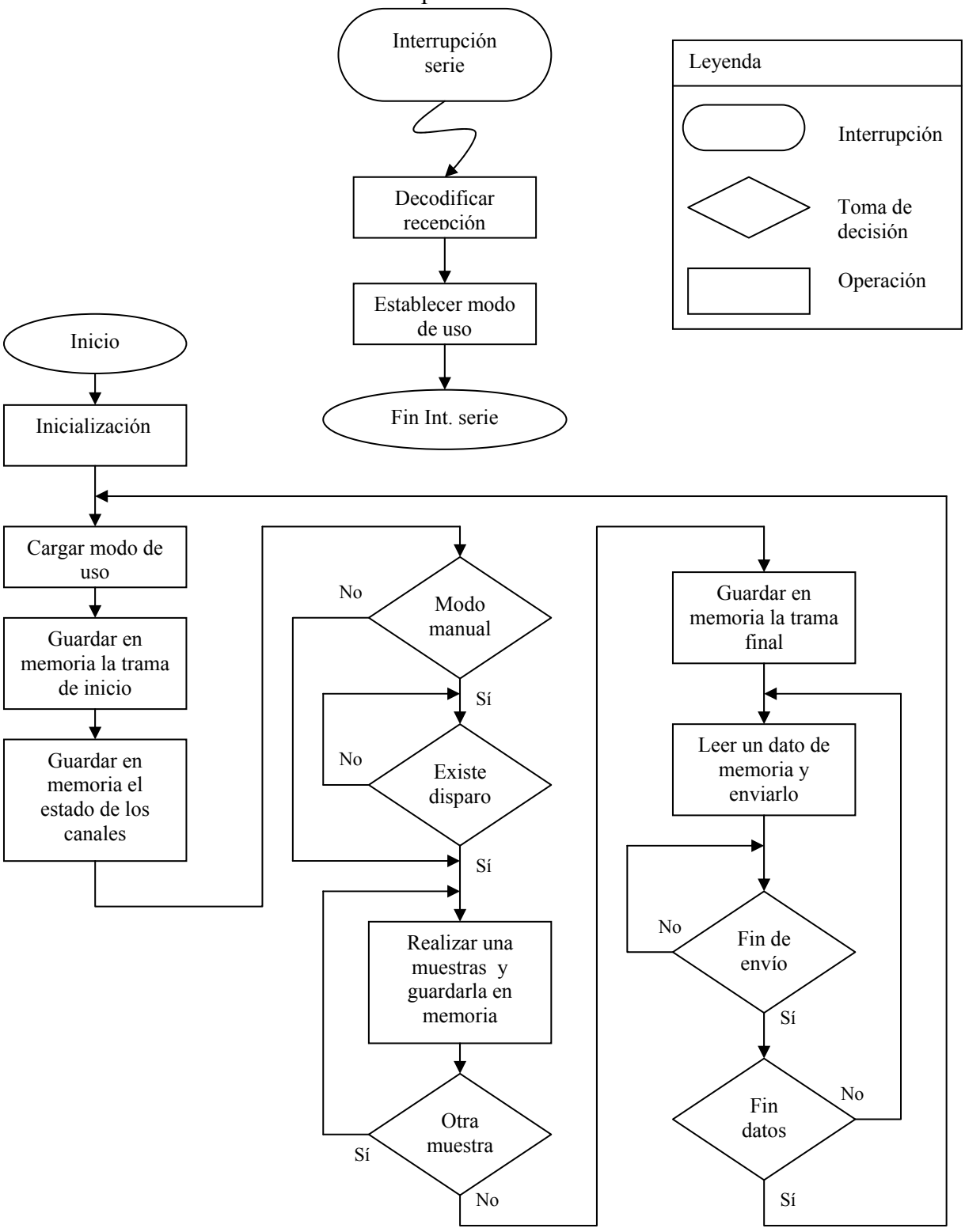

Ilustración 1.3: diagrama de flujo

El software de alto nivel se ha creado mediante el lenguaje Microsoft Visual C++ versión 5.0, se encarga de realizar las siguientes funciones:

- Presentación por pantalla e impresora de las variables a medir.
- Implementación de los controles típicos de un osciloscopio.
- Almacenamiento de la información recibida en memoria no volátil.

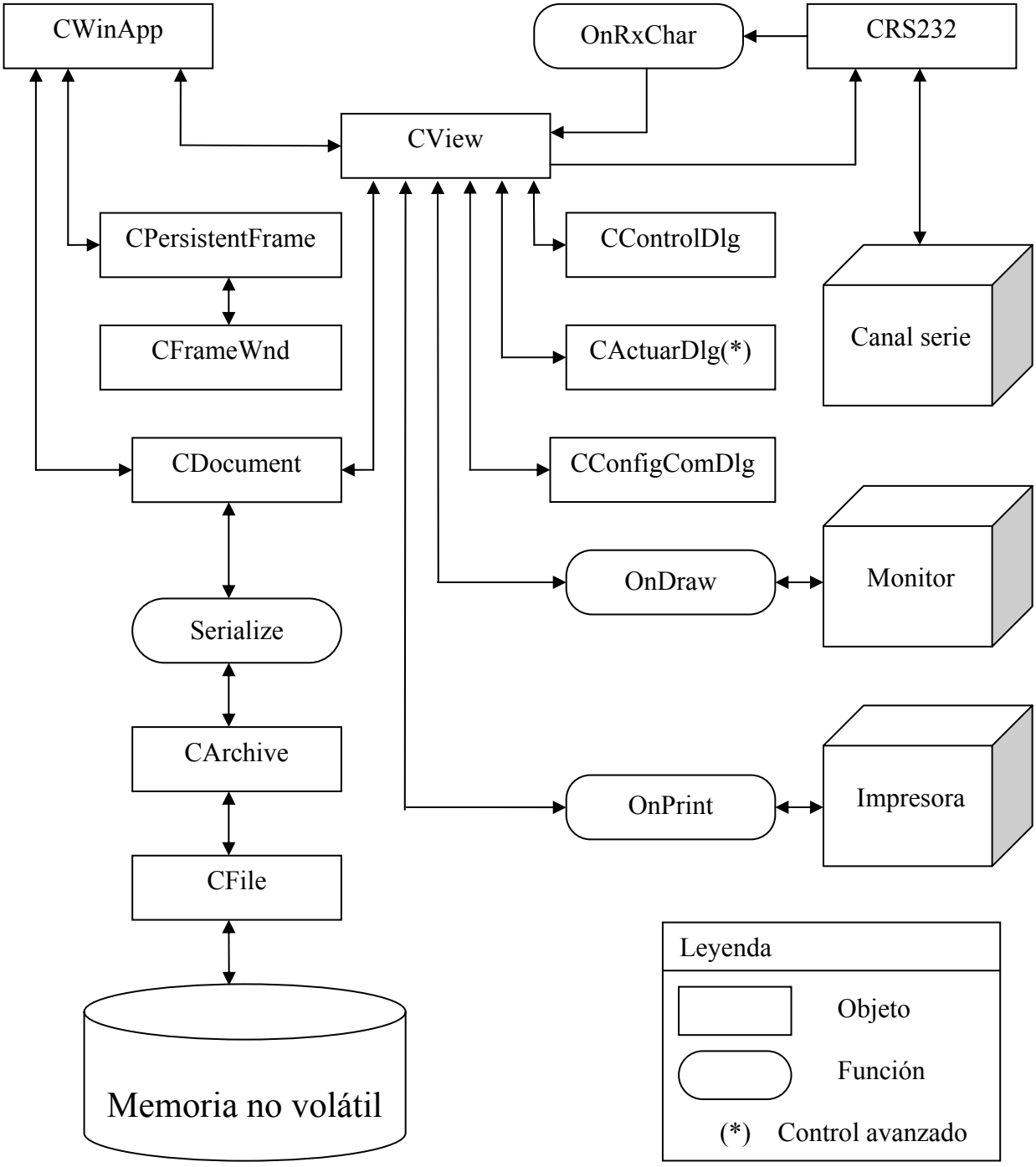

Ilustración 1.4: relación entre objetos

1.3 HARDWARE

El hardware está constituido por tres bloques:

- Circuitería digital: ésta gira entorno al microcontrolador 87C51. Gestiona una memoria externa, tiene mapeado en memoria un lach de 8 bits que controla la electrónica analógica, un conversor A/D está acoplado en el puerto cuatro y un driver 232 en los pines de comunicación
- Circuitería analógica: se dedica a procesar la señal/es analógica/s de entrada hasta llegar al conversor A/D. Está formada por cuatro etapas de entrada, las cuales poseen cada una un divisor de tensión gobernados por relés. Las señales de entrada pueden ser multiplexadas, posteriormente se aplica una ganancia variable mediante un amplificador inversor, para finalmente ser filtradas con un amplificador inversor sumador banda pasante o paso bajo.
- Circuitería de alimentación: suministra la energía necesaria para la electrónica. Existen dos bloques, uno dedicado a los elementos analógicos y otro a los digitales, de este modo se evita interferencias entre sí.

1.3.1 CIRCUITERÍA DIGITAL

1.3.1.1 Microcontrolador, memoria, líneas de control e interface serie

Seguidamente se presenta el esquema de este apartado:

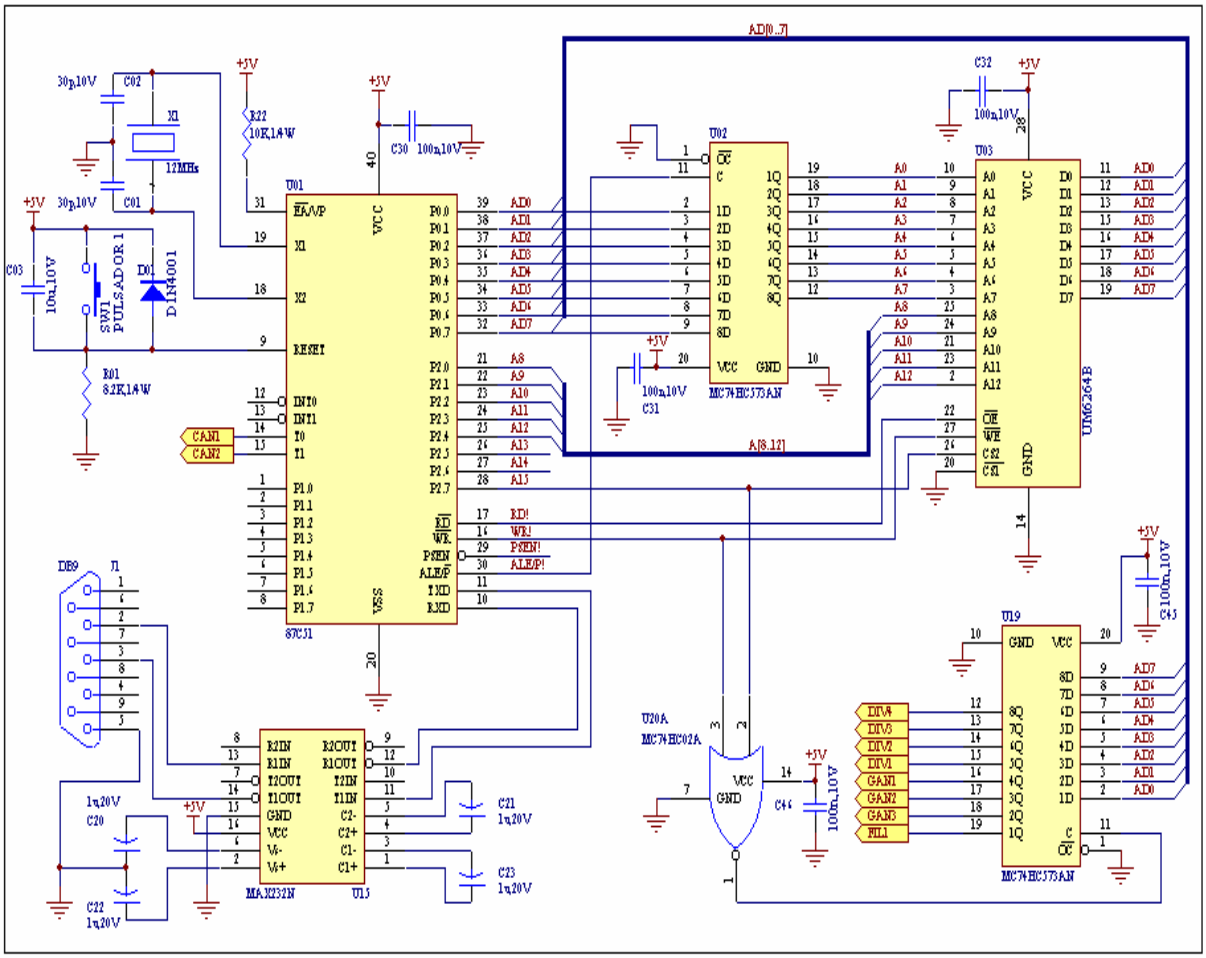

Ilustración 1.5: Microcontrolador, memoria, líneas de control e interface serie

El microcontrolador 87C51 es el encargado de gestionar toda la tarjeta de adquisición de datos y de establecer las comunicaciones con el PC mediante el canal serie. Posee una EPROM interna para poder grabar en ella el código de programa que ejecutará. Trabaja a 12 MHz, otorgados por un cristal de cuarzo de esta frecuencia. Tiene implementado un circuito de reset típico, para establecer el microcontrolador en un estado conocido en caso de perdida de estabilidad.

Posee una memoria externa que almacenará las muestras adquiridas mediante el conversor A/D (mostrado en el siguiente apartado). La estructura de control de la memoria es la clásica de un microcontrolador de estas características, es decir, se utiliza un lach acoplado en el puerto cero que mediante la línea ALE/P! mantiene en el momento correcto las direcciones, en el proceso de multiplexación entre datos y direcciones.

La memoria es del tipo RAM estática y tiene una capacidad de 8 KBy, con un ciclo de lectura de 35 ns y un ciclo de escritura de 35 ns. Estos valores son suficientes para las necesidades del microcontrolador y requisitos del proyecto.

Los datos se almacenarán de forma secuencial y se leerán de la misma manera. La estructura de la gestión de la memoria es del tipo cola FIFO (first in first out), es decir, la primera muestra almacenada en la memoria será la primera que se transmitirá.

Existe un lach mapeado en memoria que se encarga del control de la electrónica analógica junto con las líneas T0 y T1. El lach transfiere la información del puerto cero como respuesta de A15 y WR! en estado cero, esta decodificación se ha implementado mediante una puerta NOR.

En los pines TXD y RXD está acoplado un driver MAX232 para establecer los márgenes correctos de tensión de una comunicación serie. En la salida del driver se ha insertado un conector tipo D de montaje PCB, con ángulo recto y de 9 vías.

Cada circuito integrado tiene asociado un condensador de 100 nF entre su terminal de alimentación y masa. Esto es así para mantener constante esta tensión en picos de demanda de energía

1.3.1.2 Convertidor A/D

A continuación se ilustra el circuito implementado:

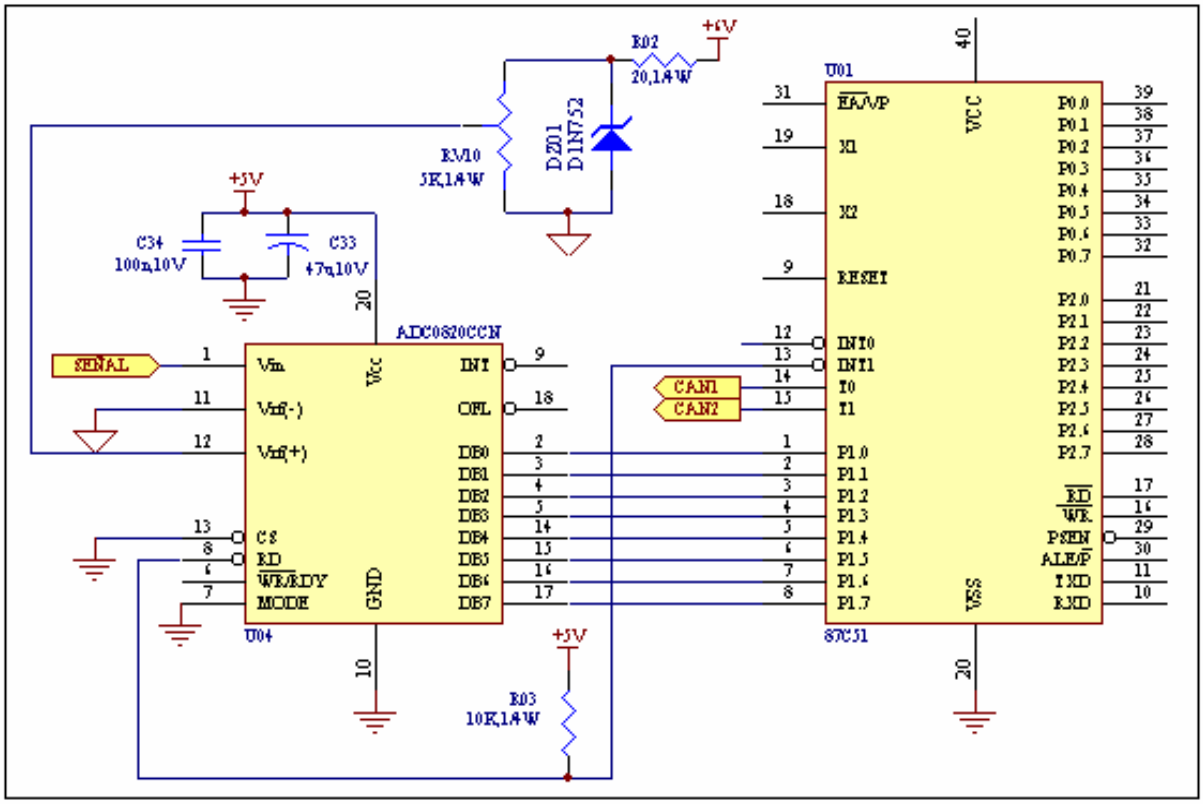

Ilustración 1.6: convertidor A/D

El convertidor A/D es un ADC0820 en configuración de lectura (RD). El montaje viene descrito en las hojas de especificaciones otorgadas por fabricante que se pueden consultar en el anexo.

El tiempo de conversión máximo en este modo es de 2.5 μs , valor más que suficiente para los requisitos del microcontrolador utilizado. Esta situación da la posibilidad futura de utilizar un microcontrolador más veloz para incrementar el ciclo de muestreo, evidentemente el cristal de cuarzo debe esta concorde a la frecuencia de uso.

Como se puede observar en convertidor A/D es de 8 bits, lo que da la posibilidad de una implementación sencilla. Éste inicia la conversión cuando el pin INT1 adquiere el estado cero y 2.5 μs después la información es válida en el puerto uno. En la línea de disparo se ha colocado una resistencia de pull-up para otorgar la energía necesaria en el estado uno, bajo esta situación el convertidor sus líneas DB0/7 en tercer estado (alta impedancia).

Según las hojas de especificación del fabricante, el conversor A/D debe trabaja entre un margen de tensiones entre 0 y 5 voltios. Existe la posibilidad de ajustar la tensión positiva de referencia mediante el potenciómetro multivuelta RV10, éste posee entre sus terminales una tensión estable otorgada por el diodo zener DZ01.

Obsérvese la segregación entre tensiones de masa analógicas de las digitales, de este modo se consigue menores interferencias entre sí, ya que ésta última posee variaciones debidas a las conmutaciones de la electrónica digital.

1.3.2 CIRCUITERÍA ANALÓGICA

1.3.2.1 Etapas de entrada y multiplexación

En la ilustración presentada se observan cuatro conectores BNC, éstos están situados en el chasis del periférico. Los conectores se unen mediante un cable plano tipo D de 10 vías a un terminal PCB encapsulado de perfil bajo, en configuración recta de 10 vías.

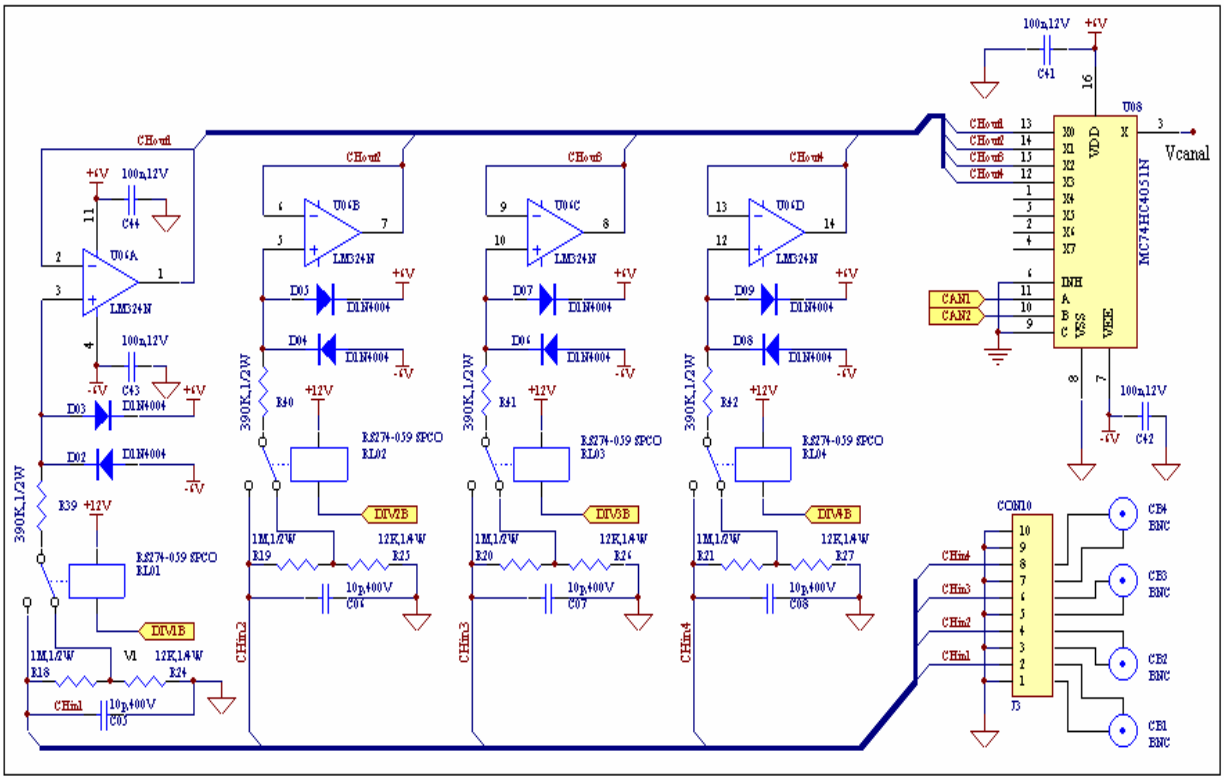

Ilustración 1.7: etapas de entrada y multiplexación

Existen cuatro etapas de entrada, asociadas cada una a su correspondiente conector BNC. Cada etapa está formada por un condensador de entrada de 10 pF y una resistencia equivalente aproximada de 1 $M\Omega$, que son valores típicos de la impedancia de entrada de un osciloscopio.

Las resistencias equivalentes de entrada están formadas a su vez por dos, creando un divisor de tensión. Los relés asociados se encargan de seleccionar entre las tensiones proveniente de los canales de entrada, atenuándola mediante el divisor de tensión o no.

A continuación se ha implementado una resistencia en serie con la salida de cada relé, ésta es superflua siempre y cuando no entren en conducción los diodos. Los diodos tienen la función de limitar la tensión otorgada a la siguiente etapa y la resistencia limitar la corriente que puede circular a través los diodos. Esta configuración recorta la tensión entre unos márgenes de 5.3 V hasta -5.3V.

Finalmente existe un multiplexor analógico de ocho entradas, donde sólo cuatro son utilizadas. Éstas adquieren las tensiones de entrada mediante unas etapas separadoras para evitar
efectos de carga. La selección de los canales se realiza mediante las líneas CAN1 y CAN2, que son los pines T0 y T1 respectivamente provenientes del microcontrolador.

Se han dimensionado los elementos para que puedan soportar la tensión de red, es decir, 220 voltios eficaces, o lo que es lo mismo 311 voltios de pico.

1.3.2.2 Amplificador inversor de ganancia variable

Para implementar la característica típica de un osciloscopio en lo que respecta a la ganancia, se ha optado por utilizar una configuración de amplificador inversor como se muestra seguidamente.

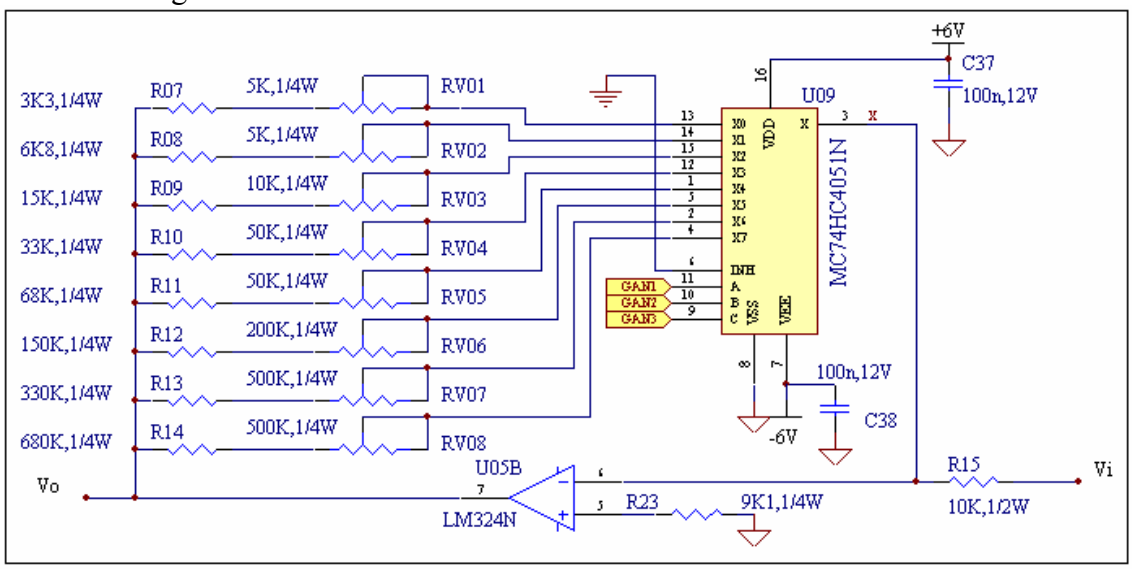

Ilustración 1.8: amplificador inversor de ganancia variable

La resistencia de entrada es constante y la resistencia de realimentación variable. Ésta última esta formada por una resistencia fija en serie con un potenciómetro multivuelta para establecer la relación de ganancia en el valor deseado.

Se puede observar que existe la posibilidad de seleccionar entre ocho posibles resistencias equivalentes de realimentación, lo que otorga ocho posibles ganancias. Éstas son seleccionadas mediante un multiplexor analógico, controlado por las líneas GAN1, GAN2 y GAN3 provenientes del lach mapeado en memoria.

Se ha implementado una resistencia entre el terminal no inversor del amplificador y masa. Su función es la de compensar las caídas internas provocadas por las corrientes de offset.

1.3.2.3 Amplificador inversor sumador banda pasante o paso bajo

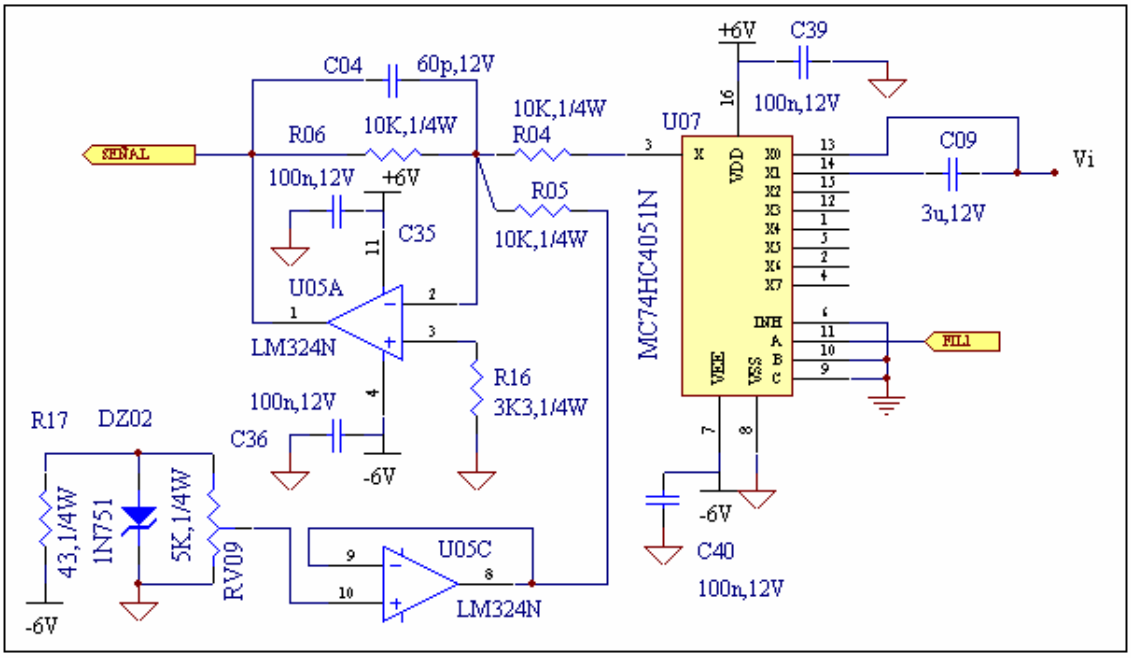

Ilustración 1.9: amplificador inversor sumador banda pasante o paso bajo

La variable SEÑAL, se dirige directamente al pin VIN del conversor A/D para ser muestreada. Este módulo realiza varias funciones.

Mediante el multiplexor analógico, controlado por la línea FIL1 proveniente del lach mapeado en memoria, se puede seleccionar el modo de funcionamiento del osciloscopio en AC o DC. Éstos corresponden al filtro banda pasante o filtro paso bajo respectivamente, es decir , si FIL es igual a cero, entonces el funcionamiento es en DC, en caso contrario es en AC.

El condensador C04 junto con R06, provocan un filtro paso bajo a la frecuencia de Niquist para evitar el solapamiento o aliassing. El condensador C09, si se selecciona con el multiplexor, ocasiona junto con la resistencia R04, un filtrado de bajas frecuencias para eliminar la componente continua de la señal.

El amplificador tiene también la característica de sumador, ésta le viene dada por el seguidor de tensión formado por U05C, que está acoplado mediante R05 en el terminal negativo. Esto es necesario debido a que el conversor trabaja en un margen de tensiones entre 0 y 5 voltios, por lo tanto la tensión en la salida del seguidor de tensión deberá ser de -2.5 voltios. Para conseguir el voltaje de referencia deseada, se ha implementado el potenciómetro multivuelta RV09, que obtiene una tensión estable mediante el diodo zener DZ02.

De igual forma que en el apartado anterior, existe la resistencia R16 para compensar las corrientes de offset.

1.3.3 CIRCUITERÍA DE ALIMENTACIÓN

1.3.3.1 Fuente de tensión de +5 voltios

La siguiente figura ilustra el circuito implementado

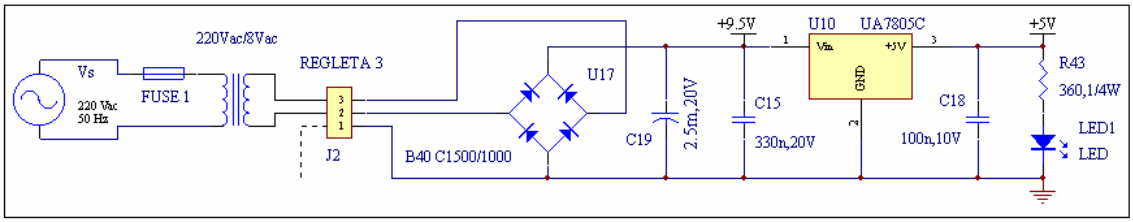

Ilustración 1.10: fuente de tensión de +5 voltios

Para otorgar la energía digital se ha implementado una fuente de tensión de 5 V, ésta gira entorno al circuito integrado UA7805. Este elemento permite un sencillo esquema de montaje a costa de un bajo rendimiento energético debido a que está constituido por un regulador de tensión lineal.

El condensador C18 otorga estabilidad en la tensión de salida ante cargar capacitivas, y el condensador C15 realiza lo propio sí el condensador de filtrado está situado lejos de C15. Los valores de estos elementos vienen dados por las recomendaciones del fabricante de este circuito integrado.

Se ha implementado un rectificador de onda completa con el puente de diodos U17. La tensión pulsante obtenida es filtrada mediante el condensador C19. Con el objetivo de no sobrecargar el UA7805, los relés de las etapas de entradas son alimentados mediante la tensión de 9.5 V, de esta forma no es necesario un disipador en el U10.

Para conseguir una tensión de entrada en U10 lo suficientemente pequeña, se hace necesaria la utilización de un transformador con una relación de transformación de 220/8, de esta forma se evita una disipación excesiva del circuito integrado.

La regleta 3 posee el terminal 1 conectado a la masa del circuito digital, que es cortocircuitado con a la masa analógica para tener una referencia común. De este modo se minimizan las interferencias acaecidas en este último por el primero.

También existe un diodo luminiscente para advertir del correcto funcionamiento del regulador de tensión, y así poder observar posibles pérdidas de tensión.

1.3.3.2 Fuente de tensión de +6 y -6 voltios

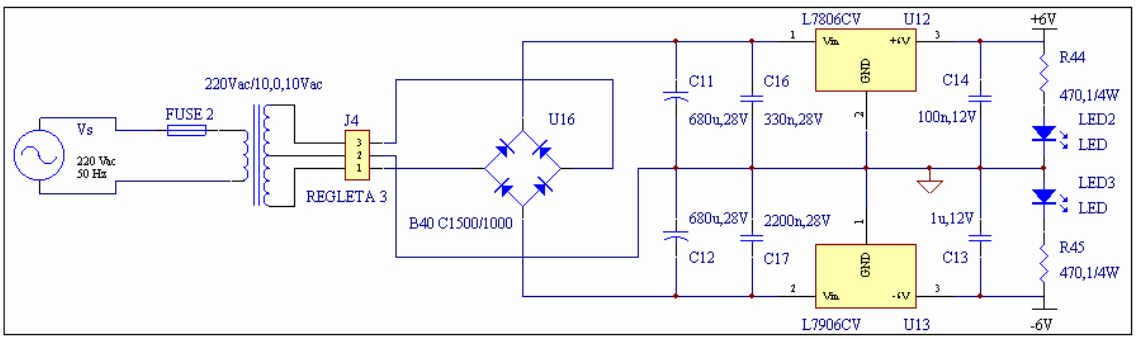

Ilustración 1.11: fuente de tensión de +6 y -6 voltios

Los razonamientos expuestos anteriormente son aplicables de igual forma en este apartado. Cabe resalta la utilización del L7906 que es el complementario del L7806 y el uso de un transformador con derivación de masa en el secundario

En este caso el condensador de filtrado está formado por el equivalente en serie de C11 y C12. El resto de condensadores vienen definidos en las hojas de especificaciones del fabricante, realizando éstos funciones equivalentes a las del apartado anterior.

El terminal 2 de la regleta se cortocircuita con el terminal 1 de la regleta de la alimentación digital. También se han un par de diodos luminiscentes para comprobar visualmente el estado de la alimentación.

1.4 SOFTWARE DE BAJO NIVEL

El software de bajo nivel está formado por instrucciones interpretables por un ensamblador del microcontrolador 8x51, que una vez linkadas pueden ser ejecutadas por este circuito integrado.

Realizar la programación basada en el microcontrolador 8x51 da la posibilidad de que el software sea compatible en versiones del hardware con mas prestaciones (8x52, 8x517, etc.), pero en contrapartida se desperdician facultades de éstos últimos.

En éste proyecto no se ha llegado a programar la EPROM del microcontrolador, en su lugar se ha utilizado un emulador del 51 debido a la inexistencia en almacén de este circuito integrado.

1.4.1 REGISTROS SFR UTILIZADOS

IE (INTERRUPT ENABLE REGISTER)							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EA	X	X	ES	ET1	EX1	ET0	EX0

Bit 0	EX0	- Si EX0 = 1 habilita interrupción externa INT0. - Si EX0 = 0 inhabilita interrupción externa INT0.
Bit 1	ET0	- Si ET0 = 1 habilita interrupción externa Timer 0. - Si ET0 = 0 inhabilita interrupción externa Timer 0.
Bit 2	EX1	- Si EX1 = 1 habilita interrupción externa INT1. - Si EX1 = 0 inhabilita interrupción externa INT1.
Bit 3	ET1	- Si ET1 = 1 habilita interrupción externa Timer 1. - Si ET1 = 0 inhabilita interrupción externa Timer 1.
Bit 4	ES	- Si ES = 1 habilita interrupción del puerto serie. - Si ES = 0 inhabilita interrupción del puerto serie.
Bit 5	X	No usada.
Bit 6	X	Reservada.
Bit 7	EA	- Si EA = 1 habilita individualmente las interrupciones que estén a uno. - Si EA = 0 no reconoce ninguna interrupción.

Cuadro 1.1: registro IE

TMOD (TIMER/COUNTER MODE CONTROL REGISTER)							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
GATE	C/T!	M1	M0	GATE			
Timer 1				Timer 0			

Bit 0 Bit 1	M0 M1	MODO	M1	M0	MODO DE OPERACIÓN
		0	0	0	Temporizador de 13 bits.
		1	0	1	Temporizador/Contador de 16 bits.
		2	1	0	Temporizador/Contador de 8 bits con Auto-recarga.
		3	1	1	Contadores múltiples específicos.
Bit 2	C/T!	Selecciona temporizador o contador. -Si C/T! = 0 entonces temporiza con los pulsos del reloj interno. -Si C/T! = 1 entonces cuenta los pulsos que llegan por T0 (pin 14).			
Bit 3	GATE	Habilita la entrada exterior INT0! (pin 12). - Si GATE = 1 entonces habilita INT0! Si TR0 = 1 (control por hard). - Si GATE = 0 entonces inhabilita INT0! Y depende sólo de TR0 (control por soft).			
Bit 4 Bit 5 Bit 6 Bit 7	M0 M1 C/T! GATE	Configuración del Timer 1. Igual que el Timer 0, sustituyendo: T0 por T1 (pin 15). INT0 por INT1 (pin 13). TR0 por TR1.			

Cuadro 1.2: registro TMOD

SCON (SERIAL PORT CONTROL REGISTER)							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Bit 0	RI	Flag de interrupción de la recepción. Se activa por hardware al finalizar la recepción del 8º bit en el Modo 0 o hacia la mitad del intervalo de tiempo del bit de stop en los otros modos (excepto ver SM2). Debe ser desactivado por software.
Bit 1	TI	Flag de interrupción de la transmisión. Se activa por hardware al finalizar la recepción del 8º bit en el Modo 0 o hacia la mitad del intervalo de tiempo del bit de stop en los otros modos. Debe ser desactivado por software.
Bit 2	RB8	En los Modos 2 y 3 es el 9º bit que se recibe. En Modo 1, si SM2 = 0, RB8 es el bit de stop. En Modo 0 no se utiliza.
Bit 3	TB8	Corresponde al 9º bit de datos en los Modos 2 y 3. Es programable por el usuario. Habitualmente es el bit de paridad.
Bit 4	REN	Si REN = 1 (por software) permite recepción. Si REN = 0 no la permite.
Bit 5	SM2	En Modo 2 y 3, Si SM2 = 1 entonces RI no se activará si el 9º bit de datos (RB8) es igual a cero. En Modo 1, si SM2 = 1 entonces RI no se activará si el bit de stop no se ha recibido. En Modo 0, SM2 debe está a cero.

Bit 6 Bit 7	SM1 SM0	MODO	SM0	SM1	DESCRIPCIÓN	VELOCIDAD
		0	0	0	Desplaza 8 bits.	Reloj / 12.
		1	0	1	UART de 8 bits.	Variable.
		2	1	0	UART de 8 bits.	Reloj / 64 o reloj / 32.
		3	1	1	UART de 8 bits.	Variable.

Cuadro 1.3: registro SCON

PCON (POWER CONTROL REGISTER)							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SMOD	X	X	X	GF1	GF1	PD	IDL

Bit 0	IDL	Bit Modo Idle. Si IDL = 1 entonces habilita este modo de operación.
Bit 1	PD	Bit Power Down. Si PD = 1 entonces activa el Modo Power Down.
Bit 2	GF0	Flag bit de propósito general.
Bit 3	GF1	Falg bit de propósito general.
Bit 4	X	No utilizado.
Bit 5	X	No utilizado.
Bit 6	X	No utilizado.
Bit 7	SMOD	Bit duplicador de baudios. Si SMOD = 1 entonces duplica la frecuencia de reloj del Timer 1 cuando éste se utiliza como generador reloj da baudios en las comunicaciones en los Modos 1, 2 y 3.

Cuadro 1.4: registro PCON

1.4.2 BYTES DEFINIDOS PARA LA APLICACIÓN

BMOD (BYTE DE MODO)							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
BMOD.7	BMOD.6	BMOD.5	BMOD.4	BMOD.3	BMOD.2	BMOD.1	BMOD.0

Bit 0 Bit 1	BMOD.0 BMOD.1	BMOD.1	BMOD.0	MODO DE MULTIPLEXACIÓN Y CANL/ES
		X	1	Sin multiplexación: Canal 1.
		1	0	Con multiplexación: Canales 1 y 2.
		0	0	Con multiplexación: Canales 1, 2, 3 y 4.
Bit 2	BMOD.2	Si BMOD.2 = 0 sin control manual, muestreo "infinito". Si BMOD.2 = 1 con control manual.		
Bit 3	BMOD.3	Si BMOD.2 = 1 entonces: Si BMOD.3 = 0 esperara activa. Si BMOD.3 = 1 Inicia muestreo y transmisión (disparo).		
Bit 4	BMOD.4	No utilizado.		
Bit 5	BMOD.5	No utilizado.		
Bit 6	BMOD.6	No utilizado.		
Bit 7	BMOD.7	No utilizado.		

Cuadro 1.5: byte BMOD

BCON (BYTE DE CONTROL)							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FAC4!	FAC3!	FAC2!	FAC1!	GAN1	GAN2	GAN3	AC/DC!

Bit 0	AC/DC!	Si AC/DC! = 1 Modo DC (con filtro paso bajo). Si AC/DC! = 0 Modo AC (con filtro banda pasante).			
Bit 1 Bit 2 Bit 3	GAN3 GAN2 GAN1	GAN3	GAN2	GAN1	GANANCIA
		0	0	0	0.5
		0	0	1	1
		0	1	0	2
		0	1	1	5
		1	0	0	10
		1	0	1	20
		1	1	0	50
		1	1	1	100
Bit 4	FAC1!	Si FAC1! = 1 Canal 1 por el factor de 0.012 (con divisor de tensión). Si FAC1! = 0 Canal 1 por el factor de 1 (sin divisor de tensión).			
Bit 5	FAC2!	Si FAC2! = 1 Canal 2 por el factor de 0.012 (con divisor de tensión). Si FAC2! = 0 Canal 2 por el factor de 1 (sin divisor de tensión).			
Bit 6	FAC3!	Si FAC3! = 1 Canal 3 por el factor de 0.012 (con divisor de tensión). Si FAC3! = 0 Canal 3 por el factor de 1 (sin divisor de tensión).			
Bit 7	FAC4!	Si FAC4! = 1 Canal 4 por el factor de 0.012 (con divisor de tensión). Si FAC4! = 0 Canal 4 por el factor de 1 (sin divisor de tensión).			

Cuadro 1.6: byte BCON

BEST (BYTE DE ESTADO)							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CH1	CH0	E	FAC!	GAN1	GAN2	GAN3	AC/DC!

Bit 0	AC/DC!	Si AC/DC! = 1 Modo DC (con filtro paso bajo). Si AC/DC! = 0 Modo AC (con filtro banda pasante).

Bit 1 Bit 2 Bit 3	GAN3 GAN2 GAN1	GAN3	GAN2	GAN1	GANANCIA
		0	0	0	0.5
		0	0	1	1
		0	1	0	2
		0	1	1	5
		1	0	0	10
		1	0	1	20
		1	1	0	50
		1	1	1	100

Bit 4	FAC!	Si FAC! = 1 Canal referenciado por el factor de 0.012 (con divisor de tensión). Si FAC! = 0 Canal referenciado por el factor de 1 (sin divisor de tensión).

Bit 5 Bit 6 Bit7	E CH0 CH1	CH1	CH0	E	CODIFICACIÓN
		0	0	0	Control de disparo (modo manual). FAC! = Enable, GAN1 = TRIGGER.
		0	0	1	Referenciado Canal 1.
		0	1	0	Referenciado Canal 2 e inhabilitado Canal 2.
		0	1	1	Referenciado Canal 2 e habilitado Canal 2.
		1	0	0	Frecuencia de muestreo variable. No implementado.
		1	0	1	Referenciado Canal 3.
		1	1	0	Referenciado Canal 4 e inhabilitado Canal 2, 3 y 4.
		1	1	1	Referenciado Canal 4 e habilitado Canal 2, 3 y 4.

Cuadro 1.7: byte BEST

1.4.3 INICIALIZACIÓN

En la inicialización estable la configuración por omisión de la electrónica del periférico. Está formado por dos grupos ejecutados de forma secuencial.

1) Programación del estado futuro de la electrónica analógica

Se hace uso de la subárea direccionable bit a bit. Ésta está comprendida entre las direcciones 20H y 2FH del área de direccionamiento directo e indirecto.

En el contenido de la dirección 20H, expresado como (20H) se almacena el modo de funcionamiento del periférico, es decir, si no se realizará multiplexación (sólo habilitado el canal 1), si se realizara multiplexación de los canales 1 y 2 o de los canales 1, 2, 3 y 4. También establece si se realizarán muestreos y transmisiones indeterminadamente (sin control manual), o por el contrario, si es el usuario el que da la orden para realizar una operación de muestreo y transmisión (con control manual). Este byte se denominara Byte de Modo o BMOD.

BMOD se configura inicialmente sin multiplexación y sin control manual.

El lach mapeado en memoria, que desde ahora se denominará Byte de Control o BCON, establece el estado de los relés de las etapas de entrada, la ganancia aplicada en el amplificador inversor y el modo AC o DC.

Al existir cuatro canales, cada uno de ellos debe poseer su propio BCON. Con este propósito se utilizan cuatro bytes (21H), (22H), (23H) y (24H), denominados BCON1, BCON2, BCON3 y BCON4. La parte alta será igual para todos, pero la baja específica de cada uno.

BCON1, BCON2, BCON3 y BCON4 se establecen sin divisor de tensión de entrada, con la ganancia mínima y en modo DC.

Se definen cuatro Bytes de Estado (25H), (26H), (27H) y (28H) denominados BEST1, BEST2, BEST3 y BEST4. Estos bytes codifican el estado actual de cada canal. Son recibidos desde el PC e informan sobre sí cada canal debe estar habilitado o no, sí se atenúan con el divisor de tensión, de su ganancia y del modo AC o CD. La parte baja cada uno coincide con la parte baja de su BCON homólogo.

BEST1, BEST2, BEST3 y BEST4 se configuran de tal forma que sólo el Canal 1 está habilitado, todos los canales sin divisor de tensión, ganancia mínima y modo DC.

Como el Canal 1 siempre está habilitado, entonces en el supuesto estado de inhabilitación se codifica el disparo del control manual. Ocurre algo similar con el Canal 3, ya que este se ha subordinado al Canal 4. Se habilita o inhabilita el Canal 3 si lo está el Canal 4, en consecuencia se aprovecha está situación para codificar la orden de muestreo a frecuencia variable.

2) Programación del interface de comunicaciones serie

Inicialmente se habilita la interrupción serie mediante el registro IE, ya que la recepción se gestionará de forma asíncrona para permitir el control manual.

Se configura el Timer 1 en modo de 8 bits con autorrecarga y disparado por software mediante el registro TMOD. Este otorga la frecuencia de comunicación por el canal serie.

Se programa el control canal serie como una UART (Universat Asynchronous Receiver and Transmiter) de 8 bits con 1 bit de start, 8 bits de datos y 1 bit de stop mediante el registro SCON.

Estableciendo el bit SMOD igual a 1 del registro PCON, se consigue duplicar la frecuencia de comunicación.

Finalmente se cargan los registros TL1 y TH1 con el valor necesario para conseguir 9600 baudios y se dispara el Timer 1 para iniciar la comunicación.

1.4.4 PRELUDIO AL MUESTREO Y ESCRITURA EN MEMORIA

Pertenecen a este apartado las instrucciones previas al muestreo. La secuencia seguida es la siguiente:

Se guarda en memoria la trama de inicio. Ésta sirve para que el PC pueda reconocer el inicio de una transmisión y está formada por la secuencia 0xFB, 0x04, 0xFB y 0x04.

Seguidamente se realiza una copia del Byte de Modo, los Bytes de Control y los Bytes de Estado a la subárea de memoria Scratch Pad desde la dirección 30H hasta la 38H. En dicho proceso se tiene la precaución de inhabilitar la interrupción serie para que no se actualice ningún Byte en el proceso de copia.

Inmediatamente después de la trama de inicio se guarda en memoria los BEST de los canales para que el PC pueda saber como han sido muestreados, y así poder presentar la información al usuario correctamente.

Finalmente se comprueba si está habilitado el control manual, y si es así, entonces se espera indeterminadamente al disparo proveniente del PC. Si no esta habilitado el control manual simplemente se inicia el muestreo.

1.4.5 MUESTREO Y ESCRITURA EN MEMORIA

En los instantes iniciales se determina el tipo de multiplexación requerido, es decir, sin multiplexación (Canal 1), multiplexación de los Canales 1 y 2 o de los Canales 1, 2, 3 y 4.

Si sólo se utiliza el Canal 1 se consigue la máxima frecuencia de muestreo, si se multiplexan todos los canales se obtiene la situación contraria. La secuencia del proceso es similar en todos los modos.

Se escribe en BCON el BCON1 y se realiza una conversión.

Comienza un bucle en el que se realizan 1012 muestras y escrituras en memoria, teniendo en cuenta los requisitos de multiplexación. Esto implica cargar en BCON el correspondiente byte de control del canal que se va a muestrear.

Mediante las bits T0 y T1 se escoge el canal a muestrear. Estos bits se dirigen directamente al multiplexor analógico de las etapas de entrada.

Finalmente se escribe en memoria la trama final formada por la secuencia 0x04, 0xFB, 0x04 y 0xFB. Esto otorga al PC la posibilidad de contar el número de bytes recibidos.

1.4.6 LECTURA DE MEMORIA Y TRANSMISIÓN

Este proceso está formado por un bucle que se ejecuta 1024 veces. En cada pasada se realiza una lectura de memoria y posterior transmisión. Dentro de este lazo se encuesta el flag TI que indica el final de una transmisión en curso para realizar una nueva.

El cuello de botella del conjunto del programa en ensamblador radica en este apartado, ya que la velocidad de comunicación es baja. No se puede enviar un gran número de muestras porque esto ocasionaría un tiempo de respuesta del periférico excesivo grande.

1.4.7 RECEPCIÓN

La recepción de información se ha implementado mediante la interrupción serie. De esta manera existe la posibilidad del modo de funcionamiento manual y posterior disparo. El byte recibido corresponde con el Byte de Estado referenciado

Cuando ha acaecido una recepción, se carga el byte recibido y mediante la parte baja se crea una máscara de unos y otra de ceros, que en función del canal referenciado se aplica a su correspondiente BCON. Esto es así porque la codificación de los cuatro bits de menos peso del BEST recibido es idéntica para los cuatro canales.

También se actualizan todos los BCON de los canales en lo que respecta al control del relé direccionado (bit 4 del BEST), ya que la parte alta de los BCON es común para todos los canales.

1.5 SOFTWARE DE ALTO NIVEL

El lenguaje de programación que se ha utilizado es el Microsoft Visual C++ 5.0.

El programa implementado realiza las siguientes funciones:

1) Controlar las funciones del periférico.
2) Presentar por pantalla la información adquirida mediante el periférico.
3) Presentar por impresora la información adquirida mediante el periférico.
4) Guardar en memoria no volátil los datos.
5) Configurar la comunicación serie.

1.5.1 VENTANA DE MARCO PRINCIPAL

La ventana de marco principal presenta el aspecto típico de una aplicación Microsoft Windows, por lo que al usuario no le costará familiarizarse con los aspectos específicos del programa.

Este marco tiene la característica notable de ser persistente, es decir, recuerda la posición, tamaño y estado de la ventana de marco, así como de las barras de herramientas y estado y diálogos asociados. Esto se consigue utilizando el registro de Windows para almacenas estos datos en el disco duro.

1.5.1.1 Barra de título

Esta barra presenta el icono de la aplicación, el nombre del documento actual, el nombre de la aplicación y los botones maximizar, minimizar y cerrar.

Pulsando con el botón izquierdo sobre el icono o con el botón derecho sobre la barra, se obtiene un menú con los elementos Mover, Tamaño, Maximizar, Minimizar y Cerrar. Éstos realizan las funciones típicas de cualquier aplicación de Microsoft Windows 95

1.5.1.2 Barra de menú

Posee cuatro elementos de menú desplegables, a continuación se describe sus funciones:

1) **Archivo:**
   1.1) Nuevo: crea un nuevo documento.
   1.2) Abrir: abre un documento.
   1.3) Guardar: guarda un documento en disco.
   1.4) Guardar como: guarda un documento con el nombre deseado.
   1.5) Imprimir: Imprime un documento.
   1.6) Presentación preliminar: muestra la vista de la presentación preliminar a la impresión.
   1.7) Configurar impresora: configura impresora.
   1.8) Archivos recientes: presenta los nombres de los cuatro últimos archivos guardados.
   1.9) Salir: sale de la aplicación.
2) **Scope:**
   2.1) Controles: presenta el diálogo para el control del osciloscopio.
   2.2) Controles avanzados: presenta el diálogo para el control avanzado del osciloscopio, también posee dos cuadros de edición que dan la posibilidad de monitorizar las comunicaciones.
   2.3) Abrir puerto: Abre el puerto de comunicaciones o lo cierra
   2.4) Configurar puerto: presenta el diálogo para la configuración las comunicaciones serie.
3) **Ver:**
   3.1) Barra de herramientas: presenta o guarda la barra de herramientas.
   3.2) Barra de estado: presenta o guarda la barra de estado.
4) **Ayuda:**
   4.1) Acerca de: presenta el diálogo informativo de la aplicación

1.5.2 VENTANAS HIJAS

Estas ventanas son todas aquellas que dependen de la ventana de marco principal

1.5.2.1 Ventana de barra de herramientas

Es una barra de herramientas reubicable, que gracias a la característica persistente del marco principal, recuerda la posición de ésta cuando se sale de la aplicación. Seguidamente se describen sus elementos.

1) Nuevo: crea un nuevo documento.
2) Abrir: abre un documento.
3) Guardar: guarda un documento.
4) Control: abre el diálogo del control del osciloscopio.
5) Control avanzado: abre el diálogo del control avanzado.
6) Abrir puerto: abre o cierra el puerto de comunicaciones.
7) Configurar puerto: abre el diálogo de configuración del puerto.
8) Imprimir: imprime el documento activo.
9) Acerca de: presenta la información de la aplicación.

1.5.2.2 Ventana de vista

En esta venta se presenta al usuario de forma gráfica la información procedente del periférico. Sufre los cambios provocados por las ventanas de diálogo de Control y Control avanzado.

Posee la rejilla de 10 x 10 cuadros típicos de un osciloscopio. Mantiene una relación de visualización 1:1 para que no se distorsione la imagen, a pesar de poder modificar el tamaño de la misma.

1.5.2.3 Ventana de diálogo de control

Este diálogo proporciona un interface agradable y sencillo para el control del osciloscopio. Seguidamente se describen sus funciones:

1) Cuadro de verificación **Control interactivo**: si está habilitado otorga al diálogo la capacidad de controlar simultáneamente la vista y el periférico, en caso contrario sólo controla la vista.
2) Cuadro de grupo **Canales**: éste sólo es útil si está habilitado el control interactivo.
   3.1) Botón de radio Canal 1: habilita el canal 1 .
   3.3) Botón de radio Canales 1 y 2: habilita los canales 1 y 2
   3.4) Botón de radio Canales 1, 2, 3 y 4: habilita los canales 1, 2, 3 y 4.
3) Cuadro de grupo **Control manual**:
   3.1) Cuadro de verificación Habilitado: habilita el control manual.
   3.2) Botón Disparo: provoca que se reciba un bloque de datos provenientes del osciloscopio.
4) Cuadros de grupo Canal 1, Canal 2, Canal 3 y Canal 4:
   4.1) **DC**: establece su correspondiente canal en modo DC.
   4.2) **AC**: establece su correspondiente canal en modo AC.
5) Cuadro de grupo **V / div**: establece los voltios / división.
   5.1) Barra de deslizamiento Canal 1: establece los V / div del Canal 1.
   5.2) Barra de deslizamiento Canal 2: establece los V / div del Canal 2.
   5.3) Barra de deslizamiento Canal 3: establece los V / div del Canal 3.
   5.4) Barra de deslizamiento Canal 4: establece los V / div del Canal 4.
6) Barra de deslizamiento **ms / div**: establece los ms / división

Este diálogo se abre por omisión al ejecutar la aplicación. Aunque se cierre no se pierden los datos asociados a éste

1.5.2.4 Ventana de diálogo de control avanzado

Este diálogo proporciona un austero control del osciloscopio y la posibilidad de monitorizar las comunicaciones mediante dos cuadros de edición.

1) Botón **Aceptar**: cierra el diálogo y actualiza su datos miembros hacia la vista.
2) Botón **Cancelar**: cierra el diálogo.
3) Botón **Aplicar**: presenta en el cuadro de edición By Out la codificación de los botones de radio y el cuadro de verificación Enable.
4) Botón **Enviar**: envía la codificación creada por el botón Aplicar.
5) Cuadro de verificación **Simular**: habilita la simulación del osciloscopio enviando por el canal serie un bloque de datos según el los canales habilitados con el botón Enable. Para apreciar el resultado deben cortocicuitarse los pines de transmisión y recepción del puerto serie del PC.
6) Cuadro de verificación **Tramas**: si está inhabilitado sólo se presenta información del canal serie en los cuadros de edición, en caso contrario las comunicaciones en la recepción deben poseer una trama de inicio y otra de final.
7) Cuadro de grupo **Canal**: los botones de radio de este cuadro codifican el canal correspondiente mediante el botón Aplicar.
8) Cuadro de verificación **Enable**: habilita el canal seleccionado mediante el botón Aplicar.
9) Cuadro de grupo **Por 0,012**: sus botones de radio codifican el divisor de tensión.
10) Cuadro de grupo **Por**: sus botones de radio codifican el amplificador inversor de ganancia variable.
11) Cuadro de grupo **Filtro**: sus botones de radio codifican el modo AC o DC.
12) Cuadro de edición **By Out**: monitoriza los bytes que están apunto de ser enviados o están siendo enviados.
    12.1) Botón **Limpiar**: limpia el cuadro de edición By Out.
    12.2) Cuadro de verificación **E**: habilita el cuadro de edición By Out.
13) Cuadro de edición **By In**: monitoriza los bytes que están siendo recibidos.
    13.1) Botón **Limpiar**: limpia el cuadro de edición By In.
    13.2) Cuadro de verificación **E**: habilita el cuadro de edición By In.

1.5.2.5 Ventana de diálogo de configuración de comunicaciones

Este diálogo otorga la posibilidad de escoger el puerto de comunicaciones serie disponible y la velocidad de comunicación.

1) Cuadro de grupo **Puerto**: sus botones de radio dan la posibilidad de escoger entre los puerto de comunicación COM1 o COM2.
2) Cuadro de grupo **Baudios**: sus botones de radio permiten escoger entre un conjunto de velocidades de comunicación, pero el periférico desarrollado sólo se comunica a 9600 baudios.

1.5.3 MODO DE EMPLEO

El modo de utilización del osciloscopio por un usuario es el siguiente:

1) Configuración de las comunicaciones: mediante el diálogo de **Configuración de comunicaciones**, se establece el puerto serie en el cual se conecta el periférico (COM1 o COM2). La velocidad de comunicación debe ser de 9600 baudios. Sólo es necesario realizar la configuración de las comunicaciones cada vez que se cambie el periférico de periférico de puerto serie, y no en cada vez que se inicie la aplicación.
2) Abrir puerto de comunicaciones: presionando el elemento de menú o botón **Abrir puerto** se abre el canal serie, si la operación es satisfactoria, un diálogo informativo alude a la acción realizada.
3) Controlar el osciloscopio: con el diálogo **Controles**, que por omisión se presenta al arrancar la aplicación, se tiene acceso a la al modo de utilización del osciloscopio. Para que el funcionamiento sea correcto, el cuadro de verificación **Control interactivo** debe estar habilitado.
4) Guardar datos e impresión: mediante el elemento de menú **Archivo**, se tiene acceso a estas características. Su utilización es equivalente a la de otras aplicaciones Microsoft Windows.

- ADVERTENCIA: No es recomendable utilizar el diálogo de **Control avanzado** si no está seguro de sus consecuencias.

1.6 CARACTERÍSTICAS OBTENIDAS

Escalas de tensión en voltios:
0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50, 100.
La frecuencia máxima de muestreo: 52 KHz.
Escalas de tiempo en mili segundos:
0.05, 0.1, 0.2, 0.5, 1, 2, 5.
La tensión máxima aproximada medible: 310 V.

2 MEMORIA DE CÁLCULO

2.1 HARDWARE

2.1.1 CIRCUITERÍA DIGITAL

El esquema presentado a continuación ilustra este apartado:

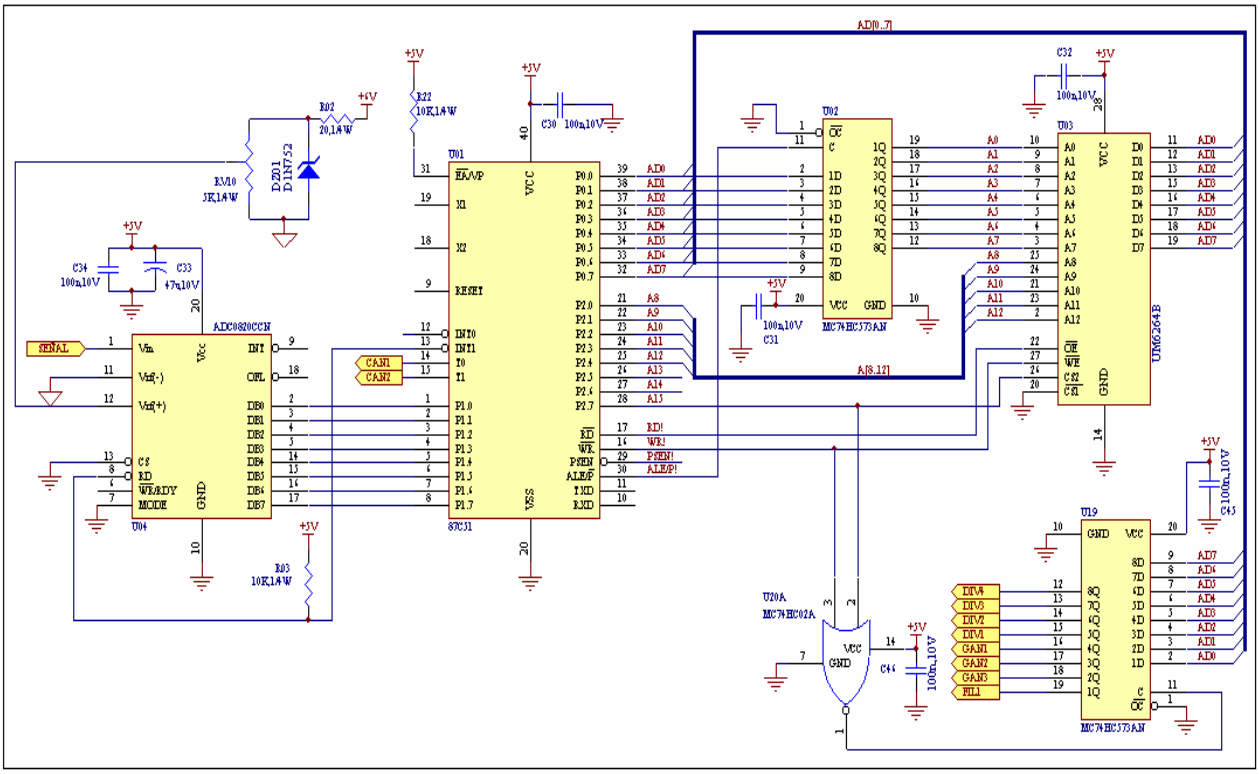

Ilustración 2.1: circuitería digital

Existen pocos cálculos que realizar sobre la electrónica digital. Por simple inspección se pueden determinar las direcciones de la memoria externa.

Dirección inicial de la memoria externa = 1xx0 0000 0000 0000 b = 0x8000.

Dirección Final de la memoria externa = 1xx1 1111 1111 1111 b = 0x8FFF.

Para mapear en memoria el lach U19, que es el encargado de gestionar la electrónica analógica, se utiliza una puerta NOR, de tal forma que sólo cuando A15 = 0 y WR! = 0 se lachea el contenido del puerto cero.

Para conseguir una tensión de referencia estable en el terminal positivo del convertidor A/D se ha utilizado un diodo zener de 5.6V. Por éste deben circular 20mA para mantener entre sus bornes una tensión estable, entonces

$$R_{02} = \frac{6 - 5.6}{20 \cdot 10^{-3}} = 20\Omega$$

Mediante el potenciómetro multivuelta RV10 se consigue ajusta la tensión de referencia a un valor de 5V como el fabricante especifica.

Nótese que en la ilustración presentada no están representados el circuito de reset ni el de oscilación. Esto es así por que éstos no participan en cálculo alguno.

2.1.2 CIRCUITERÍA ANALÓGICA

2.1.2.1 Etapa de entrada y multiplexación

El sistema de adquisición de datos consta de cuatro entradas; se presenta el diseño y análisis de una sola de ellas debido a que las tres restantes son equivalentes. En la ilustración 2.2 se presenta una de éstas.

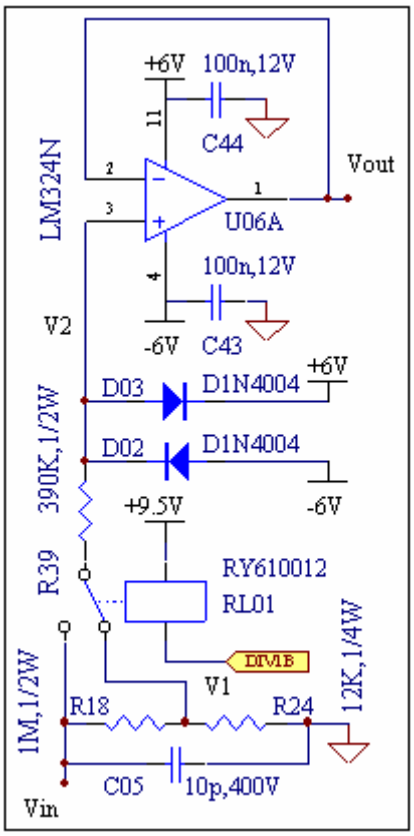

Ilustración 2.2: etapa de entrada

Vin es la señal que se desea medir y presenta un acoplamiento mediante una impedancia de entrada de aproximadamente $1M\Omega$ en paralelo con $10pF$, que son unos valores típicos para un osciloscopio.

El puerto DIV1B proviene de un buffer (circuito integrado U18) formado por un Darlington, cuya base a su vez es alimentada mediante el pin 16 (puerto DIV1) del circuito integrado U19. Ha sido necesaria la utilización de un buffer ya que la corriente demanda por el relé es muy superior a la que puede otorgar el lach MC74HC573 (U19). Para poder observar el conjunto del diseño refiérase al los planos.

Mediante el relé $RL01$ se selecciona $V1$ o Vin del siguiente modo:

Si $Vin \leq 5V \Rightarrow V2 = Vin$

Si $Vin > 5V \Rightarrow V2 = V1 = \frac{R_{24}}{R_{24} + R_{18}} Vin = \frac{12}{12 + 1000} Vin \cong 1.2 \cdot 10^{-2} \cdot Vin$

El divisor de tensión se ha diseñado de tal forma que para una tensión de entrada de 400V (la tensión máxima que se desea medir es de $220 \cdot \sqrt{2} \cong 311\,\text{V}$, pero para aumentar la seguridad se a aumentado en 89V) la salida ($V1$) sea de 5V.

Mediante esta estructura se consigue que la tensión $V2$ permanezca en un rango desde -5V hasta +5V; esto es necesario, ya que la electrónica que posteriormente

procesará la señal está alimentada a $\pm 6V$, lo que implica que tensiones superiores a éstas provocarían una saturación de componentes electrónicos (Se ha acotado la señal un voltio por arriba y por debajo de la alimentación analógica para evitar la poca linealidad de los componentes en zonas cercanas a los máximos de tensión posibles).

Nótese que R_{39} es superflua siempre y cuando D_{03} o D_{04} estén en corte. Cuando $V2$ alcance una tensión de 6.6 V o -6.6 V entonces D_{03} o D_{04} estarán en conducción respectivamente. En la situación de máxima tensión en la entrada (400 V) la corriente que circulará por R_{39} es

$$I_{R39} = V_{R39} / R_{39} = (400 - 6.6) / 390000 \cong 1mA$$

y la potencia máxima que disipará

$$P_{R39} = I_{R39}^2 \cdot R_{39} = (1mA)^2 \cdot 380K\Omega = 0.38W \Rightarrow 1/2\,W .$$

A continuación se hará un análisis para determinar la potencia máxima consumida por el divisor de tensión y la resistencia limitadora R_{39}. En la ilustración 2.3 se presenta el esquema eléctrico.

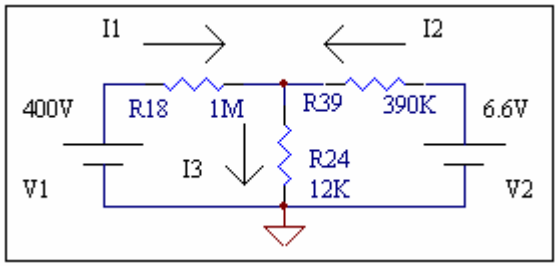

Ilustración 2.3: potencia de entrada

Si se plantea la malla exterior:

$$V_1 - V_{R18} = V_2 - V_{R3}$$

Despejando I_1

$$V_1 - R_{18} \cdot I_1 = V_2 - R_{39} \cdot I_3 \Rightarrow I_1 = \frac{V_1 - V_2 + I_3 \cdot R_{39}}{R_{18}}$$

La malla interior derecha:

$$V_2 = V_{R39} + V_{R24}$$

Expresándolo en función de I_2

$$V_2 = R_{39} \cdot I_3 + R_{24} \cdot I_3 \Rightarrow I_2 = \frac{V_2 - I_3 \cdot R_{24}}{R_{39}}$$

Planteando la ecuación del nodo

$$I_1 + I_2 = I_3$$

Sustituyendo I_1 e I_2 en la ecuación anterior y expresándolo en función de I_3

$$\frac{V_1 - V_2 + I_3 \cdot R_{39}}{R_{18}} + \frac{V_2 - I_3 \cdot R_{24}}{R_{39}} = I_3 \Rightarrow$$

$$R_{39} \cdot (V_1 - V_2) + R_{39}^2 \cdot I_3 + R_{18} \cdot V_2 - R_{24} \cdot R_{18} \cdot I_3 = R_{18} \cdot R_{39} \cdot I_3 \Rightarrow$$

$$I_3 = \frac{R_{39} \cdot (V_1 - V_2) + R_{18} \cdot V_2}{R_{24} \cdot R_{18} + R_{39} \cdot R_{18} - R_{39}^2}$$

Sustituyendo valores:

$$I_3 = \frac{390\cdot10^3\cdot(400-6.6)+6.6\cdot10^6}{12\cdot10^3\cdot10^6+390\cdot10^3\cdot10^6-(390\cdot10^3)^2} \cong 640\mu A \Rightarrow$$

$$P_{R24} = (640\mu A)^2\cdot 12K\Omega \cong 5mW \Rightarrow {}^{1}\!/\!_{4}\,W$$

Entonces

$$I_1 = \frac{400-6.6+6.4\cdot10^{-4}\cdot390\cdot10^3}{10^6} \cong 643\mu A \Rightarrow P_{R18} = (643\mu A)^2\cdot 1M\Omega \cong 0.4W \Rightarrow {}^{1}\!/\!_{2}\,W$$

Se ha insertado un seguidor de tensión entre V_2 y V_{out} para evitar efectos de carga desde la señal de entrada al siguiente bloque, que está formado por un multiplexor analógico seguido de un amplificador inversor. El multiplexor es el encargado de ir seleccionando alternadamente entre los cuatro canales disponibles. Los canales se seleccionan mediante los puertos CAN1 y CAN2 que son directamente los pines 14 (T0) y 15 (T1) del 87C51.

2.1.2.2 Amplificador inversor de ganancia variable

Se ha utilizado la estructura básica de un amplificador inversor cuya resistencia de realimentación puede ser escogida mediante un multiplexor analógico como se muestra en la siguiente figura.

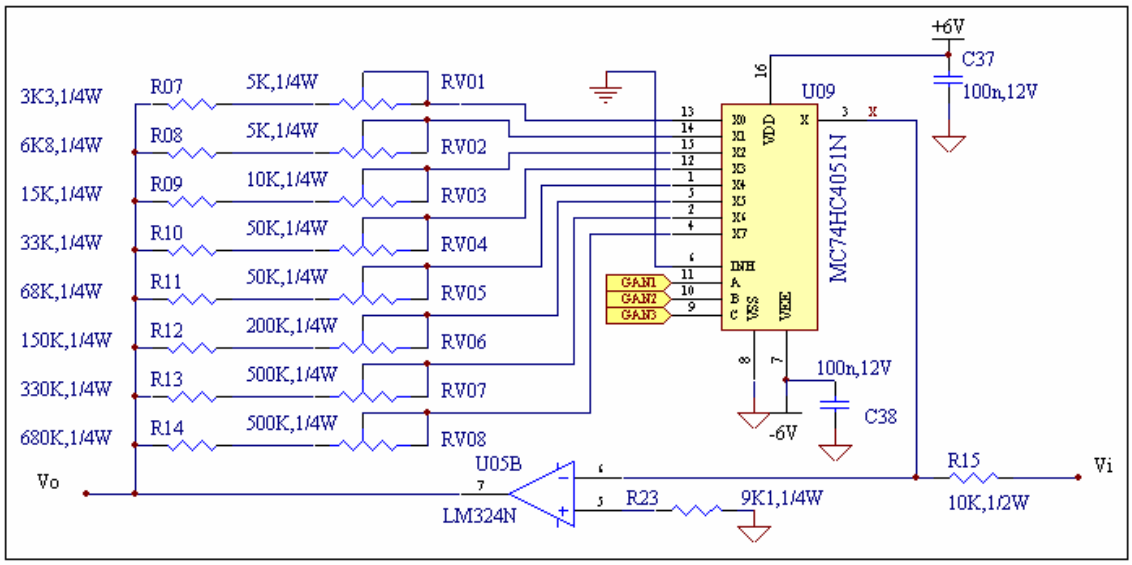

Ilustración 2.4: amplificador inversor de ganancia variable

La función de transferencia de un amplificador inversor es la siguiente:

$$H(S)=\frac{V_o(S)}{V_i(S)}=-\frac{R_X}{R_{15}}=G_X$$

Donde R_X es la resistencia equivalente de una de las entradas del multiplexor y G_X es una de las ocho posibles ganancias.

Los diodos de las etapas de entrada limitan la señal en un rango de $[-6.6,+6.6]$V y como se ha expuesto anteriormente ocurrirá una saturación en la electrónica si en algún punto la señal excede el valor absoluto de 6V. Cuando se de esta situación será el momento de provocar una amplificación menor mediante el divisor de tensión y/o el amplificador de ganancia variable.

Siguiendo el criterio expuesto en el apartado anterior, la tensión máxima que se procesará en cualquier punto a partir del seguidor de tensión de la etapa de entrada, será de 5V, así se evita la zona poco lineal en los extremos de la alimentación.

Las ganancias típicas de un osciloscopio son los múltiplos y submúltiplos de 1, 2 y 5. El rango máximo de salida del amplificador inversor debe ser de $[-2.5,+2.5]$V debido a que el conversor analógico/digital hace un muestreo dentro de un rango de 5V. La tensión máxima que se desea procesar en la entrada del amplificador inversor es de 5V. Entonces la ganancia mínima será

$$G_1=\frac{V_{omax}}{V_{imin}}=\frac{2.5}{5}=0.5$$

Si $R_{15}=10K\Omega$ y utilizando las ganancias típicas de un osciloscopio se obtiene

GANAN-CIA	VALOR IDEAL R_X	VALOR DESEADO -25%	VALOR COMERCIAL	COMPENSA-CIÓN 50%	VALOR COMERCIAL
$G_1 = 0.5$	$R_7 = G_1 \cdot R_{15} = 5K\Omega$	$R_7 = 3K75\Omega$	$R_7 = 3K3\Omega$	$RV_1 = 2K5\Omega$	$RV_1 = 5K\Omega$
$G_2 = 1$	$R_8 = G_2 \cdot R_{15} = 10K\Omega$	$R_8 = 7K5\Omega$	$R_8 = 6K8\Omega$	$RV_2 = 5K\Omega$	$RV_2 = 5K\Omega$
$G_3 = 2$	$R_9 = G_3 \cdot R_{15} = 20K\Omega$	$R_9 = 15K\Omega$	$R_9 = 15K\Omega$	$RV_3 = 10K\Omega$	$RV_3 = 10K\Omega$
$G_4 = 5$	$R_{10} = G_4 \cdot R_{15} = 50K\Omega$	$R_{10} = 37K\Omega$	$R_{10} = 33K\Omega$	$RV_4 = 25K\Omega$	$RV_4 = 50K\Omega$
$G_5 = 10$	$R_{11} = G_5 \cdot R_{15} = 100K\Omega$	$R_{11} = 75K\Omega$	$R_{11} = 68K\Omega$	$RV_5 = 50K\Omega$	$RV_5 = 50K\Omega$
$G_6 = 20$	$R_{12} = G_6 \cdot R_{15} = 200K\Omega$	$R_{12} = 150K\Omega$	$R_{12} = 150K\Omega$	$RV_6 = 100K\Omega$	$RV_6 = 200K\Omega$
$G_7 = 50$	$R_{13} = G_7 \cdot R_{15} = 500K\Omega$	$R_{13} = 375K\Omega$	$R_{13} = 330K\Omega$	$RV_7 = 250K\Omega$	$RV_7 = 500K\Omega$
$G_8 = 100$	$R_{14} = G_8 \cdot R_{15} = 1M\Omega$	$R_{14} = 750K\Omega$	$R_{14} = 680K\Omega$	$RV_8 = 500K\Omega$	$RV_8 = 500K\Omega$

Cuadro 2.1: resistencias del amplificador inversor de ganancia variable

En el diseño, se asocia a cada resistencia fija un potenciómetro multivuelta en serie para calibrar individualmente cada ganancia, esto es necesario debido a las tolerancias inherentes de las resistencias fijas. Cabría la posibilidad de poner simplemente pontenciómetros pero de esta forma se perdería sensibilidad en el calibrado de las ganancias.

Obsérvese la inserción de la resistencia R_{23}, ésta tiene la función de compensar la caída de tensión interna del amplificador operacional producida por la corriente de offset. Se ha optado por usar el equivalente en paralelo de R_{15} con R_{11}, entonces

$$R_{23} = \frac{R_{15} \cdot R_{11}}{R_{15} + R_{11}} = \frac{10 \cdot 100}{10 + 100} \cong 9K1\Omega$$

Finalmente cabe exponer que la selección de la ganancia deseada se realiza mediante los puertos GAN1, GAN2 y GAN3 del multiplexor, los cuales provienen del circuito integrado U19 (lach MC74HC573).

2.1.2.3 Amplificador inversor sumador banda pasante o paso bajo

A continuación se presenta la estructura básica de un sumador inversor.

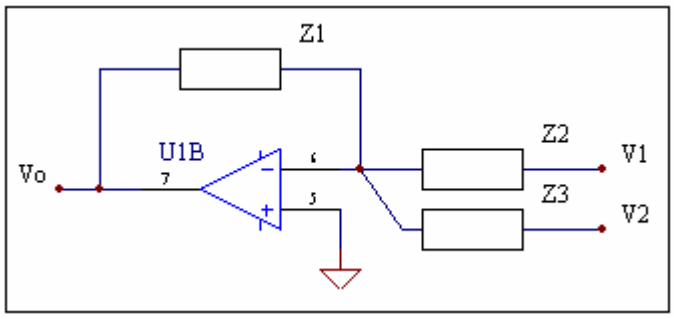

Ilustración 2.5: estructura amplificador sumador inversor

Para realizar el análisis de esta célula se utilizará el método de superposición y el concepto de cortocircuito virtual

Si $V_1 = 0 \Rightarrow V_o = -\frac{Z_1}{Z_3} V_2$

Si $V_2 = 0 \Rightarrow V_o = -\frac{Z_1}{Z_2} V_1$

Entonces

$$V_o = -\frac{Z_1}{Z_3} V_2 - \frac{Z_1}{Z_2} V_1 \qquad \text{Ecuación 2.1.}$$

En la siguiente figura se presenta la estructura de un sumador inversor con filtro de banda pasante

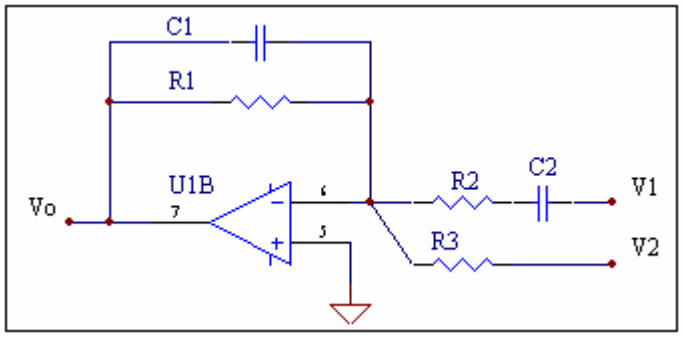

Ilustración 2.6: estructura amplificador inversor sumador banda pasante

Identificando con la Ilustración 2.5 se obtienen

$$Z_1 = \frac{1}{C_1 \cdot s} // R_1 = \frac{\frac{R_1}{C_1 \cdot s}}{\frac{1}{C_1 \cdot s} + R_1} = \frac{R_1}{1 + R_1 \cdot C_1 \cdot s}$$

$$Z_2 = \frac{1}{C_2 \cdot s} + R_2 = \frac{1 + R_2 \cdot C_2 \cdot s}{C_2 \cdot s}$$

$$Z_3 = R_3$$

Sustituyendo en la ecuación 2.1.

$$V_o = -\frac{\frac{R_1}{1+R_1\cdot C_1\cdot s}}{R_3}V_2 - \frac{\frac{R_1}{1+R_1\cdot C_1\cdot s}}{\frac{1+R_2\cdot C_2\cdot s}{C_2\cdot s}}V_1 \Rightarrow$$

$$V_o = -\frac{R_1}{R_3}\frac{1}{1+R_1\cdot C_1\cdot s}V_2 - \frac{R_1}{R_2}\frac{1}{1+R_1\cdot C_1\cdot s}\frac{R_2\cdot C_2\cdot s}{1+R_2\cdot C_2\cdot s}V_1$$

Entonces se puede identificar dos funciones de transferencia, la primera

$$H_1(S) = \frac{V_o(S)}{V_2(S)} = -\underbrace{\frac{R_1}{R_3}}_{Ganancia}\underbrace{\frac{1}{1+R_1\cdot C_1\cdot s_1}}_{Paso\_bajo}$$

Donde

$$\omega_{c1} = {}^{1}\!\!\diagup\!\!_{R_1\cdot C_1}$$

$$\left|H_1(j\cdot\omega_0)\right| = {}^{R_1}\!\!\diagup\!\!_{R_3}$$

y la segunda

$$H_2(S) = \frac{V_0(S)}{V_1(S)} = -\underbrace{\frac{R_1}{R_2}}_{Ganancia}\underbrace{\frac{1}{1+R_1\cdot C_1\cdot s}}_{Paso\_bajo}\underbrace{\frac{R_2\cdot C_2\cdot s}{1+R_2\cdot C_2\cdot s}}_{Paso\_alto}$$

En el caso de polos muy separados $R_2\cdot C_2 > 10\cdot R_1\cdot C_1$, las frecuencias de corte y la ganancia en la banda central vienen determinadas aproximadamente, por

$$\omega_{c2} = {}^{1}\!\!\diagup\!\!_{R_1\cdot C_1}$$

$$\omega_{c3} = {}^{1}\!\!\diagup\!\!_{R_2\cdot C_2}$$

$$\left|H_2(j\cdot\omega_0)\right| = {}^{R_1}\!\!\diagup\!\!_{R_2}$$

Seguidamente se presenta el diagrama de Bode asintótico

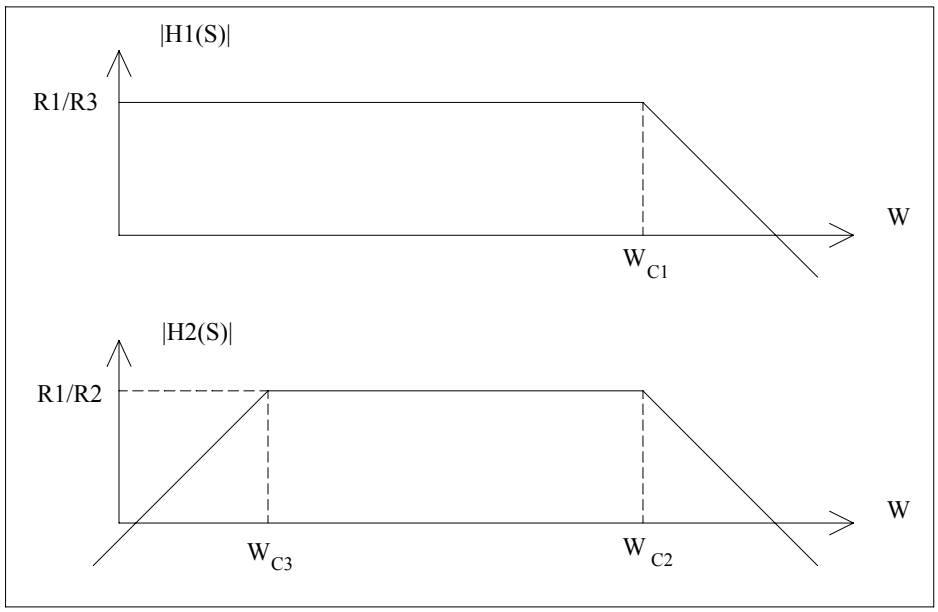

Ilustración 2.7: diagrama de Bode

La señal que se pretende medir proviene de V_1, por lo tanto la función de transferencia $H_2(S)$ será la que modifique ésta. La pulsación de corte ω_{C2} viene determinada por la frecuencia de Niquist para evitar el efecto de solapamiento o *aliasing*. Esta frecuencia es igual a la mitad de la frecuencia máxima de muestreo soportada por el conversor analógico/digital.

El periodo de ciclo de conversión en modo RD del conversor A/D es

$$T_{Smin} = 2.5\mu s \Rightarrow f_{Smax} = 400KHz$$

$$f_{Niquist} = \frac{f_{Smax}}{2} = \frac{400KHz}{2} = 200KHz$$

Esta sería la frecuencia de Niquist si el conversor A/D trabajase al máximo de sus posibilidades, pero en el caso que nos ocupa, es el microcontrolador el que fija la velocidad máxima de muestreo a $19\mu s$ (este valor está escrito en la memoria de calculo, apartado software de bajo nivel).

$$T_{Smin} = 19\mu s \Rightarrow f_{Smax} \cong 53KHz$$

$$f_{Niquist} = \frac{f_{Smax}}{2} = \frac{53KHz}{2} = 26.5KHz$$

Entonces

$$2 \cdot \pi \cdot 26.5KHz = \omega_{C2}\, {rad}/{s}$$

La pulsación de corte ω_{C3} viene determinada por el modo de funcionamiento del osciloscopio en AC, es decir, se eliminan las bajas frecuencias. Esta frecuencia de corte es de un valor típico de $5Hz$.

$$2 \cdot \pi \cdot 5Hz = \omega_{C3}\, {rad}/{s}$$

Se desea que el filtro presente una ganancia unitaria, por lo tanto

$$\left|H_2\left(j\cdot\omega_0\right)\right| = {R_1}\Big/{R_2} = 1$$

Ahora se está en disposición de determinar la función de transferencia $H_2(S)$

$$R_1 = R_2 = 10K\Omega$$

$$\omega_{c2} = 2\cdot\pi\cdot 26.5KHz = {1}\Big/{R_1\cdot C_1} = {1}\Big/{10K\Omega\cdot C_1} \Rightarrow$$

$$C_1 = \frac{1}{2\cdot\pi\cdot 26.5KHz\cdot 10K\Omega} \cong 60pF$$

$$\omega_{c3} = 2\cdot\pi\cdot 5Hz = {1}\Big/{R_2\cdot C_2} = {1}\Big/{10K\Omega} \Rightarrow C_2 = \frac{1}{2\cdot\pi\cdot 5Hz\cdot 10K\Omega} \cong 3\mu F$$

La señal proveniente del V_2 y que en consecuencia queda alterada por la función de transferencia $H_1(S)$, es una tensión continua constante, por lo tanto al ser una señal de frecuencia cero no sufre modificación alguna por el filtro paso bajo. La ganancia deseada nuevamente es la unidad, entonces

$$R_1 = R_3 = 10K\Omega$$

Finalmente se presenta el diseño global del filtro inversor sumador con filtro de banda pasante en la siguiente figura

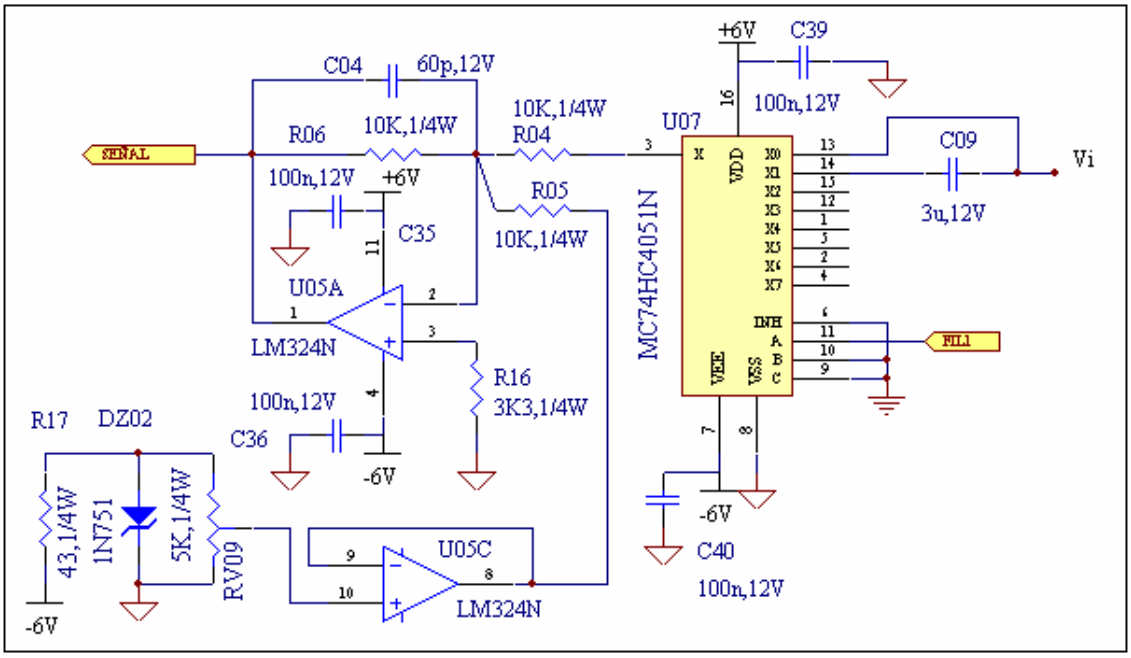

Ilustración 2.8: amplificador inversor sumador banda pasante o paso bajo

La función del multiplexor analógico U07, tiene la función de conmutar entre el modo de funcionamiento del osciloscopio de AC a DC y viceversa. Es controlado por el puerto FIL1 procedente del lach U19.

La tensión de salida del amplificado operacional U05C, en configuración de seguidor de tensión, se ha insertado para evitar efectos de carga con R_5. Esta tensión debe ser de -2.5V, ya que el conversor A/D que está conectado directamente con el puerto SEÑAL, trabaja en un rango de $[0,5]$V.

Para conseguir una tensión de referencia estable se ha implementado un regulador de tensión con un diodo zener de 5.1V. Por éste deben circular 20mA para mantener entre sus bornes una tensión estable, entonces

$$R_{17} = \frac{-5.1+6}{20\cdot 10^{-3}} = 45\Omega \Rightarrow \text{ Valor comercial } R_{17} = 43\Omega$$

De igual modo que el apartado anterior se ha puesto una resistencia en el terminal positivo del U05A para compensar la caída interna de tensión debida a las corrientes de offset, de valor

$$R_{16} = R_4 \,//\, R_5 \,//\, R_6 = \frac{R_4 \cdot R_5 \cdot R_6}{R_4 \cdot R_5 + R_5 \cdot R_6 + R_4 \cdot R_6} \text{ Como}$$

$$R_4 = R_5 = R_6 \Rightarrow R_{16} = \frac{R_4}{3} = \frac{10K\Omega}{3} \cong 3K3\Omega$$

2.1.3 CIRCUITERÍA DE ALIMENTACIÓN

2.1.3.1 Estudio preliminar

2.1.3.1.1 Rectificación y filtrado

Un rectificador de onda completa transfiere energía de la entrada a la salida durante todo el ciclo de la señal y proporciona mayor corriente promedio por cada ciclo en relación con la que se obtiene utilizando un rectificador de media onda.

El promedio de una función periódica se define como la integral de la función sobre un periodo dividida por el periodo. Es igual al primer término del desarrollo de la función en series de Fourier. El rectificador de onda completa produce el doble de corriente promedio en relación con el rectificador de media onda.

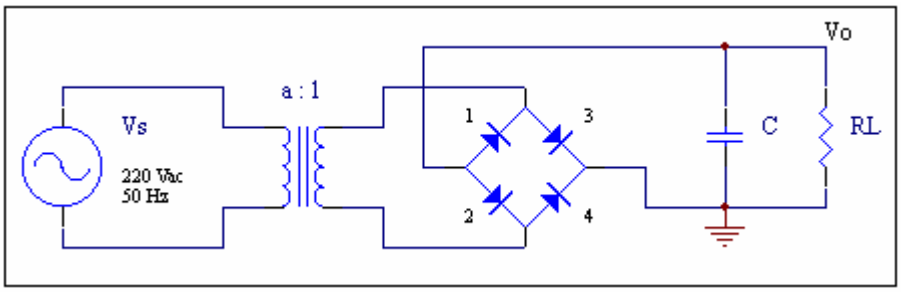

Ilustración 2.9: rectificación y filtrado

El puente rectificador de la ilustración 2.9 realiza la rectificación de onda completa. Cuando la fuente de tensión es positiva, los diodos 1 y 4 conducen y los diodos 2 y 3 son circuitos abiertos. Cuando la fuente de tensión se vuelve negativa, se invierte la situación y los diodos 2 y 3 conducen. Se ha añadido un transformador entre la fuente y el puente de diodos para aislar entre sí las dos tierras y obtener una tensión moderada en Vo.

Se ha añadido un condensador en paralelo con el resistor de carga para realizar el filtrado de la tensión del rectificador. En la siguiente ilustración se representa esta situación.

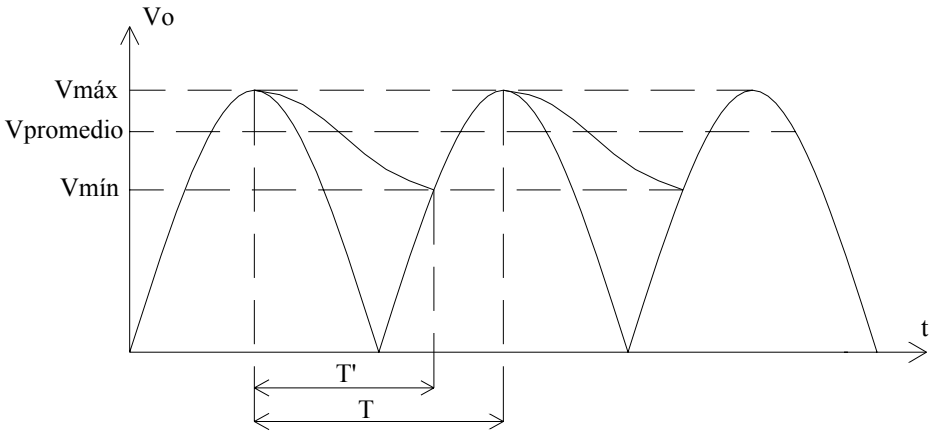

Ilustración 2.10: forma de onda real del filtrado

El condensador se carga al valor de tensión más alto $(Vmax)$ cuando la entrada alcanza su máximo valor positivo o negativo. Cuando la tensión de entrada cae por debajo de este valor, el condensador no se puede descargar a través de ninguno de los

diodos. Por tanto, la descarga se lleva a cabo a través de R_L . Esto produce un decaimiento exponencial dado por la ecuación

$$Vo = Vmax \cdot e^{-t/\tau} = Vmax \cdot e^{-t/R_l \cdot C}$$ Ecuación 2.2.

Como se indica en la ilustración 2.10, es el tiempo disponible de descarga. Se puede resolver C en términos de T' y R_L tomando el logaritmo natural de ambos lados de la ecuación 1x.x:

$$\ln\left(\frac{Vmax}{Vo}\right) = \frac{T'}{R_L \cdot C}$$ Ecuación 2.3.

Esta fórmula es difícil de utilizar en el diseño del filtro, ya que T' depende de la constante de tiempo $R_L \cdot C$ y, por tanto, de la incógnita C.

Se puede aproximar el valor del filtro capacitor necesario para una carga particular utilizando una aproximación de línea recta, como se muestra en la siguiente figura. La pendiente inicial de la exponencial en la ecuación 2.2. es

$$m_1 = \frac{-Vmax}{R_L \cdot C}$$

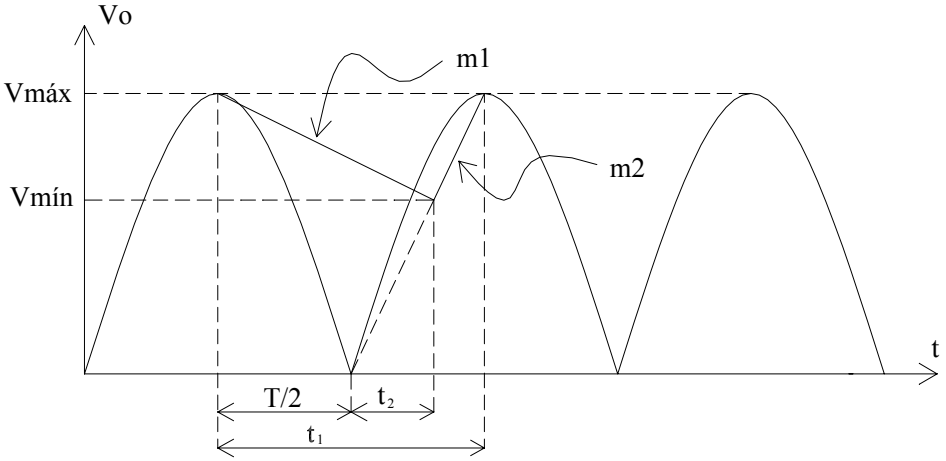

Ilustración 2.11: forma de onda aproximada del filtrado

La pendiente de la segunda recta es

$$m_2 = \frac{Vmax}{T/2}$$

Entonces

$$t_1 = \frac{-(Vmax - Vmin)}{m_1} = \frac{-\Delta V}{m_1} = \frac{R_L \cdot C \cdot \Delta V}{Vmax}$$

Por triángulos semejantes, se encuentra

$$t_1 = \frac{T}{2} + t_2 = \frac{T}{2} + \frac{T \cdot Vmin}{2 \cdot Vmax}$$

y

$$t_1 = \frac{R_L \cdot C \cdot \Delta V}{Vmax} = \frac{T\left(1 + Vmin/Vmax\right)}{2} = \frac{T}{2}\left(1 + \frac{Vmax - \Delta V}{Vmax}\right) = \frac{T\left(2 - \Delta V/Vmax\right)}{2}$$

Sustituyendo $T = 1/f_p$, donde f_p es el número de pulsos por segundo (el doble de la frecuencia original), se obtiene

$$R_L \cdot C \frac{\Delta V}{Vmax} = \frac{1}{2 \cdot f_p}\left(2 - \frac{\Delta V}{Vmax}\right) = \frac{1}{f_p}\left(1 - \frac{\Delta V}{2 \cdot Vmax}\right) \Rightarrow$$

$$2 \cdot \pi \cdot f_p \cdot R_L \cdot C = \frac{2 \cdot \pi \cdot Vmax}{\Delta V}\left(1 - \frac{\Delta V}{2 \cdot Vmax}\right)$$

pero como

$$\frac{\Delta V}{2 \cdot Vmax} << 1$$

se desprecia el segundo término para obtener

$$2 \cdot \pi \cdot f_p \cdot R_L \cdot C = \frac{2 \cdot \pi \cdot Vmax}{\Delta V}$$

o

$$C = \frac{2 \cdot \pi \cdot Vmax}{\Delta V \cdot 2 \cdot \pi \cdot f_p \cdot R_L}$$

Esta fórmula representa una solución conservativa del problema de diseño: si la línea recta nunca pasa por debajo de *Vmin*, la curva exponencial estará seguro por encima de este valor. Una regla práctica que se utilizará en el diseño es elegir

$$C = \frac{5 \cdot Vmax}{\Delta V \cdot 2 \cdot \pi \cdot f_p \cdot R_L} \qquad \text{Ecuación 2.4.}$$

2.1.3.1.2 Diseño de una fuente de tensión usando un circuito integrado

Los reguladores de tensión lineales se empaquetan como circuitos integrados (CI); se enfocará la atención sobre la serie L78XX. Las hojas de especificaciones apropiadas aparecen en el apéndice. Se puede obtener varias tensiones diferentes: 5, 5.2, 6, 8, 8.5, 9, 12, 15, 18 y 24 V. Todo lo que se requiere para diseñar un regulador alrededor de uno de estos CI es seleccionar el transformador, los diodos y el filtro.

Las hojas de especificaciones para este CI indican que debe existir una tierra común entre la entrada y la salida, y que la tensión mínima en la entrada del CI debe estar al menos 2 ó 4 V por encima de la salida regulada. Para asegurar esta última condición, es necesario filtrar la salida del rectificador. En la ilustración 2.12, C_F realiza este filtrado cuando se combina con la resistencia de entrada del CI.

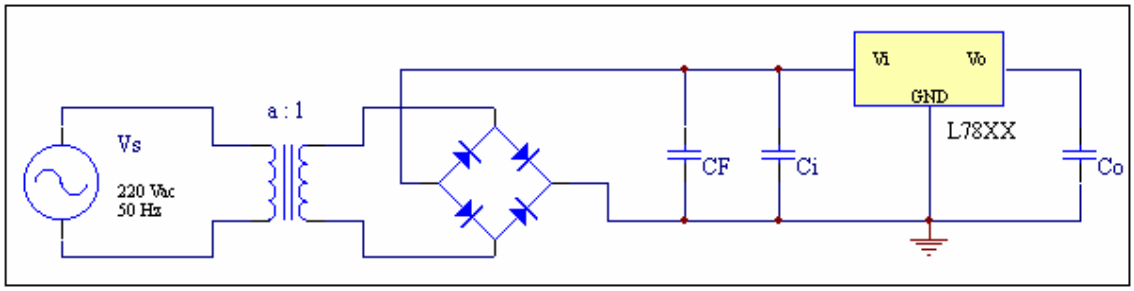

Ilustración 2.12: fuente de tensión con un circuito integrado

La resistencia de entrada equivalente más pequeña del CI está dada por $V_{\mathrm{Im}in} \big/ I_{Omax}$. Entonces

$$C_F = \frac{5\left(V_{\mathrm{Im}ax} - V_O\right)}{\Delta V \cdot 2 \cdot \pi \cdot f_p \cdot V_{\mathrm{Im}in} \big/ I_{Omax}} \qquad \text{Ecuación 2.5.}$$

donde V_{smax} es tensión más grande que se aplica al CI, ΔV es la caída de tensión del condensador (es decir, la tensión de pico más pequeña aplicada al CI menos la tensión de salida del CI más 4 V) y f_p es el número de pulsos por segundo.

El condensador de salida, C_o, se añade para cortar las variaciones de alta frecuencia provenientes de la circuitería de carga. C_i es necesario si el condensador de filtrado C_F se encuentra lejos de éste primero.

Los reguladores de la serie L79XX son idéntico a los de la serie L78XX, con la excepción de que los primeros proporcionan tensiones reguladas negativas en vez de positivas.

2.1.3.2 Implementación de la administración de energía

La gestión de la energía consta de dos grandes bloques:

a) Digital: Administración de una tensión de alimentación de 5 voltios a la circuitería digital y una tensión aproximada de 10 voltios para los relés.
b) Analógico: Proporciona una tensión positiva de 6 voltios y una negativa de -6 voltios para la alimentación de la circuitería analógica.

Las masas de ambos bloques permanecerán separadas para evitar interferencias entre si, pero se cortocircuitarán en un solo punto para que posean un mismo potencial.

2.1.3.2.1 Fuente de tensión de +5 voltios

Para implementar la alimentación digital se a utilizado un UA7805C que es equivalente al L7805. Se ha sobredimensionado la cantidad de corriente que debería pasar por el regulador calculando el sumatorio de las corrientes máximas de la circuitería digital, a continuación se presenta una tabla:

COMPONENTE	CORRIENTE	CANTIDAD	CORRIENTE PARCIAL
87C51	75 mA	1 unidad	75 mA
UM6264AB	150 mA	1 unidad	150 mA
MC74HC573AN	75 mA	2 unidades	150 mA
MC74HC02A	50 mA	1 unidad	50 mA
ADC0820CCN	15 mA	1 unidad	15 mA
MAX232N	10 mA	1 unidad	10 mA
RY610012	23 mA	4 unidades	92 mA
LED	12 mA	1 unidad	12 mA
Resistencia R01	3 mA	1 unidad	3 mA
Resistencias R03, R28 y R29	2.5 mA	3 unidades	7.5 mA
CORRIENTE TOTAL			564.5 mA

Cuadro 2.2: corriente digital

NOTA: Los valores de las corrientes máximas se han obtenido de las hojas de especificaciones incluidas en el apéndice. Las corrientes de las resistencias se han calculado mediante la expresión $I = {5^2}/{Rx}$.

Los valores necesarios para el cálculo del condensador del filtro son:

$V_O = 5V$

$V_{\mathrm{Im}in} = V_O + 4 = 5 + 4 = 9V$

$V_{\mathrm{Im}ax} = 10V$

$\Delta V = V_{\mathrm{Im}ax} - V_{\mathrm{Im}in} = 10 - 9 = 1V$

$I_{Omax} = 565mA = 0.565A$

$f_p = 100Hz$

Entonces sustituyendo en la ecuación 2.5. se obtiene

$$C_F = \frac{5(V_{\mathrm{Im}ax} - V_O)}{\Delta V \cdot 2 \cdot \pi \cdot f_p \,.\, {V_{\mathrm{Im}in}}/{I_{Omax}}} = \frac{5(10-5)}{1 \cdot 2 \cdot \pi \cdot 100 \cdot {9}/{0.565}} \cong 2.5mF$$

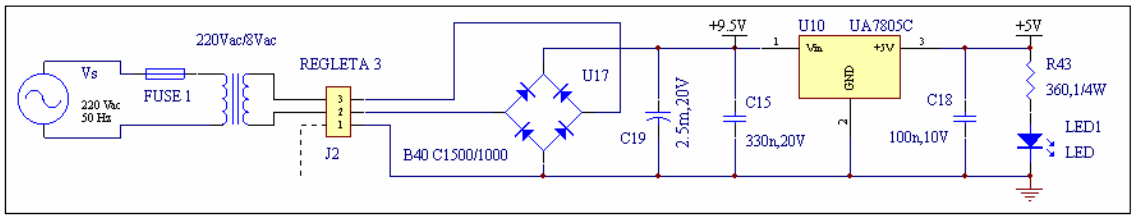

Ilustración 2.13: fuente de tensión de +5 voltios

Para determinar la tensión en el secundario del transformador se ha seguido el siguiente procedimiento:

$$V_{2^{o}trafo} = \frac{V_{\mathrm{Im}ax} + 1.2}{\sqrt{2}} \cong 8V_{ac}$$

donde el $1.2V$ proviene de la caída de tensión en directo de dos de los diodos del puente rectificados y la $\sqrt{2}$ para convertir los voltios de tensión continua a tensión alterna.

La obtención de la potencia máxima que disipa el transformador se determina de forma empírica, es decir, se utiliza una fuente de tensión de laboratorio para determinar la corriente que ésta otorga y así poder escoger un transformador que se adecue lo más posible a la corriente demandada. De igual modo, la corriente a la cual el fusible abre el circuito también se determina de forma práctica.

$$I_{2^{o}trafo} = ? \Rightarrow P_{trafo} = I_{2^{o}trafo} \cdot V_{2^{o}trafo} = ? \cdot 8 = ? W$$

La relación de transformación del transformador es

$$n = \frac{V_{2^{o}trafo}}{V_{1^{o}trafo}} = \frac{8}{220} \cong 6.63 \cdot 10^{-2}$$

Entonces

$$I_{fusible} = n \cdot I_{2^{o}trafo} = 6.63 \cdot 10^{-2} \cdot ? = ? A$$

El pin 3 de la regleta, que corresponde a la masa digital, se cortocircuitará con la masa analógica para que ambas posean un mismo potencial. El puente de diodos U17 soporta con creces la corriente que le atraviesa. Las tensiones máximas de los condensadores se han sobredimensionado al doble de la tensión máxima teórica.

La tensión de alimentación de los relés no se otorga de la salida del regulador, sino de se entrada. Se ha optado por esta opción para no sobrecargar el regulador y se aprovecha de esta manera la tolerancia de los relés al rizado de la alimentación.

Para determinar la resistencia R_{23} se ha seguido el siguiente procedimiento:

$$R_{23} = \frac{5 - 0.6}{12 \cdot 10^{-3}} \cong 366.6\Omega \Rightarrow \text{valor comercial } R_{23} = 360\Omega$$

Los valore C_{15} y C_{18} son los recomendados por el fabricante del CI.

2.1.3.2.2 Fuente de tensión de +6 y -6 voltios

Se realizará un proceso de diseño homólogo al anterior con la salvedad de que el condensador de filtrado obtenido será de un valor mitad a los condensadores C_{11} y C_{12}. Véase la ilustración 2.14.

COMPONENTE	CORRIENTE	CANTIDAD	CORRIENTE PARCIAL
LM324N	50 mA	2 unidades	100 mA
MC74HC4051N	62.5 mA	3 unidades	187.5 mA
LED	12 mA	2 unidades	24 mA
1N751	20 mA	2 unidades	40 mA
CORRIENTE TOTAL			351.5 mA

Cuadro 2.3: corriente digital

Los valores necesarios para el cálculo del condensador del filtro son:

$V_O = 2 \cdot 6 = 12V$

$V_{\mathrm{Im}in} = V_O + 2 \cdot 4 = 12 + 8 = 20V$

$V_{\mathrm{Im}ax} = 28V$

$\Delta V = V_{\mathrm{Im}ax} - V_{\mathrm{Im}in} = 28 - 20 = 8V$

$I_{Omax} = 352mA = 0.352A$

$f_p = 100Hz$

Entonces sustituyendo en la ecuación 2.5. se obtiene

$$C_F = \frac{5\cdot\left(V_{\mathrm{Im}ax} - V_O\right)}{\Delta V \cdot 2 \cdot \pi \cdot f_p \cdot {V_{\mathrm{Im}in}}/{I_{Omax}}} = \frac{5\cdot(28-12)}{8\cdot 2\cdot\pi\cdot 100\cdot {20}/{0.352}} \cong 280\mu F$$

$$C_{11} = C_{12} = 2\cdot C_F = 560\mu F \Rightarrow \text{valor comercial } C_{11} = C_{12} = 680\mu F$$

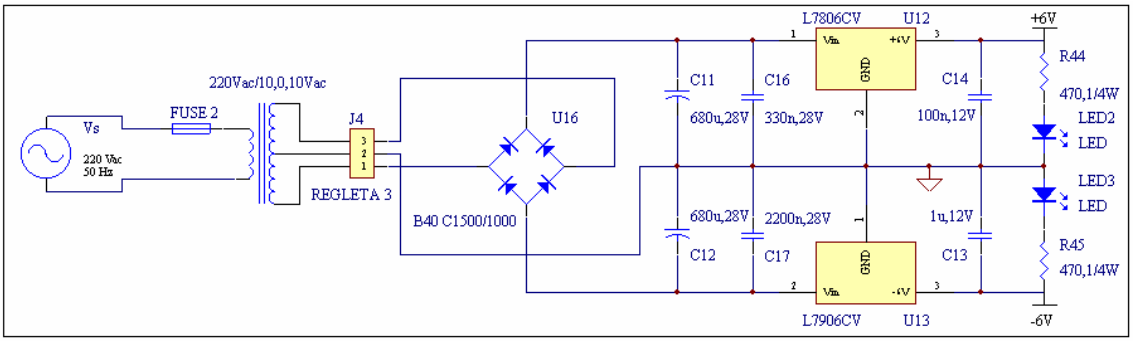

Ilustración 2.14: fuente de tensión de +6 y -6 voltios

La tensión en el secundario del transformador es

$$V_{2°trafo} = \frac{V_{\mathrm{Im}ax} + 1.2}{\sqrt{2}} \cong 20V_{ac}$$

$$I_{2°trafo} = ? \Rightarrow P_{trafo} = I_{2°trafo}\cdot V_{2°trafo} = ?\cdot 20 = ?W$$

$$n = \frac{V_{2°trafo}}{V_{1°trafo}} = \frac{20}{220} \cong 9.09\cdot 10^{-2}$$

Entonces

$$I_{fusible} = n\cdot I_{2°trafo} = 9.09\cdot 10^{-2}\cdot ? = ? A$$

Las resistencias en serie con los diodos:

$$R_{44} = R_{45} = \frac{6-0.6}{12\cdot 10^{-3}} = 450\Omega \Rightarrow \text{valor comercial } R_{23} = 470\Omega$$

2.2 SOFTWARE DE BAJO NIVEL

2.2.1 DIAGRAMAS DE FLUJO

En los siguientes apartados, se presentan los diagramas de flujo del código en ensamblador. En la memoria descriptiva se expone de forma narrada la explicación de éstos.

2.2.1.1 Preludio al muestreo y escritura

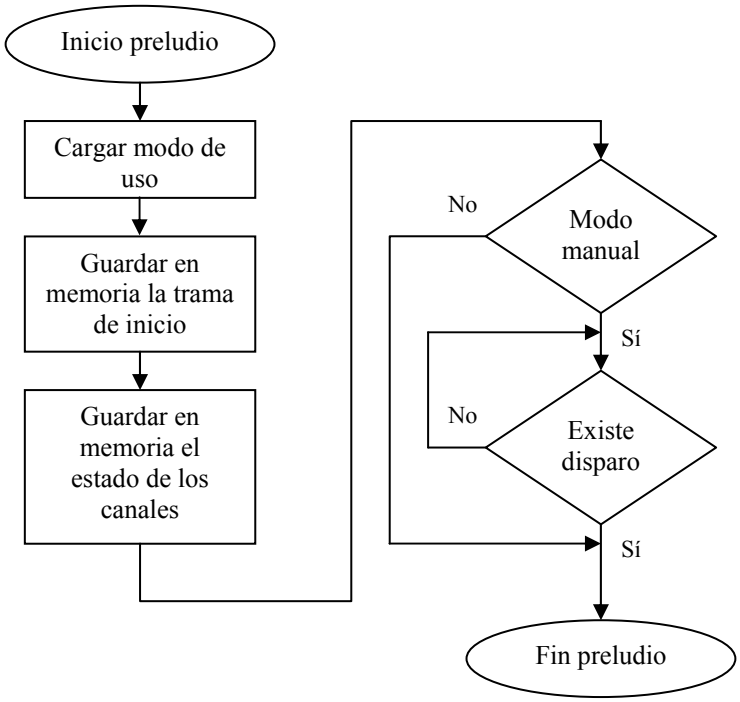

Ilustración 2.15: preludio al muestreo y escritura a memoria

2.2.1.2 Muestreo y escritura en memoria

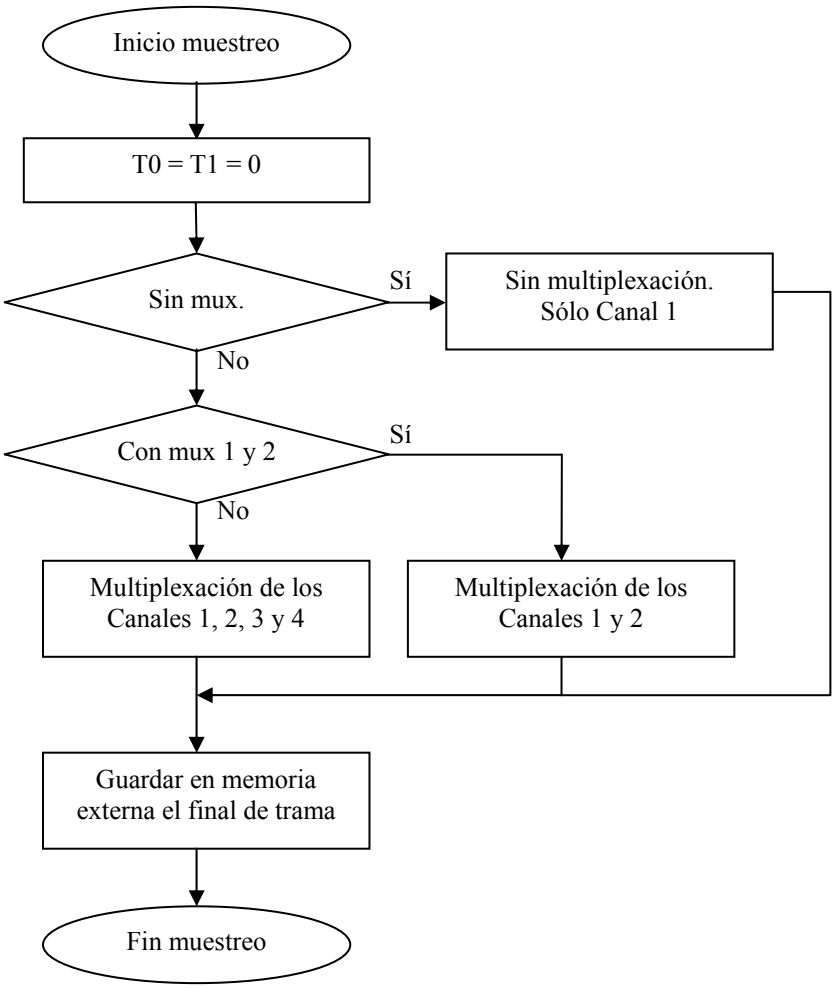

Ilustración 2.16: esquema de multiplexación

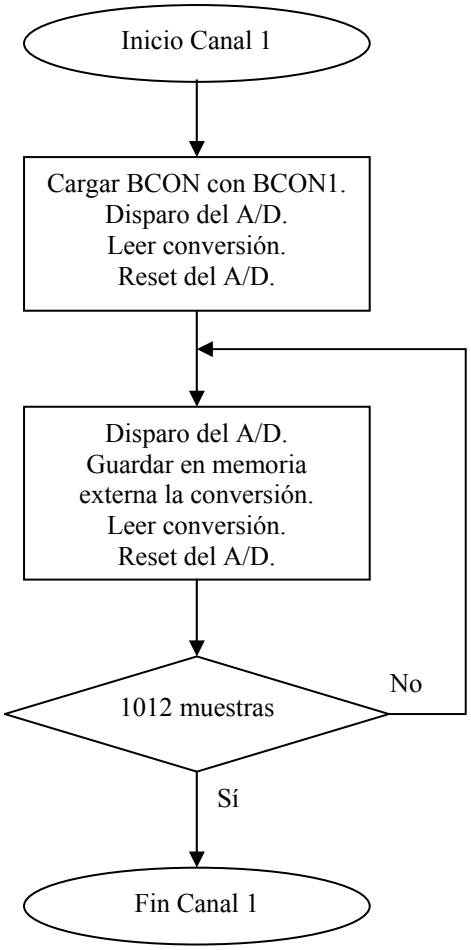

Ilustración 2.17: sin multiplexación, sólo Canal 1

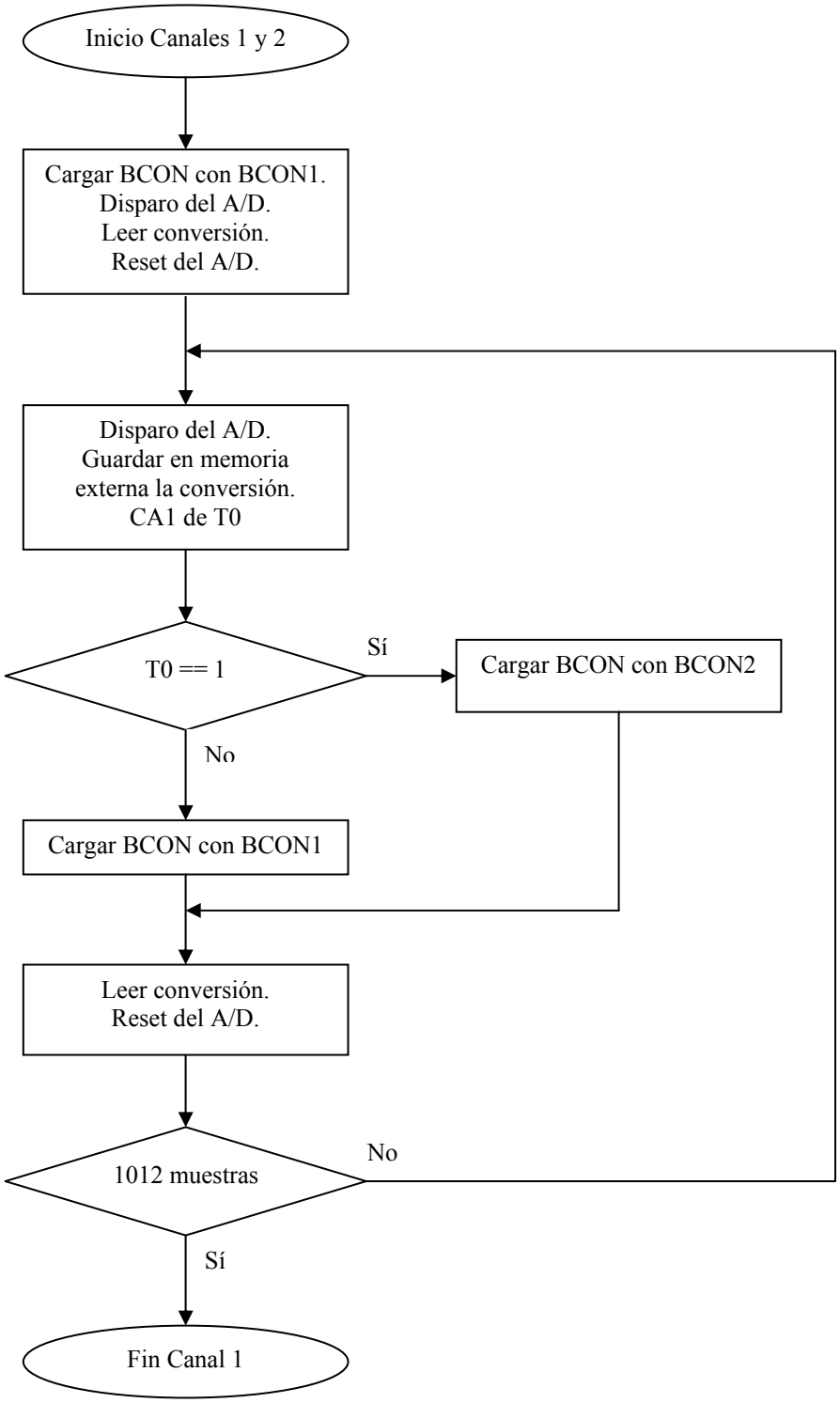

Ilustración 2.18: multiplexación de los Canales 1 y 2

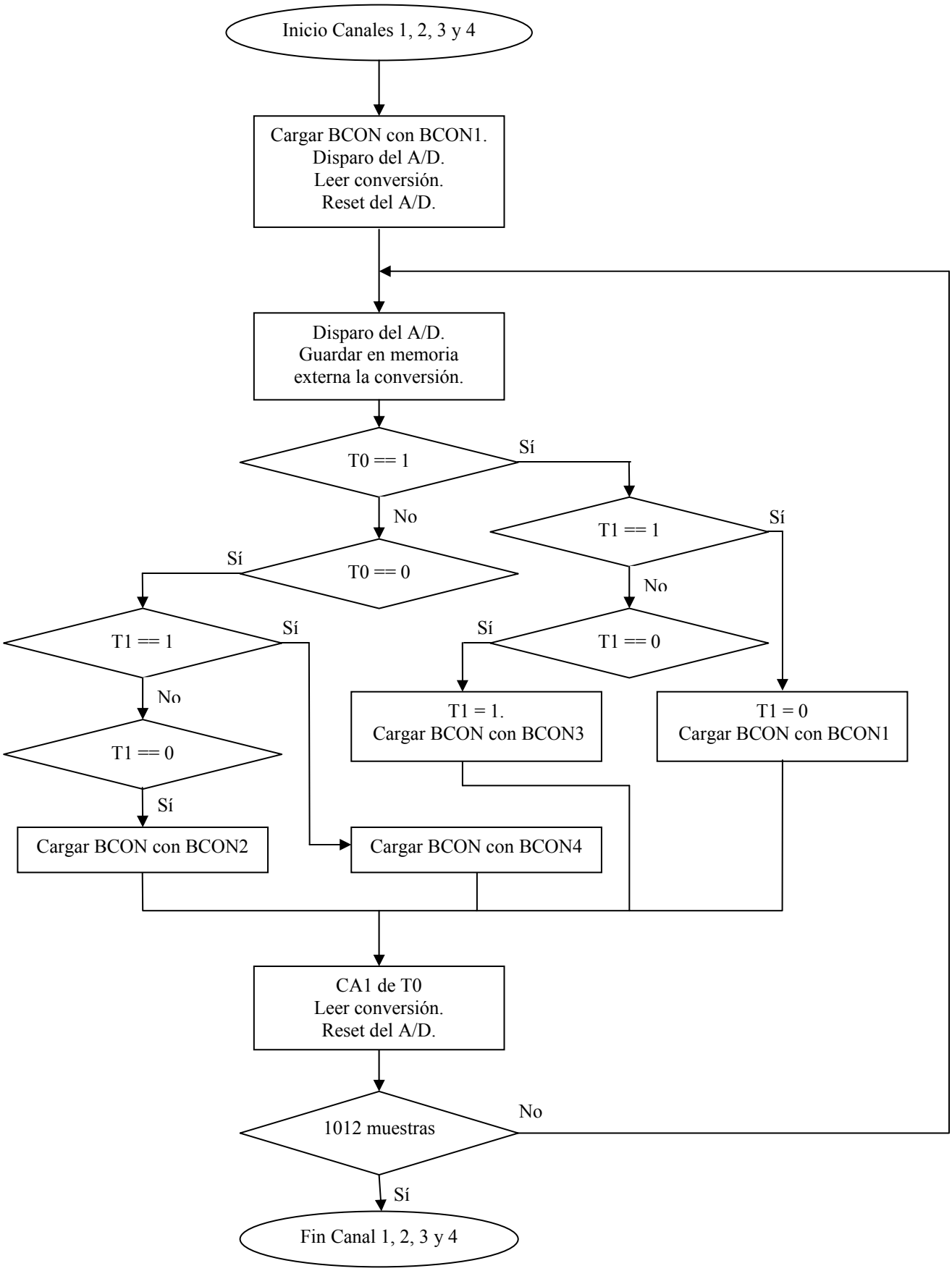

Ilustración 2.19: multiplexación de los Canales 1, 2, 3 y 4

2.2.1.3 Lectura de memoria y transmisión

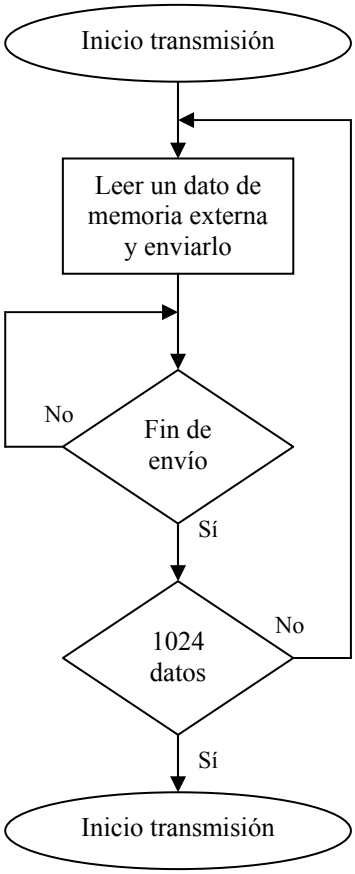

Ilustración 2.20: lectura de memoria y transmisión

2.2.1.4 Recepción

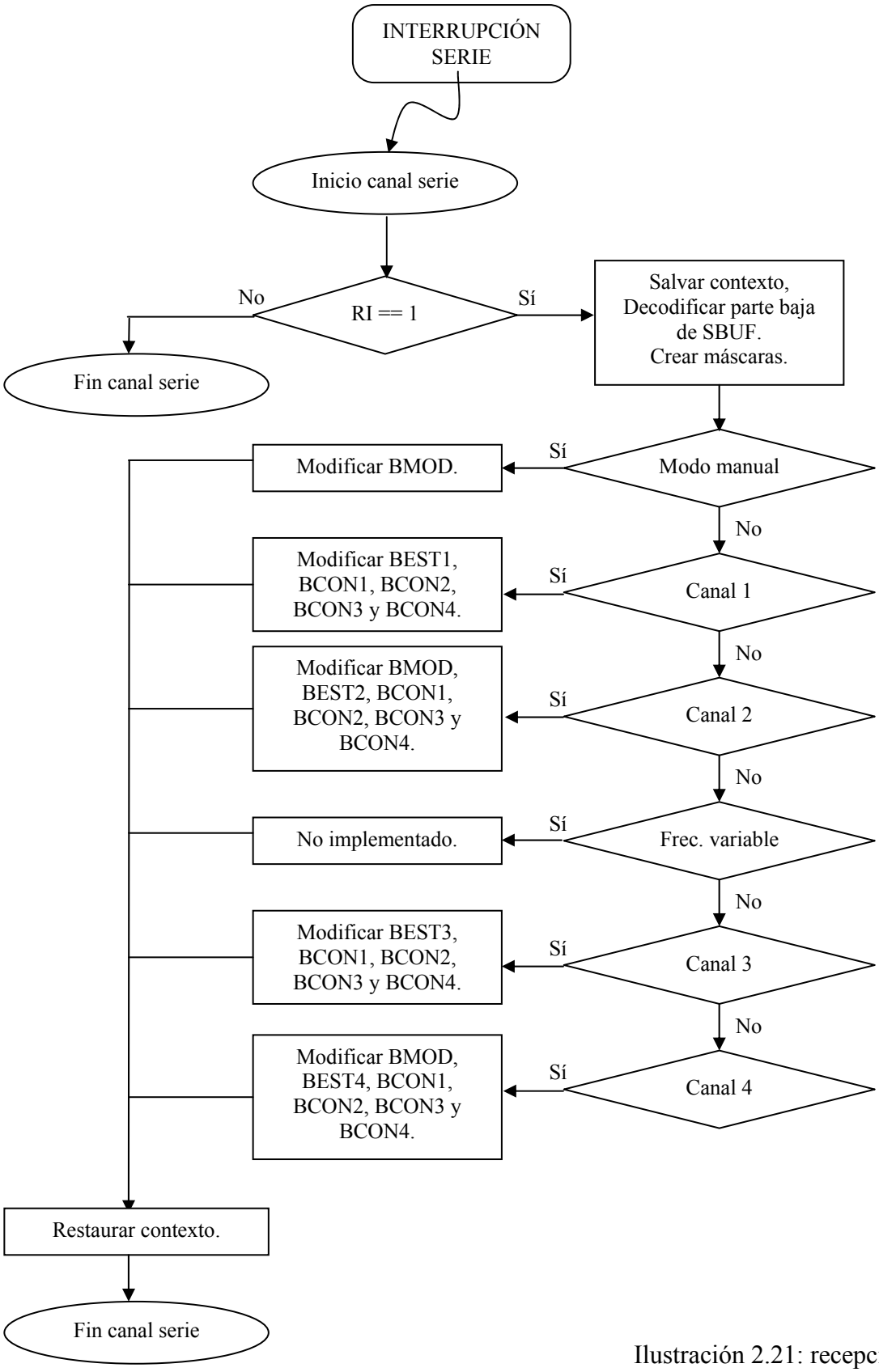

Ilustración 2.21: recepción

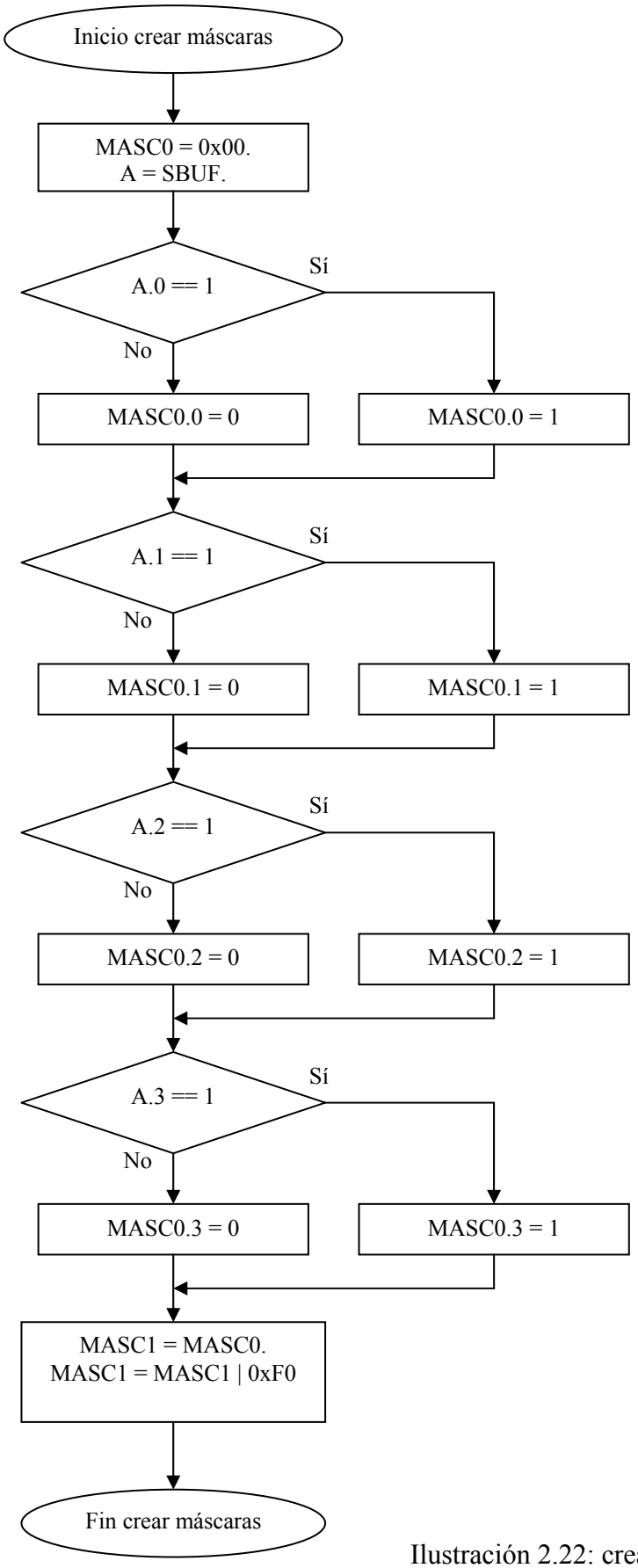

Ilustración 2.22: crear máscaras

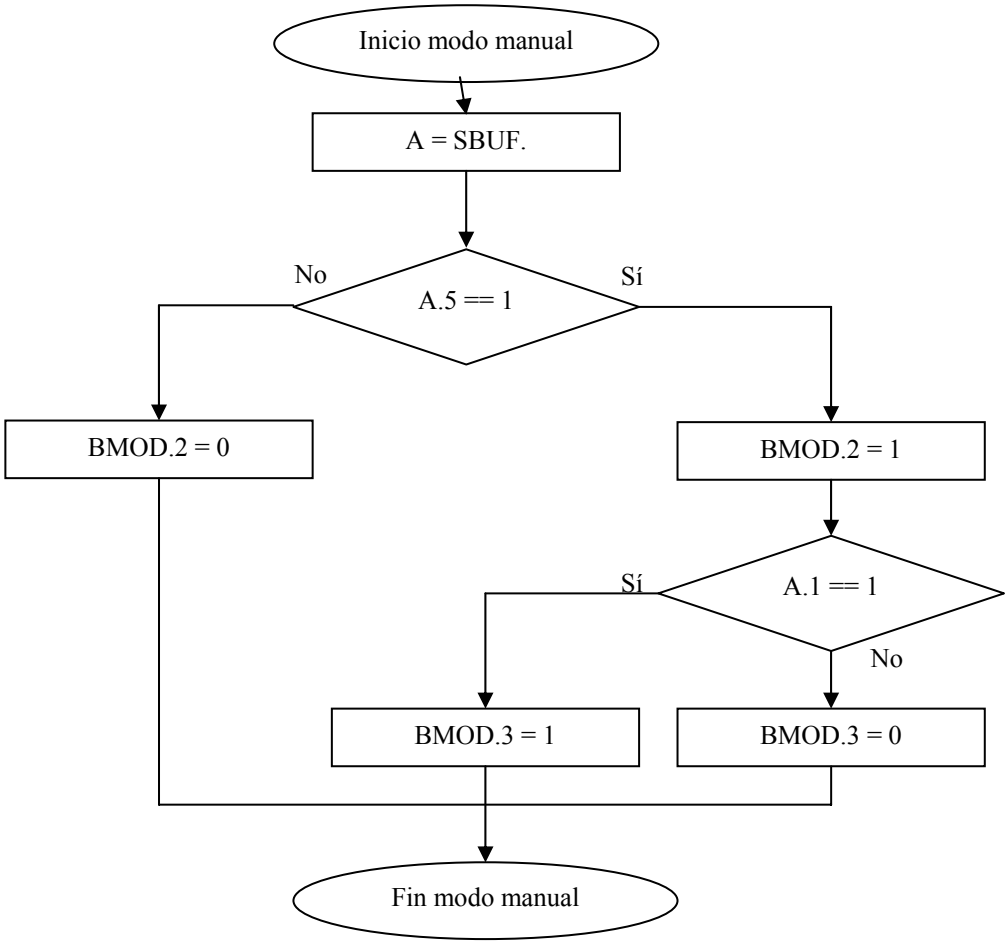

Ilustración 2.23: modo manual

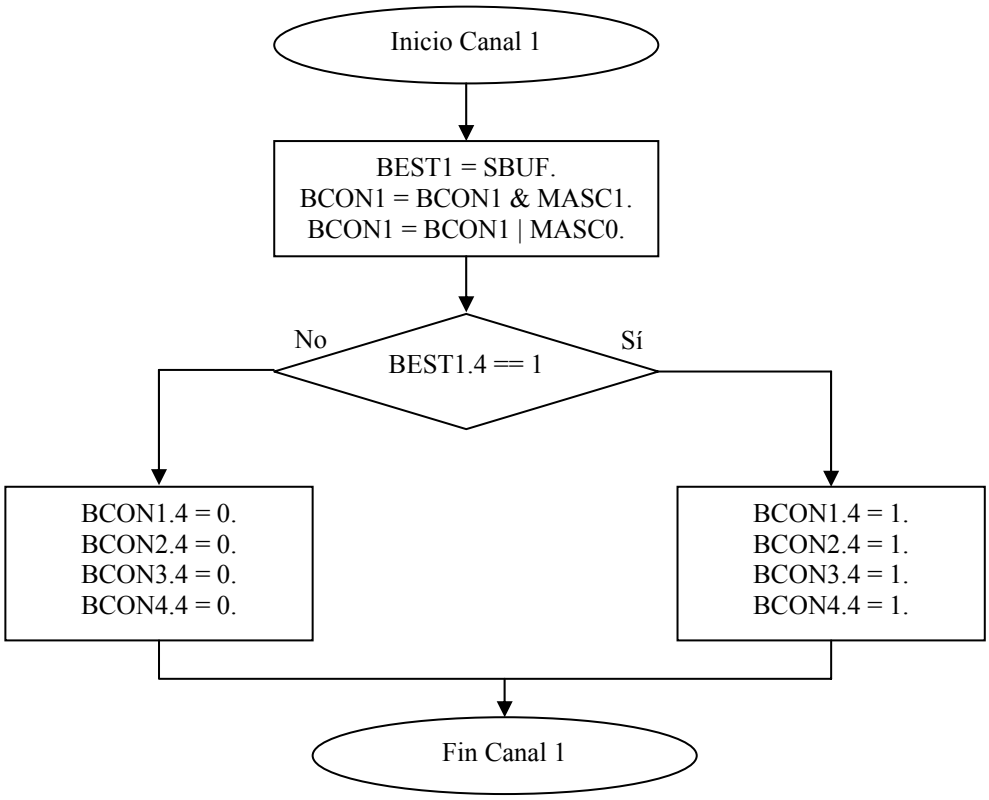

Ilustración 2.24: canal 1

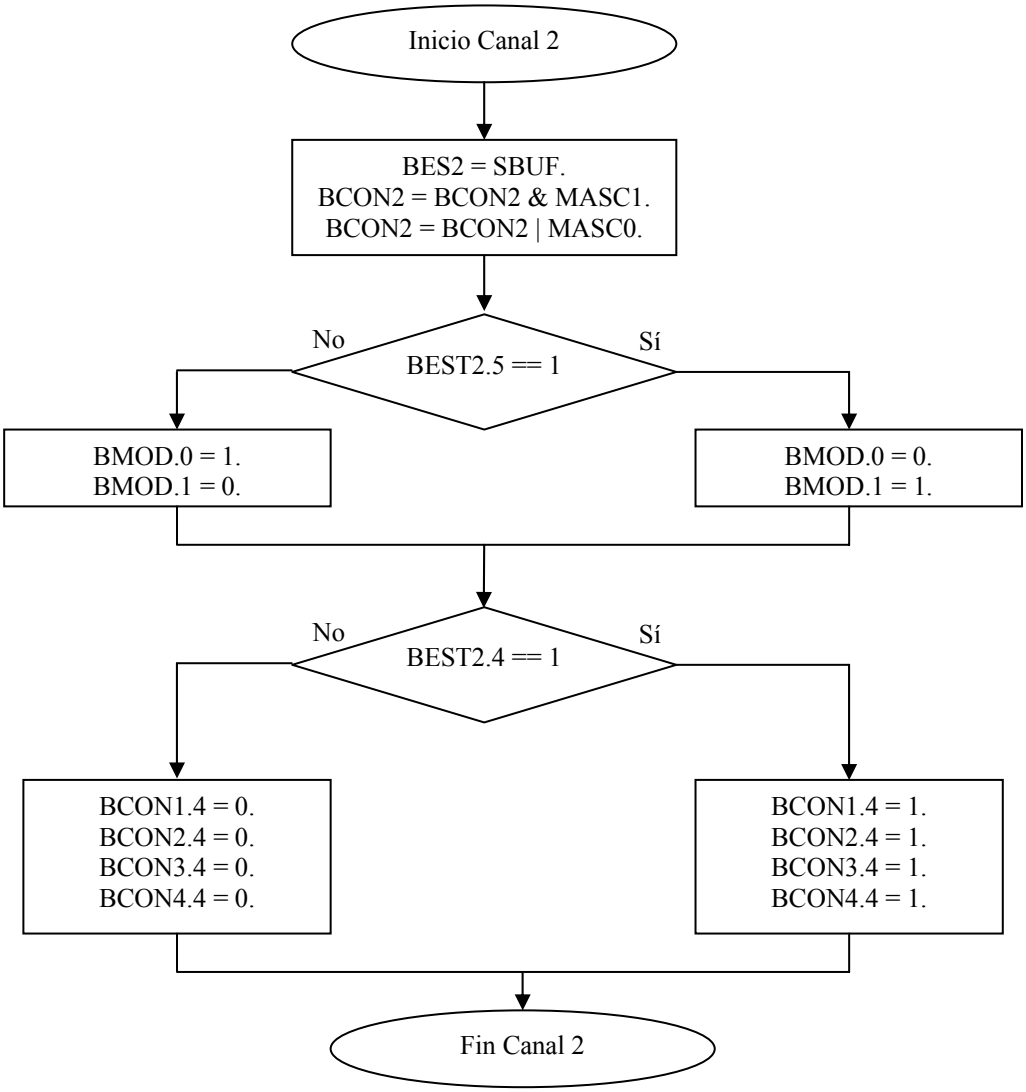

Ilustración 2.25: canal 2

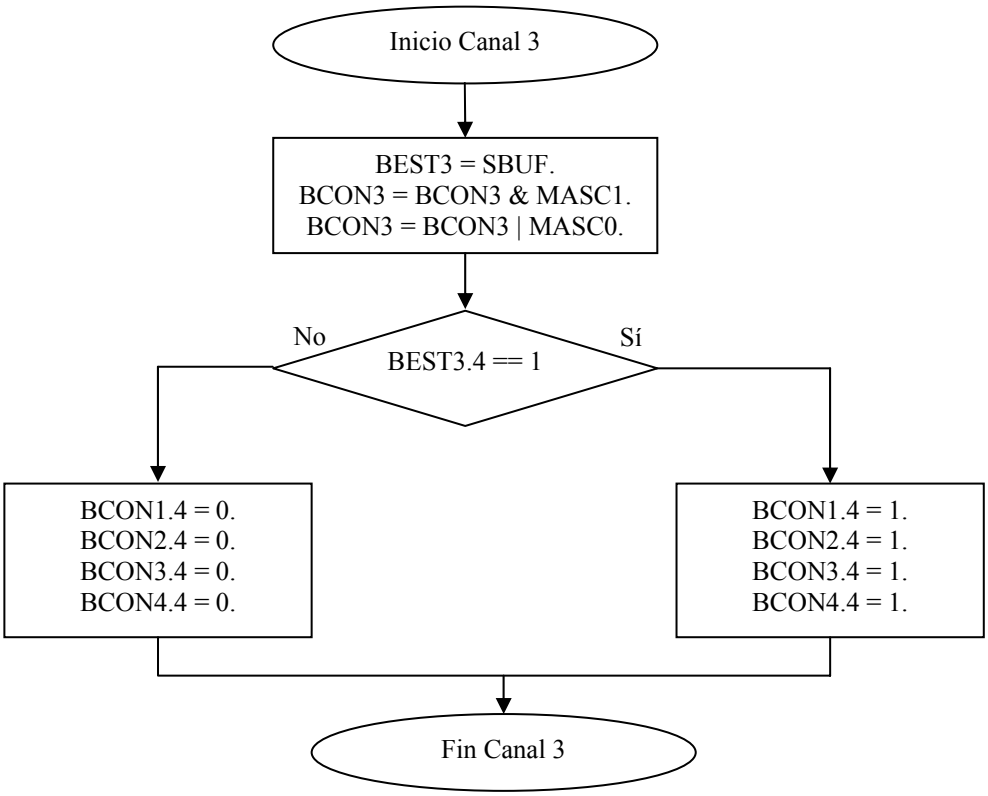

Ilustración 2.26: canal 3

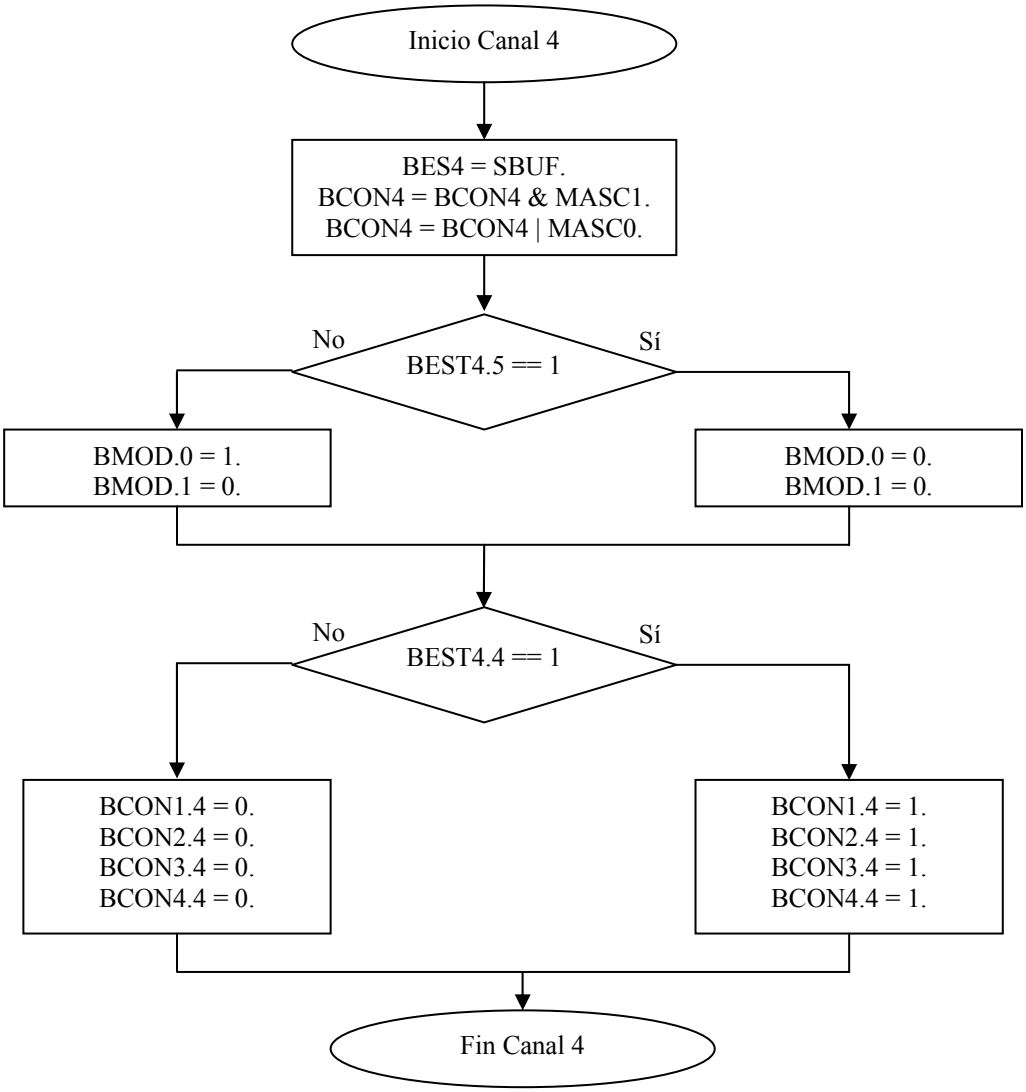

Ilustración 2.27: canal 4

2.2.2 PERIODOS DE MUESTREO

El procedimiento seguido para determinar los periodos de muestreo es el siguiente:

La frecuencia de funcionamiento del microcontrolador es de 12 MHz, y cada ciclo máquina (CM) es de 12 periodos de reloj, entonces 1 CM equivale a 1 μs .

Para determinar los periodos de muestreo de cada modo de multiplexación, sólo es necesario sumar el número de CM entre muestras, y posteriormente multiplicarlos por 1 μs .

2.2.2.1 Sin multiplexación, sólo Canal 1

Si se sitúa en el código en bajo nivel adjunto en el Anexo y observa el apartado sin multiplexación, verá que cada instrucción tiene asociada un comentario con el número de CM y Bytes asociados. Sumando todos los CM entre muestra y muestra se obtiene:

CM total = 1+2+2+1+1+2+2+2+2+2+2 = 19 CM o 19 μs .

Por lo tanto el periodo de muestreo en este modo es de 19 μs , lo que da una frecuencia de muestreo aproximada de 52631 Hz.

2.2.2.2 Multiplexación del Canal 1 y Canal 2

De forma análoga al apartado anterior:

CM total = 1+2+1+2+1+2+2+2+2+2+2+2+2+1+1+2+2+2+2+2+2 = 34 CM o 34 μs .

T muestreo = 34 μs o F muestreo = 29411 Hz

2.2.2.3 Multiplexación del Canal 1, Canal 2, Canal 3 y Canal 4

Siguiendo el mismo procedimiento:

CM total = 1+2+2+2+2+2+1+2+1+2+2+2+2+2+2+2+1+1+2+2+2+2+2+2+1 = 44 CM

T muestreo = 44 μs o F muestreo = 22727 Hz.

2.2.3 TABLA DE VELOCIDADES DE COMUNICACIÓN

Para establecer la frecuencia de comunicación en baudios, se ha programado el interface serie en Modo 1, es decir, UART de 8 bits con 1 bit de inicio, 8 bits de datos y 1 bit de final. Este modo implica también que el Timer 1 es quien establece la frecuencia de comunicación mediante su desbordamiento.

Si se programa el Timer 1 en modo 8 bits con autorrecarga, entonces la velocidad de comunicación en baudios o bits/s viene dada por la siguiente expresión:

$$Baudios = \frac{2^{SMOD}}{32} \cdot \frac{FrecuenciaOscilador}{12 \cdot (256 - TH1)}$$

Donde:

$SMOD = 1$

$FrecuenciOscilado = 12MHz$

Seguidamente se presenta un cuadro con las velocidades de comunicación más comunes. En este proyecto se ha establecido el Timer 1 de tal forma que genere 9600 baudios.

TH1	BAUDIOS
0xCC	1200
0xED	2400
0xFE	4800
0xFA	9600
0xFD	19200
0xFE	38400
0xFF	115000

Cuadro 2.2: velocidades de comunicación

2.2.4 TRAMA DE COMUNICACIÓN

La trama que construye el microcontrolador para ser enviada tiene el siguiente aspecto:

Trama de inicio				Bytes de Estado				Datos		● ● ●
0xFB	0x04	0xFB	0x04	BEST1	BEST2	BEST3	BEST4	D0	D1	

● ● ●

Datos		Trama final			
D1010	D1011	0x04	0xFB	0x04	0xFB

Ilustración 2.28: trama de comunicación

2.3 SOFTWARE DE ALTO NIVEL

El lenguaje de programación utilizado es el Microsoft Visual C++ 5.0. El programa realiza las siguientes funciones fundamentales:

- Adquisición de los datos procedentes del canal serie y presentarlos en la ventana vista
- Control de las funciones del periférico
- Monitorizado de las comunicaciones
- Programación del interface serie
- Almacenaje de los datos recibidos en memoria no volátil
- Impresión de la ventana de vista

2.3.1 DIAGRAMAS DE FLUJO

2.3.1.1 Recepción de datos

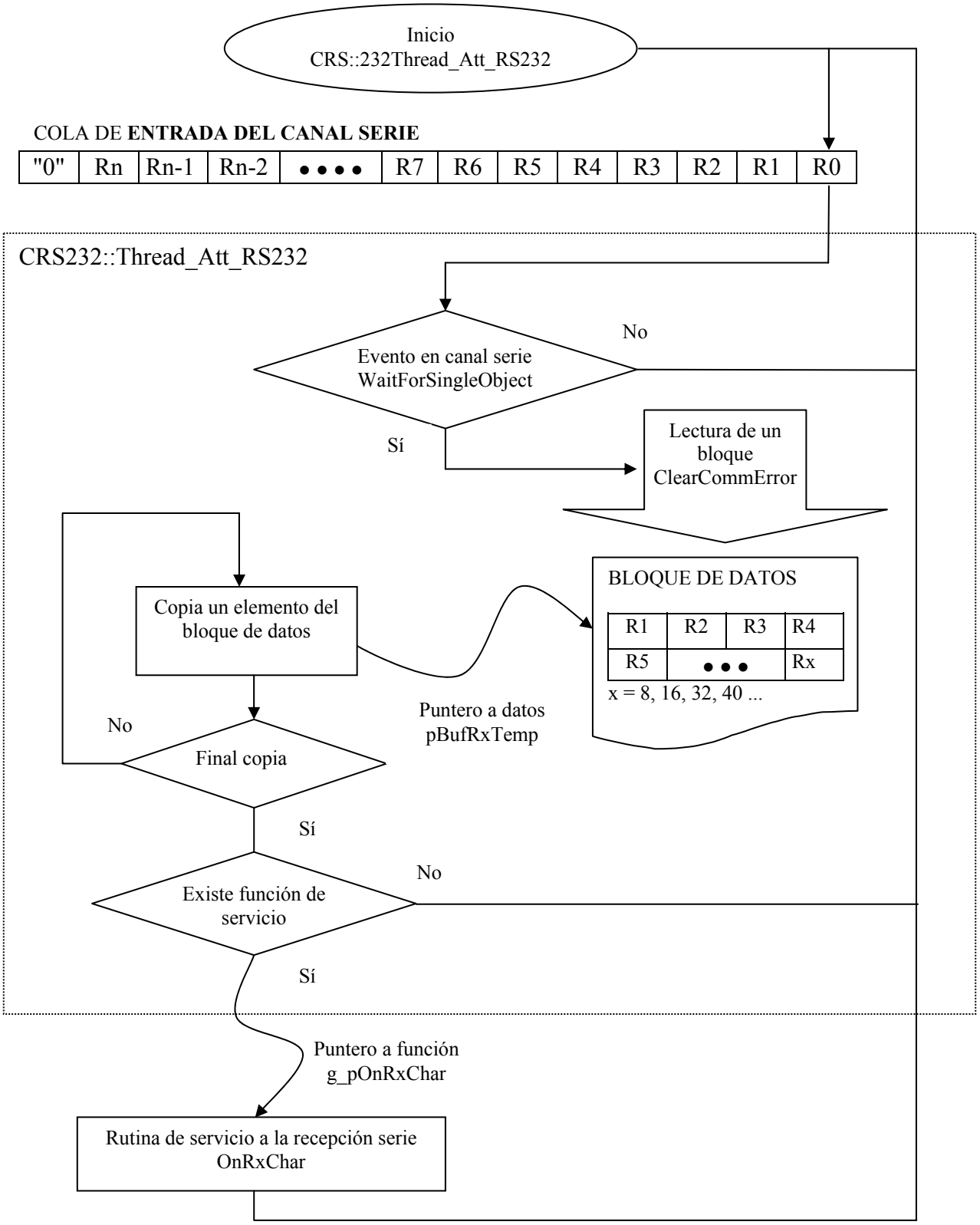

Ilustración 2.29: recepción de datos

2.3.1.2 Gestión de trama

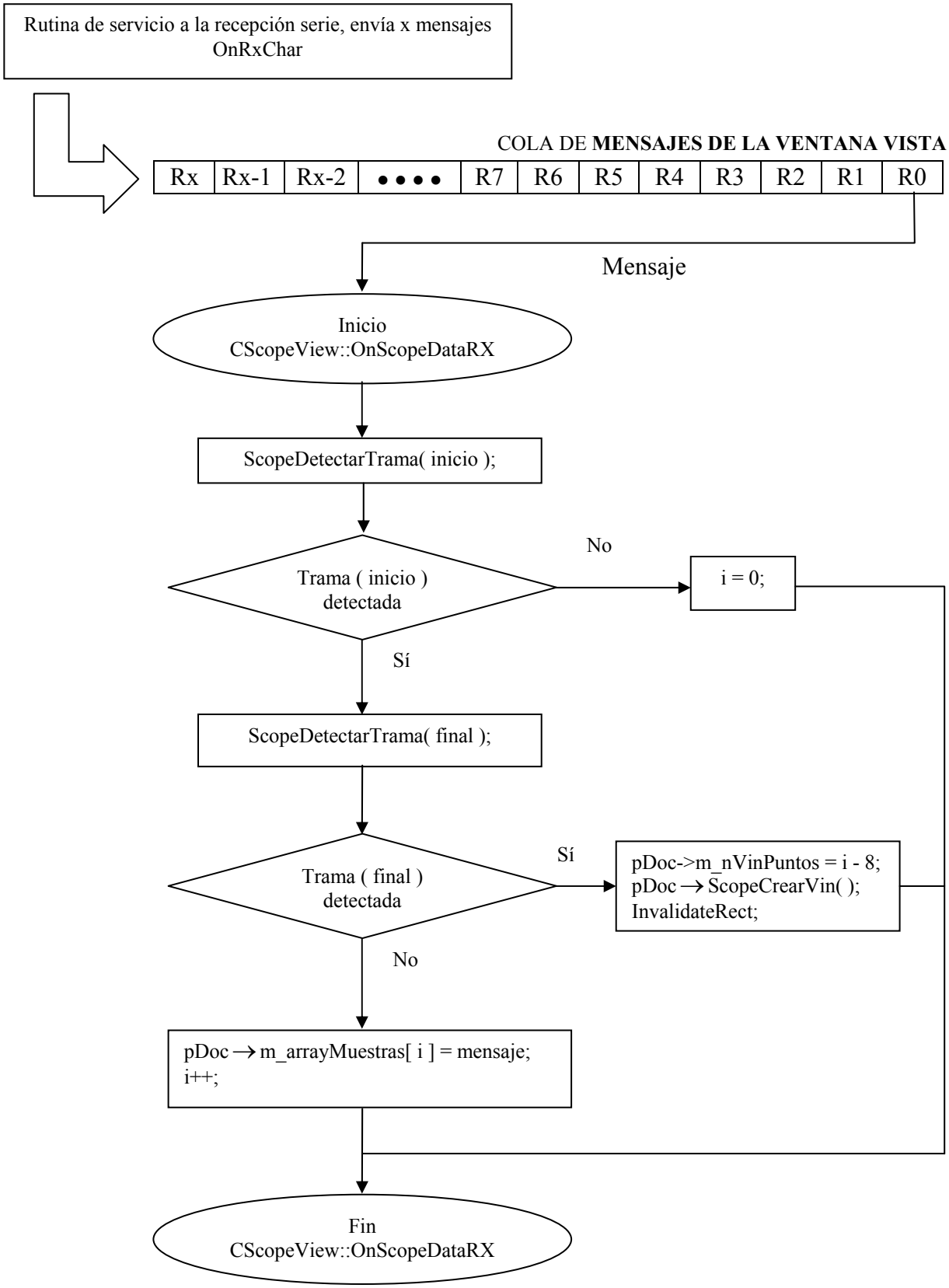

Ilustración 2.30: gestión de trama

2.3.1.3 Reconstrucción de la señal de entrada

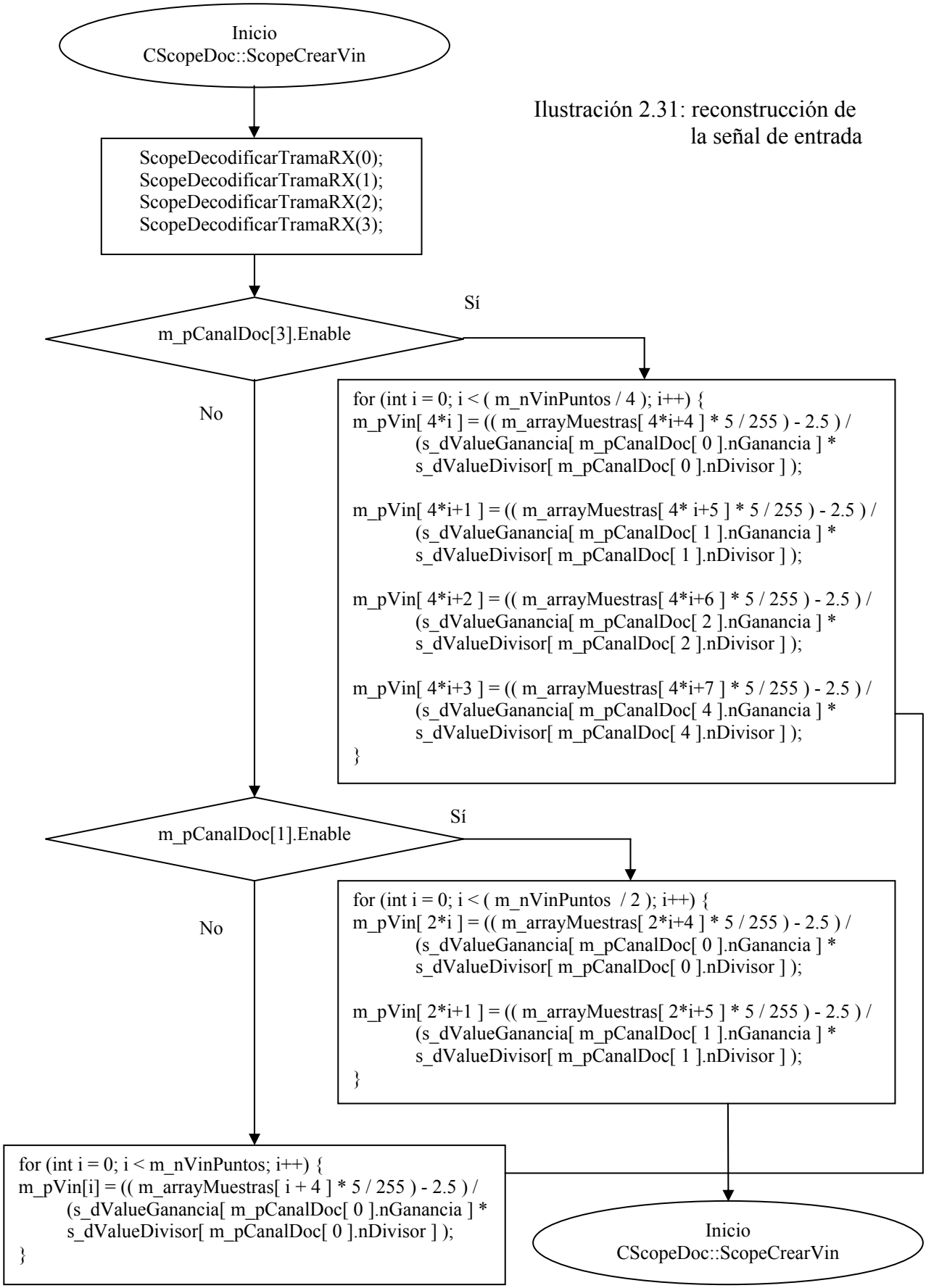

Ilustración 2.31: reconstrucción de la señal de entrada

2.3.1.4 Dibujar ventana vista

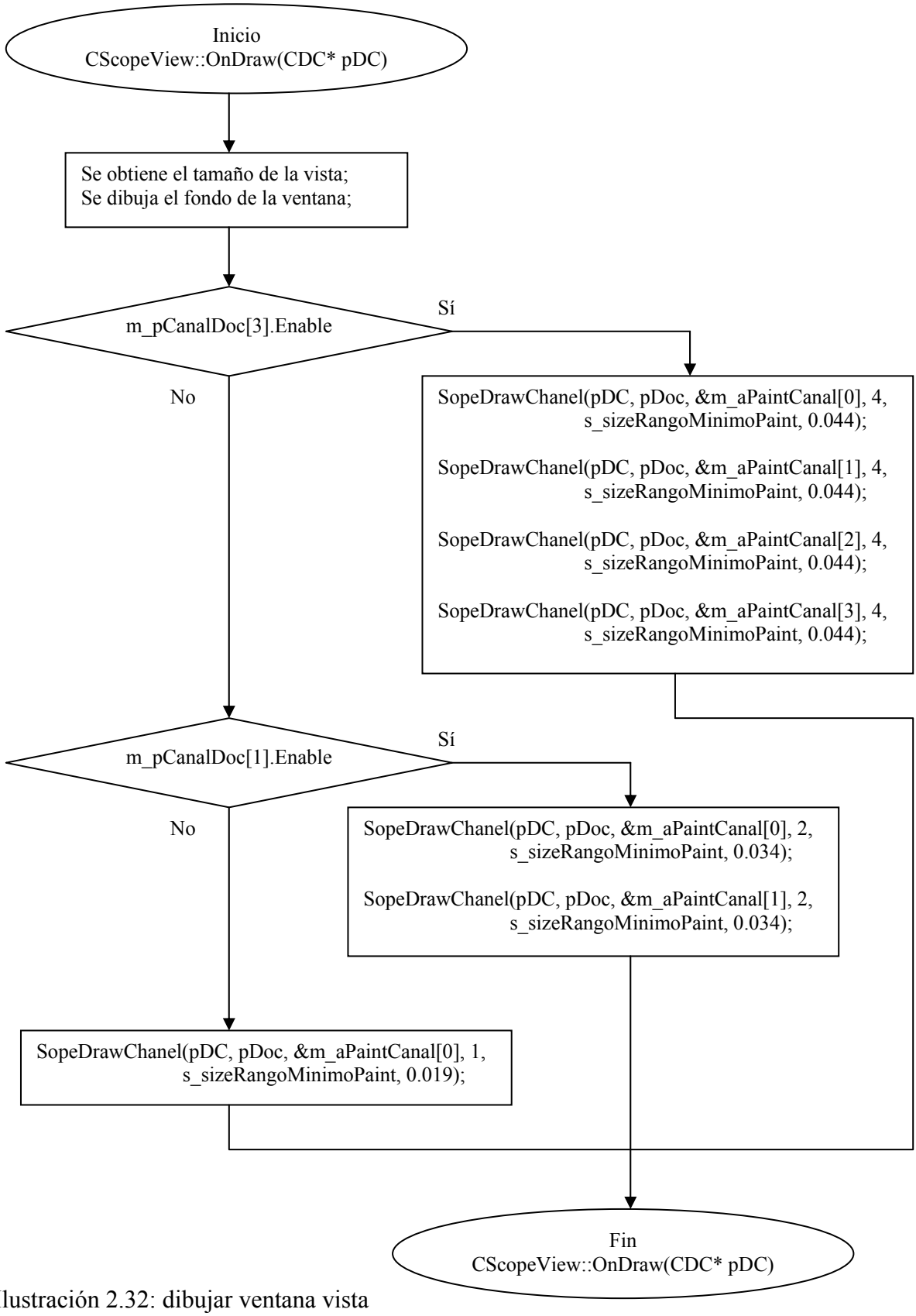

Ilustración 2.32: dibujar ventana vista

2.3.1.5 Dibujar un Canal

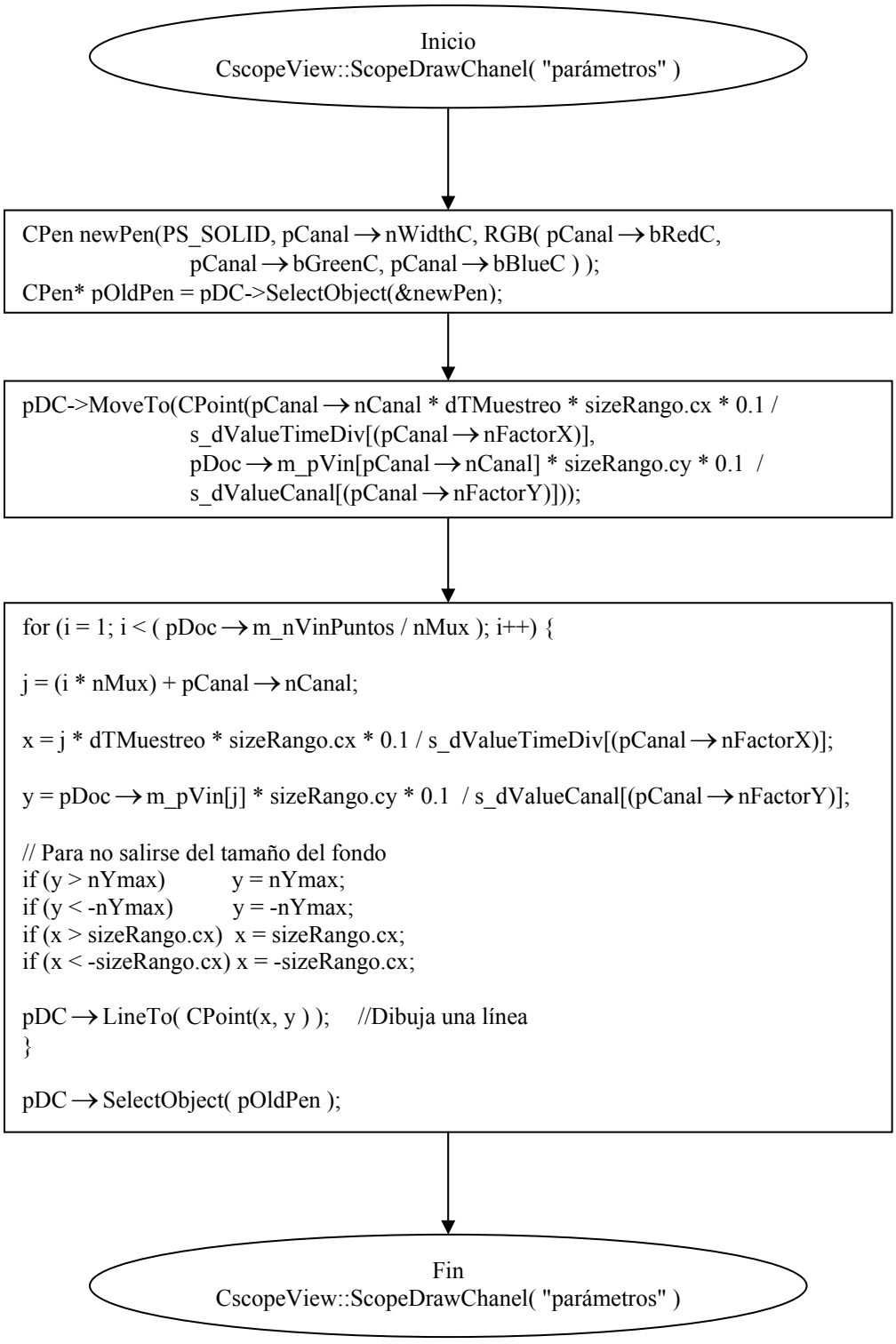

Ilustración 2.33: dibujar un canal

2.3.1.6 Controles del osciloscopio

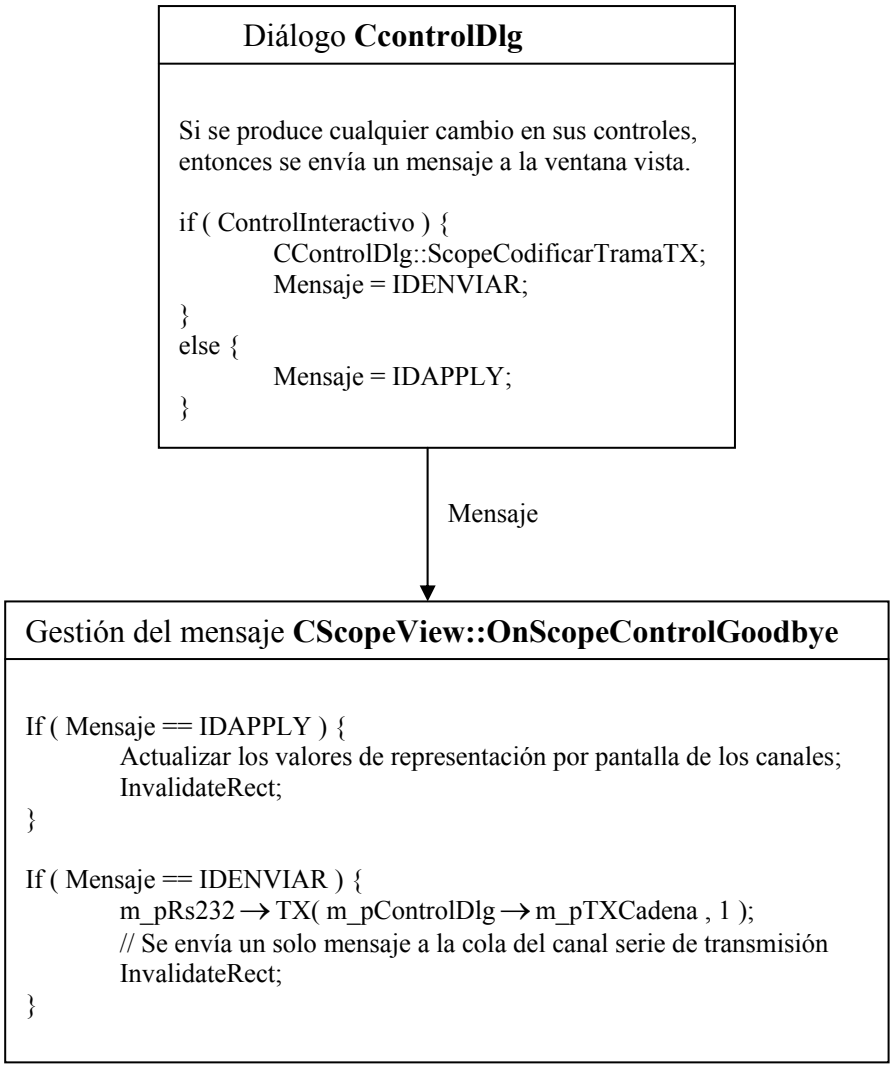

Ilustración 2.34: control del periférico

2.3.1.7 Codificación de órdenes

El proceso de codificación de órdenes lo realiza la función del diálogo de control CControlDlg::ScopeCodificarTramaTX. Ha sido necesaria la creación de una tabla para determinar la orden que precisa el periférico en función del valor de los voltios partido división. También se codifica otros parámetros como el modo AC o DC, el canal referenciado, ... pero son mucho más fáciles de implementar que el cuadro presentado a continuación.

CODIFICACIÓN DE LA GANANCIA DEL PERIFÉRICO												
Gan	Div	Geq	Vin max	Vx max	Rango Vin	V / div	V / div real	Rango real	FAC!	GAN1	GAN2	GAN3
0.5	0.012	0.006	±311	±1.9	622	62.2	100	1000	0	0	0	0
1	0.012	0.012	±208	±2.5	416	41.6	50	500	0	1	0	0
2	0.012	0.024	±104	±2.5	208	20.8	20	200	0	0	1	0
5	0.012	0.06	±41.6	±2.5	83.2	8.32	10	100	0	1	1	0
10	0.012	0.12	±20.8	±2.5	41.6	4.16	5	50	0	0	0	1
20	0.012	0.24	±10.4	±2.5	20.8	2.08	2	20	0	1	0	1
50	**0.012**	**0.6**	**±4.16**	**±2.5**	**8.32**	**0.832**	**1**	**10**	**0**	**0**	**1**	**1**
100	**0.012**	**1.2**	**±2.08**	**±2.5**	**4.16**	**0.416**	**0.5**	**5**	**0**	**1**	**1**	**1**
0.5	1	0.5	±5	±2.5	10	1	1	10	1	0	0	0
1	1	1	±2.5	±2.5	5	0.5	0.5	5	1	1	0	0
2	1	2	±1.25	±2.5	2.5	0.25	0.2	2	1	0	1	0
5	1	5	±0.5	±2.5	1	0.1	0.1	1	1	1	1	0
10	1	10	±0.25	±2.5	0.5	0.05	0.05	0.5	1	0	0	1
20	1	20	±0.125	±2.5	0.25	0.025	0.02	0.2	1	1	0	1
50	1	50	±0.05	±2.5	0.1	0.01	0.01	0.1	1	0	1	1
100	1	100	±0.025	±2.5	0.05	0.005	0.005	0.05	1	1	1	1

Cuadro 2.3: codificación de la ganancia

Los valores en negrita son los descartados.
Gan: ganancia desarrollada por el amplificador inversor de ganancia variable.
Div: atenuación otorgada por el divisor de tensión.
Geq: valor equivalente de aplificación o atenuación.
Vin max: tensión de entrada máxima de un canal.
Vx max: tensión máxima en la entrada del amplificador inversor sumador.
Rango Vin: rango máximo de la tensión de entrada.
V / div: voltios partido división teóricos.
V / div real: voltios partidos división reales o típicos.
Rango real: rango máximo de tensión con V / div real.
FAC!, GAN1, GAN2, GAN3: bits para la codificación de un Byte de Estado.
NOTA: el número de divisiones = 10, Vin max = ±311 y todos los Vx max = ±2.5 son criterios de diseño.

3 PLANOS

4 PRESUPUESTO

4.1 MEDICIONES

CÓDIGO	DESCRIPCIÓN	CANTIDAD	UNIDADES
1	Microcontrolador 87C51	1	Ud.
2	Memoria UM6264AB	1	Ud.
3	Conversor A/D ADC0820CCN	1	Ud.
4	Driver MAX232	1	Ud.
5	Lach MC74HC473AN	2	Ud.
6	Multiplexor MC74HC4051N	3	Ud.
7	Puertas NOR MC74HC02A	1	Ud.
8	Amplificador operacional LM324N	2	Ud.
9	Regulador de tensión UA7805C	1	Ud.
10	Regulador de tensión L7806CV	1	Ud.
11	Regulador de tensión L7906CV	1	Ud.
12	Puente de diodos B40 1500/1000	2	Ud.
13	Relé RS274-059	4	Ud.
14	Driver ULN2003A	1	Ud.
15	Diodo D1N4001	1	Ud.
16	Diodo D1N4004	8	Ud.
17	Diodo zener 1N751	1	Ud.
18	Diodo zener 1N752	1	Ud.
19	Diodo luminiscente	3	Ud.
20	Cristal de cuarzo de 12 MHz	1	Ud.
21	Condensador MKT de 100 nF	18	Ud.
22	Condensador MKT de 10 pF	4	Ud.
23	Condensador MKT de 60 pF	1	Ud.
24	Condensador MKT de 3 uF	1	Ud.
25	Condensador MKT de 330 nF	3	Ud.
26	Condensador cerámico de 30 pF	2	Ud.
27	Condensador electrolítico de 2200 mF	1	Ud.
28	Condensador electrolítico de 680 mF	2	Ud.
29	Condensador electrolítico de 47 uF	1	Ud.
30	Condensador electrolítico de 1 uF	5	Ud.
31	Resistencia de película de carbón de 1/4 W	26	Ud.
32	Resistencia de película de carbón de 1/2 W	8	Ud.
33	Potenciómetro multivuelta de 1/4 W	10	Ud.
34	Conector tipo D montaje PCB de 9 vías	1	Ud.
35	Terminal roscado PCB de 3 vías	2	Ud.
36	Terminal encapsulado PCB de 10 vías	1	Ud.

4.2 PRECIOS UNITARIOS

CÓDIGO	DESCRIPCIÓN	PRECIO UNITARIO EN PTS
1	Microcontrolador 87C51	1500
2	Memoria UM6264AB	1200
3	Conversor A/D ADC0820CCN	2723
4	Driver MAX232	632
5	Lach MC74HC473AN	174
6	Multiplexor MC74HC4051N	243
7	Puertas NOR MC74HC02A	60
8	Amplificador operacional LM324N	135
9	Regulador de tensión UA7805C	134
10	Regulador de tensión L7806CV	327
11	Regulador de tensión L7906CV	450
12	Puente de diodos B40 1500/1000	216
13	Relé RS274-059	622
14	Driver ULN2003A	238
15	Diodo D1N4001	13
16	Diodo D1N4004	22
17	Diodo zener 1N751	14
18	Diodo zener 1N752	14
19	Diodo luminiscente	25
20	Cristal de cuarzo de 12 MHz	150
21	Condensador MKT de 100 nF	10
22	Condensador MKT de 10 pF	10
23	Condensador MKT de 60 pF	10
24	Condensador MKT de 3 uF	30
25	Condensador MKT de 330 nF	15
26	Condensador cerámico de 30 pF	25
27	Condensador electrolítico de 2200 mF	215
28	Condensador electrolítico de 680 mF	54
29	Condensador electrolítico de 47 uF	15
30	Condensador electrolítico de 1 uF	12
31	Resistencia de película de carbón de 1/4 W	7
32	Resistencia de película de carbón de 1/2 W	10
33	Potenciómetro multivuelta de 1/4 W	480
34	Conector tipo D montaje PCB de 9 vías	259
35	Terminal roscado PCB de 3 vías	165
36	Terminal encapsulado PCB de 10 vías	218

4.3 APLICACIÓN DE PRECIOS

CÓD.	DESCRIPCIÓN	CANT.	PRECIO UNITARIO EN PTS	TOTAL PARCIAL EN PTS
1	Microcontrolador 87C51	1	1500	1000
2	Memoria UM6264AB	1	1200	1200
3	Conversor A/D ADC0820CCN	1	2723	2723
4	Driver MAX232	1	632	632
5	Lach MC74HC473AN	2	174	348
6	Multiplexor MC74HC4051N	3	243	729
7	Puertas NOR MC74HC02A	1	60	60
8	Amplificador operacional LM324N	2	135	270
9	Regulador de tensión UA7805C	1	134	134
10	Regulador de tensión L7806CV	1	327	327
11	Regulador de tensión L7906CV	1	450	450
12	Puente de diodos B40 1500/1000	2	216	431
13	Relé RS274-059	4	622	2488
14	Driver ULN2003A	1	238	238
15	Diodo D1N4001	1	13	13
16	Diodo D1N4004	8	22	176
17	Diodo zener 1N751	1	14	14
18	Diodo zener 1N752	1	14	14
19	Diodo luminiscente	3	25	75
20	Cristal de cuarzo de 12 MHz	1	150	150
21	Condensador MKT de 100 nF	18	10	180
22	Condensador MKT de 10 pF	4	10	40
23	Condensador MKT de 60 pF	1	10	10
24	Condensador MKT de 3 uF	1	30	30
25	Condensador MKT de 330 nF	3	15	45
26	Condensador cerámico de 30 pF	2	25	50
27	Condensador electrolítico de 2200 mF	1	215	215
28	Condensador electrolítico de 680 mF	2	54	108
29	Condensador electrolítico de 47 uF	1	15	15
30	Condensador electrolítico de 1 uF	5	12	60
31	Resistencia de película de carbón de 1/4W	26	7	183
32	Resistencia de película de carbón de 1/2W	8	10	80
33	Potenciómetro multivuelta de 1/4 W	10	480	4800
34	Conector tipo D montaje PCB de 9 vías	1	259	259
35	Terminal roscado PCB de 3 vías	2	165	330
36	Terminal encapsulado PCB de 10 vías	1	218	218
TOTAL				**18092**

5 PLIEGO DE CONDICIONES

5.1 CONDICIONES GENERALES

5.1.1 INTRODUCCIÓN

El presente proyecto desarrolla un osciloscopio para Windows 95 utilizando el puerto serie.

Dada la condición de "Final de carrera" del proyecto, las consideraciones de tipo contractual poseen un carácter de suposición.

El presente pliego de Condiciones tiene por objeto definir al contratista el alcance del trabajo y la ejecución cualitativa del mismo.

El alcance del trabajo del Contratista incluye el diseño y preparación de todos los planos, diagramas, lista de material y requisitos para la adquisición e instalación del trabajo.

5.1.2 REGLAMENTOS Y NORMAS

Todas las unidades de obra se ejecutaran cumpliendo las prescripciones indicadas en los Reglamentos de Seguridad y Normas Técnicas de obligado cumplimiento para este tipo de instalaciones, tanto de ámbito nacional, autonómico como municipal, así como todas las otras establecidas en el proyecto.

Se adaptarán además, a las presentes condiciones particulares que complementarán las indicadas por los Reglamentos y Normas citadas.

5.1.3 MATERIALES

Todos los materiales empleados serán de primera calidad, cumplirán las especificaciones y tendrán las características indicadas en el proyecto y en las normas técnicas generales.

Toda especificación o característica de materiales que figuren en uno solo de los documentos del Proyecto, aún sin figurar en los otros es igualmente obligatoria.

En caso de existir contradicción u omisión en los documentos del proyecto, el contratista obtendrá la obligación de ponerlo de manifiesto al Técnico Director de la obra, quien decidirá sobre el particular. En ningún caso podrá suplir la falta directamente, sin la autorización expresa.

No podrá utilizarse materiales que no hayan sido aceptados por el director Técnico.

5.1.4 EJECUCION DEL PROYECTO

Comienzo. El contratista dará comienzo al proyecto en el plazo que figure en el contrato establecido con la Propiedad, o en su defecto a los quince días de la adjudicación definitiva de la obra.

El contratista está obligado a notificar por escrito al Técnico director la fecha de comienzo de los trabajos.

Plazo de ejecución. La obra se ejecutará en el plazo que se estipule en el contrato suscrito con la Propiedad o en su defecto en el que figure en las condiciones de este pliego.

Cuando el ritmo de trabajo establecido por el contratista, no sea el normal, o bien a petición de una de las partes, se podrá convenir una programación de inspecciones obligatorias de acuerdo con el plan de obra.

Libro de órdenes. El contratista dispondrá en la realización de proyecto de un Libro de Ordenes en el que se escribirán las que el Técnico Director estime darle a

través del encargado o persona responsable, sin perjuicio de las que le dé por oficio cuando lo crea necesario y que tendrá la obligación de firmar el enterado.

5.1.5 INTERPRETACION Y DESARROLLO

El Director Técnico es la persona a quien le corresponde interpretar los documentos del proyecto, a él se le tiene que someter cualquier duda, aclaración o contradicción que surja durante la ejecución de la obra, siempre con la suficiente antelación en función de la importancia del asunto.

El contratista se hace responsable de cualquier error de la ejecución motivado por la no consulta y consecuentemente deberá rehacer a su costa los trabajos que correspondan a la correcta interpretación del Proyecto.

El contratista ha de hacer todo los necesario para la buena ejecución de la obra, aún cuando no se halla expresado en el proyecto.

El contratista ha de notificar por escrito o personalmente al director de obra, las fechas en que quedarán preparadas para inspección, cada una de las partes del proyecto o para aquellas que, total o parcialmente deban posteriormente quedar ocultas.

5.1.6 TRABAJOS COMPLEMENTARIOS

El contratista ha de realizar todas los trabajos complementarios necesarios para ejecutar el proyecto tal y como estaba previsto, aunque en él no figuren explícitamente dichos trabajos complementarios. Todo ello sin variación del importe contratado.

5.1.7 MODIFICACIONES

El contratista está obligado a realizar las variaciones (ampliaciones, reducciones o modificaciones) del proyecto siempre que estas no supongan una variación sobre el global proyectado superior al 25%.

Si el contratista, desea realizar alguna modificación, deberá darla a conocer por escrito al Técnico Director, si se considera razonable y se acepta, será confirmada por escrito, así como las nuevas condiciones económicas que mutuamente se acuerden. Si lo anterior no se da como se especifica, no se aceptará modificación alguna.

La valoración se hará de acuerdo, con los valores establecidos en el presupuesto por el Contratista y que ha sido tomado como base del contrato.

5.1.8 REALIZACIÓN DEFECTUOSA

Cuando el contratista halle cualquier unidad de trabajo que no se ajuste a lo especificado en el proyecto o en este Pliego de Condiciones, el Técnico Director podrá aceptarlo o rechazarlo, en el primer caso, éste fijará el precio que crea justo con arreglo a las diferencias que hubiera, estando obligado el Contratista a acatar dicha valoración, en el otro caso, se reconstruirá a expensas del Contratista la parte mal ejecutada sin que ello sea motivo de reclamación económica o de ampliación del plazo de ejecución.

5.1.9 MEDIOS AUXILIARES

Serán de cuenta del Contratista todos los medios y máquinas auxiliares que sean precisas para la ejecución del proyecto. En el uso de los mismos estará obligado a hacer cumplir todos los Reglamentos de Seguridad en el trabajo vigentes y a utilizar los medios de protección a sus operarios.

5.1.10 RECEPCIÓN DEL PROYECTO

Recepción provisional. Una vez terminadas los trabajos, tendrá lugar la recepción provisional y para ello se practicará en ellas un detenido reconocimiento por el Técnico Director y la Propiedad en presencia del Contratista, levantando acta y empezando a correr desde ese día el plazo de garantía si se hallan en estado de ser admitidos.

De no ser admitidos se hará constar en el acta y se darán instrucciones al Contratista para subsanar los defectos observados, fijándose un plazo para ello, expirando el cual se procederá a un nuevo reconocimiento a fin de proceder a la recepción provisional.

Plazo de garantía. El plazo de garantía será como mínimo de un año, contando desde la fecha de la recepción provisional, o bien el que se establezca en el contrato también contado desde la misma fecha. Durante este período queda a cargo del Contratista la conservación del sistema y arreglo de los desperfectos causados por mala construcción.

Recepción definitiva. Se realizará después de transcurrido el plazo de garantía de igual forma que la provisional. A partir de esta fecha cesará la obligación del Contratista de conservar y reparar a su cargo los desperfectos, si bien subsistirán las responsabilidades que pudiera tener por defectos ocultos y deficiencias de causa dudosa.

5.1.11 RESPONSABILIDADES

El contratista es responsable de la ejecución de los trabajos como fija el proyecto, y tendrá que reconstruir toda parte que no se ajuste al programa, sin servir de excusa la razón de que el director de obra haya examinado y reconocido la obra.

El contratista es el único responsable de los posibles fallos cometidos por él o su personal, así como de los accidentes o perjuicios producidos a la propiedad, vecinos o terceros a causa de la inexperiencia o métodos inadecuados.

El contratista es el único responsable del incumplimiento de las disposiciones vigentes en la materia laboral respecto de su personal y por tanto los accidentes que puedan sobrevenir y de los derechos que puedan derivarse de ellos.

5.1.12 FIANZA

En el contrato se establecerá la fianza que el contratista deberá depositar en garantía del cumplimiento del mismo, o, se convendrá una retención sobre los pagos realizados a cuenta del trabajo ejecutado.

De no estipularse la fianza en el contrato se entiende que se adopta como garantía una retención del 5% sobre los pagos a cuenta citados.

En el caso de que el Contratista se negase a hacer por su cuenta los trabajos para ultimar el proyecto en las condiciones contratadas, o a atender la garantía, la Propiedad podrá ordenar ejecutarlas a un tercero, abonando su importe con cargo a la retención o fianza, sin perjuicio de las acciones legales a que tenga derecho la Propiedad si el importe de la fianza no bastase.

La fianza retenida se abonará al Contratista en un plazo no superior a treinta días una vez firmada el acta de recepción definitiva de la obra.

5.2 CONDICIONES TÉCNICAS

5.2.1 CONDICIONES DE LAS PLACAS DE C.I.

Las placas de circuito impreso, deberán ser diseñadas por el fabricante bajo las siguientes normas:

- Ancho de las pistas de señal: 0.5 mm.
- Ancho de las pistas de alimentación: 2 mm.
- Osciladores de cuarzo tumbados sobre plano de masa.
- Los condensadores de desacoplo deberán ir situados lo más cerca posible del pin de alimentación.
- Dimensiones de los taladros:
  - 1 mm para los circuitos integrados y componentes discretos.
  - 1.25 mm para regletas y reguladores de tensión.
  - 3.2 mm para los tornillos de los disipadores.
  - 4 mm para los taladros de sujeción de la placa.

Todas las placas una vez salidas de producción deberán ser testeadas, de tal forma que el índice de fallos en pistas, sea inferior al 1%.

5.2.2 CONDICIONES DE LOS COMPONENTES ELECTRÓNICOS

La premisa básica a seguir en la compra de los componentes electrónicos, es buscar componentes de marcas de reconocido prestigio y que posean un índice de rechazo en producción, inferior al 20%.

Así por ejemplo, se recomienda la utilización de componentes de la firma MOTOROLA o cualquiera de sus subsidiaras (RCA. AMD) que hasta la fecha han demostrado un muy alto grado de fiabilidad de los componentes suministrados.

No se recomienda recurrir bajo ningún concepto a las pleyades de fabricantes de Oriente que han surgido en Corea, Singapur, Malasia, etc., puesto que en anteriores producciones han puesto de manifiesto un bajísimo grado de fiabilidad, dándose el caso de encontrar partidas enteras de componentes que a los 6 meses de funcionamiento han fallado íntegramente.

5.2.3 CONDICIONES DEL MONTAJE DE PLACAS

El montaje de placas deberá ser realizado por inserción automatizada, puesto que el índice de errores es prácticamente nulo, no así cuando se recurre al montaje manual de componentes.

La soldadura de las placas debe ser realizada por ola, con estaño de buena calidad, y una vez finalizado el proceso, las placas deben ser perfectamente limpias con algún producto específico basado en flúor de los muchos que hay disponibles en el mercado.

De cada partida de placas producidas, al menos lo de ellas deberán ser verificadas en horno a 40 grados centígrados y en funcionamiento de tal forma que se les proporcione el equivalente a 6 meses de funcionamiento ininterrumpido durante las pruebas, sea superior al 20%, la partida entera deberá ser retirada y sustituida por una nueva.

5.3 CONDICIONES FACULTATIVAS

5.3.1 NORMAS A SEGUIR

El diseño de la instalación eléctrica estará de acuerdo con las exigencias o recomendaciones expuestas en la última edición de los siguientes códigos:

- Reglamento Electrotécnico de Baja Tensión e Instrucciones complementarias.
- Normativa UNE.
- Normativa DIN.
- Plan nacional y ordenanza general de Seguridad e Higiene en el trabajo.
- Normas de la Compañía Suministradora.
- Publicaciones del Comité Electrotécnico Internacional (CEI).
- Lo indicado en este pliego de condiciones con preferencia a todos los códigos y normas.

5.3.2 PERSONAL

El Contratista tendrá al frente de la obra un encargado con autoridad sobre los demás operarios y conocimientos acreditados y suficientes para la ejecución del proyecto.

El encargado recibirá, cumplirá y transmitirá las instrucciones y órdenes del Técnico Director de la obra.

El Contratista tendrá el número y clase de operarios que haga falta para el volumen y naturaleza de los trabajos que se realicen, los cuales serán de reconocida aptitud y experimentados en el oficio. El Contratista estará obligado a separar de la realización del proyecto, a aquel personal que a juicio del Técnico Director no cumpla con sus obligaciones, realice el trabajo defectuosamente, bien por falta de conocimientos o por obrar de mala fe.

5.3.3 RECONOCIMIENTO Y ENSAYOS PREVIOS

Cuando lo estime oportuno el Técnico Director, podrá encargar y ordenar el análisis, ensayo o comprobación de los materiales, elementos o instalaciones, bien sea en fábrica de origen, laboratorios oficiales o en la misma obra, según crea más conveniente, aunque estos no estén indicados en este pliego.

En el caso de discrepancia, los ensayos o pruebas se efectuarán en el laboratorio oficial que el Técnico Director de obra designe.

Los gastos ocasionados por estas pruebas y comprobaciones, serán por cuenta del Contratista.

5.3.4 ENSAYOS

Antes de la puesta en servicio del sistema eléctrico, el Contratista habrá de hacer los ensayos adecuados para probar, a la entera satisfacción del Técnico Director del proyecto, que todo el equipo, aparatos y cableado han sido instalados correctamente de acuerdo con las normas establecidas y están en condiciones satisfactorias para le funcionamiento.

Todos los ensayos serán presenciados por el Ingeniero que representa el Técnico Director de obra.

Los resultados de los ensayos serán pasados en certificados indicando fecha y nombre de la persona a cargo del ensayo, así como categoría profesional.

Los cables, antes de ponerse en funcionamiento, se someterán a un ensayo de resistencia de aislamiento entre fase y tierra.

5.3.5 ENSAYOS DE APARELLAJE

Antes de poner el aparellaje bajo tensión, se medirá la resistencia de aislamiento de cada equipo entre fases y tierra.

Las medidas deben repetirse con los interruptores en posición de funcionamiento.

Todo relé de protección que sea ajustable será calibrado y ensayado, usando contador de ciclos, caja de carga, amperímetro y voltímetro, según se necesite.

Se dispondrá, en lo posible, de un sistema de protección selectiva. De acuerdo con esto, los relés de protección se elegirán y coordinarán para conseguir un sistema que permita actuar primero el dispositivo de interrupción más próximo a la falta.

Todos los interruptores automáticos se colocarán en posición de prueba y cada interruptor será cerrado y disparado desde su interruptor de control. Los interruptores deben ser disparados por accionamiento manual y aplicando corriente a los relés de protección. Se comprobarán todos los enclavamientos.

5.4 CONDICIONES ECONOMICAS

5.4.1 PRECIOS

El contratista presentará, al formalizarse el contrato, relación de los precios de las unidades de trabajo que integran el proyecto, los cuales de ser aceptados tendrán valor contractual y se aplicarán a las posibles variaciones que puedan haber.

Estos precios unitarios, se entiende que comprenden la ejecución total de la unidad del proyecto, incluyendo todos los trabajos aún los complementarios y los materiales así como la parte proporcional de imposición fiscal, las cargas laborales y otros gastos repercutibles.

En caso de tener que realizarse unidades de trabajo no prevista en el proyecto, se fijará su precio entre el Técnico Director y el Contratista antes de iniciar la obra y se presentará a la propiedad para su aceptación o no.

5.4.2 ABONO DEL PROYECTO

En el contrato se deberá dejar detalladamente la forma y plazas que se abonarán las partes del proyecto. Las liquidaciones parciales que puedan establecerse tendrán carácter de documentos provisionales a buena cuenta, sujeto a las certificaciones que resulten de la liquidación final. No suponiendo, dichas liquidaciones, aprobación ni recepción de las obras que comprenden.

Terminadas las obras se procederá a la liquidación final que se efectuará de acuerdo con los criterios establecidos en el contrato.

5.4.3 REVISIÓN DE PRECIOS

En el contrato se establecerá si el contratista tiene derecho a revisión de precios y la fórmula a aplicar para calcularla. En defecto de esta última, se aplicará a juicio del Técnico Director alguno de los criterios oficiales aceptados.

5.4.4 PENALIZACIONES

Por retraso en los plazos de entrega de las obras, se podrán establecer tablas de penalización cuyas cuantías y demoras se fijarán en el contrato.

5.4.5 CONTRATO

El contrato se formalizará mediante documento privado, que podrá elevarse a escritura a petición de cualquiera de las partes. Comprenderá la adquisición de todos los materiales, transporte, mano de obra, medios auxiliares para la ejecución de la obra proyectada en el plazo estipulado, así como la reconstrucción de las unidades defectuosas, la realización de las obras complementarias y las derivadas de las modificaciones que se introduzcan durante la ejecución, éstas últimas en los términos previstos.

La totalidad de los documentos que componen el Proyecto técnico de la obra serán incorporados al contrato y tanto el contratista como la Propiedad deberán firmarlos en testimonio de que los conocen y aceptan.

5.4.6 RESCISIÓN DEL CONTRATO

Causas de rescisión: Se consideran causas suficientes para la rescisión del contrato las siguientes:

1- Muerte o incapacitación del Contratista.
2- La quiebra del contratista.
3- Modificación del proyecto cuando produzca alteración en más o menos 25% del valor contratado.
4- Modificación de las unidades de obra en número superior al 40% del original.
5- La no iniciación de los trabajos en el plazo estipulado cuando sea por causas ajenas a la Propiedad.
6- La suspensión de las obras ya iniciadas siempre que el plazo de suspensión sea mayor de seis meses.
7- Incumplimiento de las condiciones del Contrato cuando implique mala fe.
8- Terminación del plazo de ejecución de la obra sin haberse llegado a completar ésta.
9- Actuación de mala fe en la ejecución de los trabajos.
10- Destajar o subcontratar la totalidad o parte de los trabajos a terceros sin la autorización del Técnico Director y la propiedad.

5.4.7 LIQUIDACION EN CASO DE RESCISION DEL CONTRATO

Siempre que se rescinda el contrato por causas anteriores o bien por acuerdo de ambas partes, se abonará al Contratista las unidades del proyecto ejecutado y los materiales acopiados que reúnan las condiciones y sean necesarios para el mismo.

Cuando se rescinda el contrato llevará implícito la retención de la fianza para obtener los posibles gastos de conservación de el período de garantía y los derivados del mantenimiento hasta la fecha de nueva adjudicación.

6 ANEXO

6.1 CÓDIGO DE BAJO NIVEL

```
;*****************************************************************************
;***                                                                       ***
;***    SOFTWARE PARA EL CONTROL DEL OSCILOSCOPIO PARA WINDOWS 95          ***
;***                                                                       ***
;*****************************************************************************

;-----------------------------------------------------------------------------
;---                                                                       ---
;---    Por: Raúl Bartolomé Castro                                         ---
;---    Versión: 2.1.                                                      ---
;---    Fecha: 28 de Enero del 2000                                        ---
;---    Archivo: OSCIL1.ASM                                                ---
;---                                                                       ---
;---    Descripción:                                                       ---
;-----------------------------------------------------------------------------
;---    Software para el microcontrolador 89C51, con una frecuencia de      ---
;---            trabajo de 12 MHz => 1 Ciclo Maquina (CM) = 1 us            ---
;-----------------------------------------------------------------------------
;---    Existe una memoria de datos externa de 8 KBy en                    ---
;---            @InicialMemoria = 1xx0 0000 0000 0000 B = 8000 H           ---
;---            @FinalMemoria   = 1xx1 1111 1111 1111 B = 9FFF H           ---
;-----------------------------------------------------------------------------
;---    El Byte de Control ( BCON ) esta mapeado en memoria el la dirección:
;---            @BCON = 0xxx xxxx xxxx xxxx B = 0000H                      ---
;---    BCON.7--BCON.6--BCON.5--BCON.4--BCON.3--BCON.2--BCON.1--BCON.0     ---
;---    FAC4!---FAC3!---FAC2!---FAC1!---GAN1----GAN2----GAN3----AC/DC!     ---
;---                                                                       ---
;---            FACx! = 0 => Canal x por factor de 0.012                   ---
;---            FACx! = 1 => Canal x por factor de 1 (sin factor)          ---
;---            Donde x = 1, 2, 3, 4                                       ---
;---    GAN3--GAN2--GAN1  GANANCIA                                         ---
;---    0-----0-----0   => 0.5                                             ---
;---    0-----0-----1   => 1                                               ---
;---    0-----1-----0   => 2                                               ---
;---    0-----1-----1   => 5                                               ---
;---    1-----0-----0   => 10                                              ---
;---    1-----0-----1   => 20                                              ---
;---    1-----1-----0   => 50                                              ---
;---    1-----1-----1   => 100                                             ---
;---            AC/DC! = 0 => Modo DC                                      ---
;---            AC/DC! = 1 => Modo AC                                      ---
;-----------------------------------------------------------------------------
```

```
;-------------------------------------------------------------------------
;---    El Byte de Estado ( BEST ) es especifico para cada canal, que a   ---
;---            a su vez es el byte que se transmite por canal serie        ---
;---    BEST.7--BEST.6--BEST.5--BEST.4--BEST.3--BEST.2--BEST.1--BEST.0      ---
;---    SBUF.7--SBUF.6--SBUF.5--SBUF.4--SBUF.3--SBUF.2--SBUF.1--SBUF.0      ---
;---    CH1-----CH0-----E-------FAC!----GAN1----GAN2----GAN3----AC/DC!    ---
;---                                                                        ---
;---            CH1----CH0     CANAL                                        ---
;---            0------0    => 1                                            ---
;---            0------1    => 2                                            ---
;---            1------0    => 3                                            ---
;---            1------1    => 4                                            ---
;---    E : Enable. -- El Canal 1 siempre esta habilitado !!                ---
;---    E  CH1 CH0    CODIFICACION                                          ---
;---    0---0---0  => Control de disparo manual => FAC!=Enable, GAN1=TRIGGER
;---    1---0---0  => Referenciado Canal 1                                  ---
;---    0---0---1  => Referenciado Canal 2 e inhabilitado  Canal 2   ---
;---    1---0---1  => Referenciado Canal 2 y habilitado    Canal 2          ---
;---    0---1---0  => Frecuencia de muestreo variable                       ---
;---    1---1---0  => Referenciado Canal 3                                  ---
;---    0---1---1  => Referenciado Canal 3 e inhabilitados Canales 2, 3 y 4
;---    1---1---1  => Referenciado Canal 4 y habilitados   Canales 2, 3 y 4
;---            FAC! = 0 => Por el factor de 0.012 del Canal referenciado ---
;---            FAC! = 1 => Por el factor de 1 del Canal referenciado       ---
;---    GAN3--GAN2--GAN1  GANANCIA DEL CANAL REFERENCIADO                   ---
;---    0-----0-----0   => 0.5                                              ---
;---    0-----0-----1   => 1                                                ---
;---    0-----1-----0   => 2                                                ---
;---    0-----1-----1   => 5                                                ---
;---    1-----0-----0   => 10                                               ---
;---    1-----0-----1   => 20                                               ---
;---    1-----1-----0   => 50                                               ---
;---    1-----1-----1   => 100                                              ---
;---            AC/DC! = 0 => Modo DC del Canal referenciado                ---
;---            AC/DC! = 1 => Modo AC del Canal referenciado                ---
;-------------------------------------------------------------------------
;---    El Byte de Modo ( BMOD ) especifica el modo de funcionamiento del ---
;---            osciloscopio de la siguiente manera                         ---
;---    BMOD.7--BMOD.6--BMOD.5--BMOD.4--BMOD.3--BMOD.2--BMOD.1--BMOD.0    ---
;---                                                                        ---
;---            BMOD.1--BMOD.0    MODO DE MULTIPLEXACION Y CANAL/ES         ---
;---            x-------1      => Sin multiplexación: Canal 1               ---
;---            1-------0      => Con multiplexación: Canales 1 y 2         ---
;---            0-------0      => Con multiplexación: Canales 1, 2, 3 y 4 ---
;---    BMOD.2 = 0 => Sin control manual, muestreo "infinito".              ---
;---    BMOD.2 = 1 => Con control manual.                                   ---
;---            Si (BMOD.2 = 1) entonces                                    ---
;---            BMOD.3 = 0 => Espera activa                                 ---
;---            BMOD.2 = 1 => Inicia muestreo (disparo).                    ---
;-------------------------------------------------------------------------
        .org 00H
        ljmp INICIO

        .org 23H
        ljmp SERVCOM

        .org 33H
INICIO:
```

```
;-------------------------------------------------------------------------
;---    INICIALIZACIÓN DEL OSCILOSCOPIO                                  ---
;-------------------------------------------------------------------------

        mov A,#F0H      ;1111 0000 B = F0 H. Será para el BC
                        ;FAC4!, FAC3!, FAC3!, FAC1! = 1 => Sin factor
                        ;GAN3 = 0, GAN2 = 0, GAN1 = 0 => Ganancia de 0.5
                        ;AC/DC! = 0 => Modo DC
        mov DPTR,#0000H ;Dirección del BC
        movx @DPTR,A    ;Actúo sobre la electrónica analógica

        mov 20H,#01H    ;20H guarda el modo del osciloscopio. BMOD
                        ;Sin multiplexación
                        ;muestreo sin control manual

        mov 21H,A       ;21H guarda el control del Canal 1. BCON1
        mov 22H,A       ;22H guarda el control del Canal 2. BCON2
        mov 23H,A       ;23H guarda el control del Canal 3. BCON3
        mov 24H,A       ;24H guarda el control del Canal 4. BCON4

        mov 25H,#30H    ;0011 0000 B = 30 H. 25H guarda el estado del Canal 1. BEST1
        mov 26H,#50H    ;0101 0000 B = 50 H. 26H guarda el estado del Canal 2. BEST2
        mov 27H,#90H    ;1001 0000 B = 90 H. 27H guarda el estado del Canal 3. BEST3
        mov 28H,#D0H    ;1101 0000 B = D0 H. 28H guarda el estado del Canal 4. BEST4

        mov 29H,#00H    ;Mascara de 1s. Para decodificar la recepción
        mov 2AH,#00H    ;Mascara de 0s. Para decodificar la recepción

        setb INT1       ;A/D listo para ser disparado

;------------Programación del Timer e Interrupciones-------------------------
                        ;Si reset => IE (Interrupt Eneble Register) = 0XX0 0000 b
                        ;IE = EA:0,  X:0,   X:0,   ES:0
                        ;     ET1:0, EX1:0, ET0:0, EX0:0
        mov IE,#90H     ;Habilito solo la interrupción serie y todas
                        ;aquellas que estén a uno ( solo la serie)
                        ;IE = EA:1,  X:0,   X:0,   ES:1
                        ;     ET1:0, EX1:0, ET0:0, EX0:0
                        ;Si reset => IP (Interrupt Priority Register) = XXX0 0000 b
                        ;IP = X:x,   X:x,   X:x,   PS:0
                        ;     PT1:0, PX1:0, PT0:0, PX0:0
        mov TMOD,#20H   ;Timer 1:Disparado por soft y modo 8b con autorrecarga
                        ;Si reset => TMOD (Timer/Counter Mode Control Register)= 00 H
                        ;TMOD = GATE:0, C/T!:0, M1:1, M0:0 (TIMER 1)
                        ;       GATE:0, C/T!:0, M1:0, M0:0 (TIMER 0)
                        ;Si reset => TCON (Timer/Counter Control Register) = 00H
                        ;TCON = TF1:0, TR1:0, TF0:0, TR0:0
                        ;       IE1:0, IT1:0, IE0:0, IT0:0
;------------Fin Programación del Timer e Interrupciones---------------------
```

```
;------------Programación del interfaz de comunicaciones serie---------------
        mov SCON,#50H   ;Modo 1: UART 8 bits, 1b start, 8b datos y 1b stop
                        ;SM2=0: Un solo procesador
                        ;REN=1: Habilito la recepción serie
                        ;TI=0: Buffer de transmisión disponible
                        ;RI=0: Buffer de recepción libre
                        ;Velocidad de comunicación variable con Timer 1
                        ;Si reset => SCON (Serial Port Control Register)= 00 H
                        ;SCON = SM0:0,SM1:1,SM2=0,REN=1
                        ;       TB8:0,RB8:0,TI:0,RI:0
        mov PCON,#80H   ;Timer 1 a frecuencia boble
                        ;Si reset => PCON (Power Control Register)= 0XXX00000 B
                        ;PCON = SMOD:1,X:0,X:0,X:0
                        ;       GF1:0,GF0:0,PD:0,IDL:0
        mov TL1,#FAH    ;Para Timer 1, generador de baudios o bits/s
        mov TH1,#FAH    ;Tabla de velocidad de comunicación
                        ; TH1 - Baudios o bps
                        ;CC H -   1200 bps
                        ;EG H -   2400 bps
                        ;FE H -   4800 bps
                        ;FA H -   9600 bps
                        ;FD H -  19200 bps
                        ;FE H -  38400 bps
                        ;FF H - 115000 bps
                        ;Baudios = 2^SMOD/32 * FrecuenciaOscilador/12*(256-TH1)
                        ;TH1 = 256 - (FrecuenciaOscilador/xxx)/Baudios
                        ;xxx = 384 si SMOD = 0
                        ;xxx = 192 si SMOD = 1
;------------Fin Programación del interfaz de comunicaciones serie-----------

;-------------------------------------------------------------------------
;---    FIN INICIALIZACIÓN DEL OSCILOSCOPIO                              ---
;-------------------------------------------------------------------------
        setb TR1        ;Disparo el Timer 1, comienza a transmitir

INFINITO:
        mov DPTR,#8000H ;Dirección inicial de la memoria externa

;------------------Guardar en memoria la trama de inicio---------------------
        mov A,#FBH
        movx @DPTR,A
        inc DPTR        ;8001 H
        mov A,#04H
        movx @DPTR,A
        inc DPTR        ;8002 H
        mov A,#FBH
        movx @DPTR,A
        inc DPTR        ;8003 H
        mov A,#04H
        movx @DPTR,A
        inc DPTR        ;8004 H
;--------------Fin Guardar en memoria trama de inicio------------------------

;----Inhabilitar interrupción serie y realizar copia de BMOD, BEST y BCON----
                        ;La copia se realiza en la zona Scratch Pad
                        ;Esta zona esta comprendida desde 30H hasta 7FH
        clr IE.4        ;Inhabilito la interrupción serie
        mov 30H,20H     ;BMOD
        mov 31H,21H     ;BCON1
        mov 32H,22H     ;BCON2
        mov 33H,23H     ;BCON3
        mov 34H,24H     ;BCON4
        mov 35H,25H     ;BEST1
        mov 36H,26H     ;BEST2
        mov 37H,27H     ;BEST3
        mov 38H,28H     ;BEST4
        setb IE.4       ;Habilito la interrupción serie
;----Fin Inhabilitar interrupción serie y realizar copia de BMOD, BEST y BCON
```

```
;-----------Guardar en memoria los BEST-------------------------------------
        mov A,35H       ;Cargo la copia BEST1 en A
        movx @DPTR,A    ;Guardo en memoria externa el BE1
        inc DPTR        ;8005 H
        mov A,36H       ;Cargo la copia BEST2 en A
        movx @DPTR,A    ;Guardo en memoria externa el BE2
        inc DPTR        ;8006 H
        mov A,37H       ;Cargo la copia BEST3 en A
        movx @DPTR,A    ;Guardo en memoria externa el BE3
        inc DPTR        ;8007 H
        mov A,38H       ;Cargo la copia BEST4 en A
        movx @DPTR,A    ;Guardo en memoria externa el BE4
;-----------Fin Guardar en memoria los BEST---------------------------------

;------------Tratamiento del control manual---------------------------------
NOYA:
        mov A,20H       ;Cargo el BMOD
        jnb A.2,NOMANUAL
        jnb A.3,NOYA
        clr 20H.3
NOMANUAL:
;------------Fin Tratamiento del control manual-----------------------------

;-------------------------------------------------------------------------
;---     MUESTREAR Y GUARDAR EN MEMORIA 1012 MUESTRAS                    ---
;-------------------------------------------------------------------------
        clr T0          ;Selecciono el Canal 1
        clr T1

        mov A,30H       ;Copia de BMOD
        jb A.0,SINMUXX
        jb A.1,MUX1Y2X
        sjmp MUX1234
SINMUXX:
        ljmp SINMUX
MUX1Y2X:
        ljmp MUX1Y2
MUX1234:
;-----------Con multiplexación de canales 1, 2, 3 y 4--------------------
        mov A,31H       ;Cargo la copia de BCON1
        mov DPTR,#0000H ;Dirección del BCON
        movx @DPTR,A    ;Actúo sobre la electrónica con BCON1

        mov DPTR,#8008H ;Dirección inicial para los datos

        clr INT1        ;2By,1CM;Disparo el A/D
        nop             ;1By,1CM;
        nop             ;1By,1CM;
        nop             ;1By,1CM;Para esperar los 2.5 us
        mov A,P1        ;2By,1CM;Leo el valor de la conversión
        setb INT1       ;2By,1CM;En 500 ns el A/D listo para otra conversión

        nop             ;1By,1CM;
        mov R7,#02H     ;2By,1CM; Retardo de 3 * 10 + 1 CM
RETARDO4:
        nop             ;1By,1CM;
        nop             ;1By,1CM;
        nop             ;1By,1CM;
        nop             ;1By,1CM;
        nop             ;1By,1CM;
        nop             ;1By,1CM;
        nop             ;1By,1CM;
        nop             ;1By,1CM;
        djnz R7,RETARDO4   ;2By,2CM;
```

```
MUES24:
        nop             ;1By,1CM;
        nop             ;1By,1CM;
        nop             ;1By,1CM;
        nop             ;1By,1CM;
        nop             ;1By,1CM;
        nop             ;1By,1CM;
MUES4:
        clr INT1        ;2By,1CM;Disparo el A/D
        movx @DPTR,A    ;1By,2CM;Guardo en memoria la muestra anterior

        jb T0,T0ES14    ;3By,2CM;
        jnb T0,T0ES04   ;3By,2CM;
T0ES14:
        nop
        nop
        jb T1,T1ES1A    ;3By,2CM;
        jnb T1,T1ES0A   ;3By,2CM;
T1ES1A:
        nop             ;1By,1CM;
        nop             ;1By,1CM;T0 = 1, T1 = 1
        clr T1          ;2By,1CM;
        mov A,31H       ;2By,1CM;
        sjmp YAMUX      ;2By,2CM;
T1ES0A:                 ;T0 = 1, T1 = 0
        setb T1         ;2By,1CM;
        mov A,33H       ;2By,1CM;
        sjmp YAMUX      ;2By,2CM;
T0ES04:
        jb T1,T1ES1B    ;3By,2CM;
        jnb T1,T1ES0B   ;3By,2CM;
T1ES1B:
        nop             ;1By,1CM;
        nop             ;1By,1CM;T0 = 0, T1 = 1
        nop             ;1By,1CM;
        mov A,34H       ;2By,1CM;
        sjmp YAMUX      ;2By,2CM;
T1ES0B:                 ;T0 = 0, T1 = 0
        nop             ;1By,1CM;
        mov A,32H       ;2By,1CM;
        sjmp YAMUX      ;2By,2CM;
YAMUX:
        cpl T0          ;2By,1CM;CA2 de T0
        mov R1,DPL      ;2By,2CM;Guardo el DPTR
        mov R2,DPH      ;2By,2CM;
        mov DPTR,#0000H ;2By,2CM;@ de BCON
        movx @DPTR,A    ;1By,2CM;Actúo sobre la electrónica
        mov DPL,R1      ;2By,2CM;Restauro el DPTR
        mov DPH,R2      ;2By,2CM

        inc DPTR        ;1By,2CM;Incremento el puntero
        mov A,P1        ;2By,1CM;Leo el valor de la conversión actual
        setb INT1       ;2By,1CM;En 500 ns el A/D listo para otra conversión

        mov 0H,DPL          ;3By,2CM;En R0 la parte baja de DPTR
        xrl 0H,#FCH         ;3By,2CM;R0 xor #FC, si son iguales => R0 = 0
        cjne R0,#0,MUES24   ;3By,2CM;If ( R0 != 0 ) goto MUESTRA

        mov 0H,DPH          ;3By,2CM;En R0 la parte baja de DPTR
        xrl 0H,#83H         ;3By,2CM;R0 xor #83, si son iguales => R0 = 0
        cjne R0,#0,MUES4    ;3By,2CM;If ( R0 != 0 ) goto MUESTRA

        ljmp FSAMPLER
        ;CMtotal = 1+2+2+2+2+2+1+2+1+2+2+2+2+2+2+2+1+1+2+2+2+2+2+2+1=44CM
;-----------Fin Con multiplexacion de canales 1, 2, 3 y 4--------------------
```

```
SINMUX:
;--------------------Sin multiplexación-------------------------------------
        mov A,31H       ;Cargo la copia de BCON1
        mov DPTR,#0000H ;Dirección del BCON
        movx @DPTR,A    ;Actúo sobre la electrónica con BCON1

        mov R7,#04H
RETARDO1:
        nop
        nop
        nop
        nop
        djnz R7,RETARDO1

        mov DPTR,#8008H ;Dirección inicial para los datos
                        ;Quiero crear una trama de 1024 Bytes.
                        ;1024 = 4 By trama inicial + 4 By de Bytes de Estado +
                        ;       x By de datos + 4 By trama final =>
                        ;x By de datos = 1024 - 4 - 4 - 4 = 1012 D = 3F4 H
                        ;@ inicial datos = 8008 H
                        ;@ final datos   = 8007 H + 3F4 H = 83FB H

        clr INT1        ;2By,1CM;Disparo el A/D
        nop             ;1By,1CM;
        nop             ;1By,1CM;
        nop             ;1By,1CM;Para esperar los 2.5 us
        mov A,P1        ;2By,1CM;Leo el valor de la conversión
        setb INT1       ;2By,1CM;En 500 ns el A/D listo para otra conversión
        nop             ;1By,1CM;
        nop             ;1By,1CM;
        nop             ;1By,1CM;
        nop             ;1By,1CM;
        nop             ;1By,1CM;
        nop             ;1By,1CM;
        nop             ;1By,1CM;
MUESTRA2:
        nop             ;1By,1CM;
        nop             ;1By,1CM;
        nop             ;1By,1CM;
        nop             ;1By,1CM;
        nop             ;1By,1CM;
        nop             ;1By,1CM;
MUESTRA:
        clr INT1        ;2By,1CM;Disparo el A/D
        movx @DPTR,A    ;1By,2CM;Guardo en memoria la muestra anterior
        inc DPTR        ;1By,2CM;Incremento el puntero
        mov A,P1        ;2By,1CM;Leo el valor de la conversión actual
        setb INT1       ;2By,1CM;En 500 ns el A/D listo para otra conversión

        mov 0H,DPL              ;3By,2CM;En R0 la parte baja de DPTR
        xrl 0H,#FCH             ;3By,2CM;R0 xor #FC, si son iguales => R0 = 0
        cjne R0,#0,MUESTRA2     ;3By,2CM;If ( R0 != 0 ) goto MUESTRA

        mov 0H,DPH              ;3By,2CM;En R0 la parte baja de DPTR
        xrl 0H,#83H             ;3By,2CM;R0 xor #83, si son iguales => R0 = 0
        cjne R0,#0,MUESTRA      ;3By,2CM;If ( R0 != 0 ) goto MUESTRA

        ljmp FSAMPLER
                        ;CMtotal = 1+2+2+1+1+2+2+2+2+2+2 = 19 CM
;----------------------Fin sin multiplexación-------------------------------
```

```
MUX1Y2:
;------------------Con multiplexación de Canales 1 y 2----------------------
        mov A,31H       ;Cargo la copia de BCON1
        mov DPTR,#0000H ;Dirección del BCON
        movx @DPTR,A    ;Actúo sobre la electrónica con BCON1

        mov DPTR,#8008H ;Dirección inicial para los datos

        clr INT1        ;2By,1CM;Disparo el A/D
        nop             ;1By,1CM;
        nop             ;1By,1CM;
        nop             ;1By,1CM;Para esperar los 2.5 us
        mov A,P1        ;2By,1CM;Leo el valor de la conversión
        setb INT1       ;2By,1CM;En 500 ns el A/D listo para otra conversión

        nop             ;1By,1CM;
        mov R7,#01H     ;2By,1CM; Retardo de 2 * 10 + 1 CM
RETARDO2:
        nop             ;1By,1CM;
        nop             ;1By,1CM;
        nop             ;1By,1CM;
        nop             ;1By,1CM;
        nop             ;1By,1CM;
        nop             ;1By,1CM;
        nop             ;1By,1CM;
        nop             ;1By,1CM;
        djnz R7,RETARDO2   ;2By,2CM;
MUES2:
        nop             ;1By,1CM;
        nop             ;1By,1CM;
        nop             ;1By,1CM;
        nop             ;1By,1CM;
        nop             ;1By,1CM;
        nop             ;1By,1CM;
MUES:
        clr INT1        ;2By,1CM;Disparo el A/D
        movx @DPTR,A    ;1By,2CM;Guardo en memoria la muestra anterior

        cpl T0          ;1By,1CM;CA2 de T0
        jb T0,T0ES1     ;3By,2CM;
        mov A,31H       ;2By,1CM;T0 = 0, se refiere al Canal 1 => BCON1
        sjmp T0ES0      ;2By,2CM;
T0ES1:
        mov A,32H       ;2By,1CM;T0 = 1, se refiere al Canal 2 => BCON2
        nop             ;1By,1CM;
        nop             ;1By,1CM;
T0ES0:
        mov R1,DPL      ;2By,2CM;Guardo el DPTR
        mov R2,DPH      ;2By,2CM;
        mov DPTR,#0000H ;2By,2CM;@ de BCON
        movx @DPTR,A    ;1By,2CM;Actúo sobre la electrónica
        mov DPL,R1      ;2By,2CM;Restauro el DPTR
        mov DPH,R2      ;2By,2CM

        inc DPTR        ;1By,2CM;Incremento el puntero
        mov A,P1        ;2By,1CM;Leo el valor de la conversión actual
        setb INT1       ;2By,1CM;En 500 ns el A/D listo para otra conversión

        mov 0H,DPL          ;3By,2CM;En R0 la parte baja de DPTR
        xrl 0H,#FCH         ;3By,2CM;R0 xor #FC, si son iguales => R0 = 0
        cjne R0,#0,MUES2    ;3By,2CM;If ( R0 != 0 ) goto MUESTRA

        mov 0H,DPH          ;3By,2CM;En R0 la parte baja de DPTR
        xrl 0H,#83H         ;3By,2CM;R0 xor #83, si son iguales => R0 = 0
        cjne R0,#0,MUES     ;3By,2CM;If ( R0 != 0 ) goto MUESTRA

        ljmp FSAMPLER
        ;CMtotal = 1+2+1+2+1+2+2+2+2+2+2+2+2+1+1+2+2+2+2+2+2 = 34 CM
;------------------Con multiplexación de Canales 1 y 2----------------------

FSAMPLER:
;-----------------------------------------------------------------------------
;---    Fin MUESTREAR Y GUARDAR EN MEMORIA 1012 MUESTRAS                   ---
;-----------------------------------------------------------------------------
```

```
;-----------------------------Trama final-----------------------------------
        mov A,#04H
        movx @DPTR,A
        inc DPTR        ;83FD H
        mov A,#FBH
        movx @DPTR,A
        inc DPTR        ;83FE H
        mov A,#04H
        movx @DPTR,A
        inc DPTR        ;83FF H
        mov A,#FBH
        movx @DPTR,A
;-----------------------------Fin Trama final-------------------------------

;-------------------------------------------------------------------------
;---    LECTURA DE MEMORIA Y TRANSMISION POR EL CANAL SERIE DE 1024 DATOS
;-------------------------------------------------------------------------
        mov DPTR,#8000H ;Puntero a @InicialMemoria
        movx A,@DPTR    ;Leo de memoria una muestra
        inc DPTR
        ljmp TRANS
                        ;El gran cuello de botella
ESPERA:                 ;Espera activa hasta fin de transmisión
        jnb TI,ESPERA   ;If ( TI == 0 ) then goto ESPERA
        clr TI          ;Reset por soft del flag de fin de transmisión
TRANS:
        mov SBUF,A              ;Al finalizar la transmisión se pone TI = 1 por hard

        movx A,@DPTR            ;Leo de memoria una muestra
        inc DPTR                ;1By,2CM;Incremento el puntero

        mov 0H,DPL              ;3By,2CM;En R0 la parte baja de DPTR
        xrl 0H,#01H              ;3By,2CM;R0 xor #0H, si son iguales => R0 = 0
        cjne R0,#0,ESPERA       ;3By,2CM;If ( R0 != 0 ) goto ESPERA

        mov 0H,DPH              ;3By,2CM;En R0 la parte baja de DPTR
        xrl 0H,#84H             ;3By,2CM;R0 xor #84, si son iguales => R0 = 0
        cjne R0,#0,ESPERA       ;3By,2CM;If ( R0 != 0 ) goto ESPERA
;-------------------------------------------------------------------------
;--- fin LECTURA DE MEMORIA Y TRANSMISION POR EL CANAL SERIE DE 1024 DATOS
;-------------------------------------------------------------------------

        ljmp INFINITO

;-------------------------------------------------------------------------
;---    SERVICIO A LA INTERRUPCION DE COMUNICACIONES                    ---
;-------------------------------------------------------------------------
                        ;¡¡¡ R0 esta ocupado !!!
                        ;¡¡¡ R1 esta ocupado !!!
                        ;¡¡¡ R2 esta ocupado !!!
SERVCOM:
        jb RI,SERVRI
                        ;Es una transmisión
        reti

SERVRI:                 ;Es una recepción
        clr RI          ;La petición se debe limpiar por hard

;---------------------Salvar contexto--------------------------
        mov R3,A        ;La rutina utiliza R3 para salvar A
                        ;LA rutina utiliza R4 para guardar SBUF
;---------------------Fin Salvar contexto----------------------
```

```
;-------------------------------------------------------------------------
;-----------------------INICIO DECODIFICACION-----------------------------

        mov A,SBUF      ;Leo el dato recibido
        mov R4,A        ;Guardo la recepción, evito error si recepción

;----------Común para todos los canales-----------------------------------
        mov 2AH,#00H    ;Reset mascara 0s de BC
        clr C           ;Reset del Carry

        rrc A           ;Obtengo AC/DC!
        jc ACDCES1
        clr 2AH.0       ;Mascara 0s, AC/DC! = 0
YGAN3:
        rrc A           ;Obtengo G3
        jc GAN3ES1
        clr 2AH.1       ;Mascara 0s, G3 = 0
YGAN2:
        rrc A           ;Obtengo G2
        jc GAN2ES1
        clr 2AH.2       ;Mascara 0s, G2 = 0
YGAN1:
        rrc A           ;Obtengo G1
        jc GAN1ES1
        clr 2AH.3       ;Mascara 0s, G1 = 0
        ljmp FINCOMUN

ACDCES1:
        setb 2AH.0      ;Mascara 0s, AC/DC! = 1
        sjmp YGAN3
GAN3ES1:
        setb 2AH.1      ;Mascara 0s, G3 = 1
        sjmp YGAN2
GAN2ES1:
        setb 2AH.2      ;Mascara 0s, G2 = 1
        sjmp YGAN1
GAN1ES1:
        setb 2AH.3      ;Mascara 0s, G1 = 1
        sjmp FINCOMUN

FINCOMUN:
        mov 29H,2AH     ;29H será la mascara de 1s de BC
        orl 29H,#F0H    ;(29H) = 1-1-1-1-G1-G2-G3-AC/DC!. Mascara 1s
                        ;(2AH) = 0-0-0-0-G1-G2-G3-AC/DC!. Mascara 0s
                        ;La mascara de 1s pondrá los ceros
                        ;La mascara de 0s pondrá los unos
;----------Fin Comun para todos los canales-----------------------------------

        mov A,R4        ;Cargo el valor de recepción

        clr C           ;Reset del Carry

        rlc A           ;Obtengo CH1
        jc CH1ES1

        rlc A
        jc CH0ES1X       ;Obtengo CH0
        sjmp SON0
CH0ES1X:
        ljmp CH0ES1
SON0:
;--------CH1, CH0 = 0----------
        rlc A           ;Obtengo E
        jc ENABLE1
```

```
;.....................Modo manual.........................................
                      ;CH1, CH0, E = 0
        rlc A
        jc MANUAL
        clr 20H.2       ;No manual
        ljmp FINRX
MANUAL:
        setb 20H.2      ;Manual
        rlc A
        jc FUEGO
        clr 20H.3       ;No existe disparo
        ljmp FINRX
FUEGO:
        setb 20H.3      ;Disparo ON
        ljmp FINRX
;.....................Fin Modo manual.....................................

ENABLE1:
;---------------------------- CANAL 1 ---------------------------------------
        mov 25H,R4      ;¡ Actualizo el BE1 !

        mov R5,A        ;Guardo A
        mov A,21H       ;Cargo (21H)
        anl A,29H       ;Actualizo la parte alta de BC1
        orl A,2AH       ;Mediante las mascaras de 1s y 0s
        mov 21H,A       ;Cambio (21H)
        mov A,R5

        rlc A           ;Obtengo FAC1!
        jc FAC1ES1
        clr 21H.4       ;BC1, FAC1! = 0
        clr 22H.4       ;BC2, FAC1! = 0
        clr 23H.4       ;BC3, FAC1! = 0
        clr 24H.4       ;BC4, FAC1! = 0
        ljmp FINRX
FAC1ES1:
        setb 21H.4      ;BC1, FAC1! = 1
        setb 22H.4      ;BC2, FAC1! = 1
        setb 23H.4      ;BC3, FAC1! = 1
        setb 24H.4      ;BC4, FAC1! = 1
        ljmp FINRX
;---------------------------- Fin CANAL 1 ----------------------------------

;----------CH1 = 1.-----------
CH1ES1:
        rlc A           ;Obtengo CH0
        jnc CH0ES0
;-----CH1 = 1, CH0 = 1.-------

;---------------------------- CANAL 4 ---------------------------------------
        mov 28H,R4      ;¡ Actualizo el BE4 !

        mov R5,A        ;Guardo A
        mov A,24H       ;Cargo (24H)
        anl A,29H       ;Actualizo la parte baja de BC4
        orl A,2AH       ;Mediante las mascaras de 1s y 0s
        mov 24H,A       ;Cambio (24H)
        mov A,R5

        rlc A           ;Obtengo E
        jc ENABLE4

;------------------DESHABILITADOS Canales 2, 3 y 4-------------------------
        setb 20H.0      ;Sin multiplexación
        clr 20H.1       ;Solo canal 1
        sjmp CANAL4
;------------------Fin DESHABILITADOS Canales 2 ,3 y 4--------------------



;------------------HABILITADOS Canales 2, 3 y 4-----------------------
ENABLE4:
        clr 20H.0       ;multiplexación de Canales 1, 2, 3 y 4
        clr 20H.1       ;Mediante T0 y T1
;------------------Fin HABILITADOS Canales 2 ,3 y 4-----------------------
```

```
CANAL4:
        rlc A
        jc FAC4ES1
        clr 21H.7       ;BC1, FAC4! = 0
        clr 22H.7       ;BC2, FAC4! = 0
        clr 23H.7       ;BC3, FAC4! = 0
        clr 24H.7       ;BC4, FAC4! = 0
        sjmp FINRX
FAC4ES1:
        setb 21H.7       ;BC1, FAC4! = 1
        setb 22H.7       ;BC2, FAC4! = 1
        setb 23H.7       ;BC3, FAC4! = 1
        setb 24H.7       ;BC4, FAC4! = 1
        sjmp FINRX
;---------------------------- Fin CANAL 4 ----------------------------------

;-------CH1 = 1, CH0 = 0.----------
CH0ES0:
        rlc A           ;Obtengo E
        jc ENABLE3

;..................Con frecuencia variable.................................
                        ;CH1 = 1, CH0 = 0, E = 0
        sjmp FINRX
;..................Fin Con frecuencia variable.............................

ENABLE3:
;---------------------------- CANAL 3 ----------------------------------
        mov 27H,R4      ;¡ Actualizo el BE3 !

        mov R5,A        ;Guardo A
        mov A,23H       ;Cargo (23H)
        anl A,29H       ;Actualizo la parte baja de BC3
        orl A,2AH       ;Mediante las mascaras de 1s y 0s
        mov 23H,A       ;Cambio (23H)
        mov A,R5

        rlc A
        jc FAC3ES1
        clr 21H.6       ;BC1, FAC3! = 0
        clr 22H.6       ;BC2, FAC3! = 0
        clr 23H.6       ;BC3, FAC3! = 0
        clr 24H.6       ;BC4, FAC3! = 0
        sjmp FINRX
FAC3ES1:
        setb 21H.6       ;BC1, FAC3! = 1
        setb 22H.6       ;BC2, FAC3! = 1
        setb 23H.6       ;BC3, FAC3! = 1
        setb 24H.6       ;BC4, FAC3! = 1
        sjmp FINRX
;---------------------------- Fin CANAL 3 ----------------------------------
```

```
;---------CH1 = 0, CH0 = 1.---------

CH0ES1:
;--------------------------- CANAL 2 ----------------------------------------
        mov 26H,R4      ;¡ Actualizo el BE2 !

        mov R5,A        ;Guardo A
        mov A,22H       ;Cargo (22H)
        anl A,29H       ;Actualizo la parte baja de BC2
        orl A,2AH       ;Mediante las mascaras de 1s y 0s
        mov 22H,A       ;Cambio (22H)
        mov A,R5


        rlc A           ;Obtengo E
        jc ENABLE2

;---------------------INHABILITADO Canal 2-----------------------------------
        setb 20H.0      ;Sin multiplexación
        clr 20H.1       ;Solo canal 1
        sjmp CANAL2
;---------------------Fin INHABILITADO Canal 2---------------------------------

;---------------------HABILITADO Canal 2-------------------------------------
ENABLE2:
        clr 20H.0       ;Modo de multiplexación del Canal 1 y 2
        setb 20H.1      ;Mediante T0 y T1
;---------------------HABILITADO Canal 2-------------------------------------

CANAL2:
        rlc A
        jc FAC2ES1
        clr 21H.5       ;BC1, FAC2! = 0
        clr 22H.5       ;BC2, FAC2! = 0
        clr 23H.5       ;BC3, FAC2! = 0
        clr 24H.5       ;BC4, FAC2! = 0
        sjmp FINRX
FAC2ES1:
        setb 21H.5       ;BC1, FAC2! = 1
        setb 22H.5       ;BC2, FAC2! = 1
        setb 23H.5       ;BC3, FAC2! = 1
        setb 24H.5       ;BC4, FAC2! = 1
        sjmp FINRX
;--------------------------- Fin CANAL 2 ------------------------------------

FINRX:

;------------------Restaurar contexto-----------------------
        mov A,R3
;------------------Fin Restaurar contexto-------------------

        reti
;-------------------------------------------------------------------------------
;---    FIN SERVICIO A LA INTERRUPCION DE COMUNICACIONES            ---
;-------------------------------------------------------------------------------

        .end
```

6.2 CÓDIGO DE ALTO NIVEL

6.2.1 CLASE CRS232

```
////////////////////////////////////////////////////////////////////////////
//                                RS232v3.h                                //
////////////////////////////////////////////////////////////////////////////
// Clase creada por Ernest Gil y modificada por Raúl Bartolomé             //
//                                                                          //
////////////////////////////////////////////////////////////////////////////
// 1- Al rebre un char EVENT genera una crida a una funcio ext. predefinida
//    amb un temps invertit de 1,5 milisegons si el Thread te la prioritat
//    THREAD_PRIORITY_TIME_CRITICAL
// 2- Exemple de creació:
//
//     void (*pOnRxChar)(int,CRS232*)= &OnRxChar;  // Def. punter pOnRxChar
//     CRS232* m_pRs232 =    new CRS232(1024,1024);// Creació obj RS232 i punter
//     CWinThread* m_pThread1;                     // Declaracio Thread
//     //-----------------------------------------------------
//     UINT Thread1(LPVOID pParam)
//     {
//     CRS232* punter_Rs232 = (CRS232*) pParam;
//
//     punter_Rs232->Thread_Att_RS232();
//     return 0;
//     }
//     //-----------------------------------------------------
//     ...
//     m_pThread1=AfxBeginThread(    //posta en marxa del thread
//                                   Thread1,
//                                   m_pRs232,
//                                   THREAD_PRIORITY_TIME_CRITICAL);
//     ...
//     //-----------------------------------------------------
//     ...
//     //-----------------------------------------------------
//     void OnRxChar(int NumBytes, CRS232* pRs232) // funcio d'atencio RX
//     {
//     int            x;
//     BYTE    pCadena[1024];
//     BYTE    pHola[10];
//
//     pRs232->RXcopia(pCadena);
//     for (x=0;x<NumBytes;x++) {
//            ::PostMessage(hPantalla,WM_DADA_RX,pCadena[x],0);
//            }
//     }
//     //-----------------------------------------------------
//     void OnEspiaTx(int NumBytes, BYTE* buf) // funcio d'atencio Espia TX
//     {      // bytes a punt d'enviar
//     ...
//     }
//     //-----------------------------------------------------
//     void OnEspiaRx(int NumBytes, BYTE* buf) // funcio d'atencio Espia RX
//     {      // bytes acabats de rebre
//     ...
//     }
//     //-----------------------------------------------------
////////////////////////////////////////////////////////////////////////////

#include "winbase.h"



// Constants A DEFINIR EN CADA APLICACIO
#define XON           0x11
#define XOFF          0x13
#define EOT           0x04
#define ESC           0x1b
#define XON_S         0xff
#define XOFF_S 0xfe
#define EOT_S         0xfd
#define ESC_S         0xfc
```

```
#define EVENT         0x04

#define WM_ERROR_ACCES_PORT  WM_USER+10             // missatge

//////////////////////////////////////////////////////////////////////
class CRS232
{
public:
        // Constructor ..........................................................
        CRS232(int LBufRX, int LBufTX);
        // Funcions de suport ...................................................
        void    Thread_Att_RS232();                             // Thread de control RX del
RS232
        // Funcions normals .....................................................
        void    ConfigurarPort(
                                int             Port,           // 0=Com1,1=Com2,....
                                int             Baudis, // 0=1k2,..,4=19k2,5=38k4,6=115k
                                int             BitsCar,        //
0=4bits,1=5b,2=6b,3=7b,4=8bits
                                int             BitsParada,     // 0=1bit,1=1.5bits,2=2bits
                                int             Paritat,        //
0=No,1=Par,2=Senar,3=Mar,4=Esp
                                int             ControlFlux,    // 0=No,1=DTR,2=RTS,3=Xon
                                HWND    hFinestra,              // finestra a enviar
missatges
                                void    (*pRxApli)(int,CRS232*));// adreça aten. apli Tx
        BOOL    ObrirPort();
        void    TancarPort();
        BOOL    TX(BYTE* pCad,int bytes);                       // Transmitir una cadena
        BOOL    TXacabat();                                     // Torna TRUE si ha acabat TX
        int     RXnumBytes();                                   // Torna Numero bytes rebuts
        int     RXcopia(BYTE* pCad);                // Copiar el buffer de recepcio
                                                                // Torna: Numero bytes rebuts
        void    EspiaHab(void (*pEspTx)(int, BYTE*), // adreça aten. esp Tx
                        void (*pEspRx)(int, BYTE*));  //adreça aten. esp Rx
        void    EspiaDeshab();

        // Dades
        BOOL    PortObert;
        HWND    DestiMsg;                       // handler Window per enviar missat WM_..
private:        //.....................................................
        // Funcions
        BOOL    CarregarDCB();
        void    (*pRutinaAtencio)(int,CRS232*);             // Avís de recepcio RX
        void    (*pEspAtenTX)(int, BYTE*);                  // Avís espia d'emissio TX
        void    (*pEspAtenRX)(int, BYTE*);                  // Avís espia de recepcio RX
        // Dades
        CRS232* pAquestObjecte;                 // punter a aquest objecte
        HANDLE  m_idDisCom;                             // Handler del dispositiu
COMx
        UINT    m_wTablaBaudios[6];                     // Taula de baudis
        BYTE    m_TablaParidad[4];                      // Taula de paritat
        BYTE    m_TablaBitsParada[2];           // Taula de bits de parada
        Int     m_nPort;                                // Config.
        int     m_nBaudis;                              // Config.
        int     m_nBitsCar;                             // Config.
        int     m_nBitsParada;                  // Config.
        int     m_nParitat;                             // Config.
        int     m_nControlFlux;                         // Config.
        int     LongBufRX;                              // Config.--Pasado al
constructor
        int     LongBufTX;                              // Config.--Pasado al
constructor
        BYTE*   pBufRX;                         // punter del buffer de recepció
        BYTE*   pBufTX;                 // punter del buffer de transmissió
        int     NumBytesBufRX;                  // Numero de bytes---leídos
        int     NumBytesBufTX;                  // Numero de bytes---enviados

        struct _COMSTAT             COMSTAT_1;
        // COMSTAT es una estructura de Win32 que contiene información sobre
        // una comunicación de dispositivo. Esta estructura es llenada por la función
        // ClearCommError. Algunas siglas:
        // CTS: Clear To Send
        // DSR: Data Set Ready
        // RLSD: Receive Line Signal Detect
        // EOF: End Of File
        /*
```

```
        typedef struct _COMSTAT { // cst
                DWORD fCtsHold : 1;   // Tx waiting for CTS signal
                DWORD fDsrHold : 1;   // Tx waiting for DSR signal
                DWORD fRlsdHold : 1;  // Tx waiting for RLSD signal
                DWORD fXoffHold : 1;  // Tx waiting, XOFF char rec'd
                DWORD fXoffSent : 1;  // Tx waiting, XOFF char sent
                DWORD fEof : 1;       // EOF character sent
                DWORD fTxim : 1;      // character waiting for Tx
                DWORD fReserved : 25; // reserved
                DWORD cbInQue;        // bytes in input buffer
                DWORD cbOutQue;       // bytes in output buffer
        } COMSTAT, *LPCOMSTAT;
        */
        struct _COMMTIMEOUTS  COMMTIMEOUTS_1;
        // La estructura COMMTIMEOUTS es usada con las funciones
        // SetCommTimeouts y GetCommTimeouts para establecer y preguntar los
        // parámetros time-out para una comunicación con dispositivo. Los
        // parámetros determinan el comportamiento de las operaciones ReadFile,
        // WriteFile, ReadFileEx y WriteFileEx en el dispositivo.
        /*
        typedef struct _COMMTIMEOUTS {   // ctmo
                DWORD ReadIntervalTimeout;
        // Especifica el máximo tiempo en milisegundos permitidos para
        // transmitir entre la llegada de dos carácteres en la línea de comunicación
                DWORD ReadTotalTimeoutMultiplier;
        // Especifica el multiplicador en milisegundos usado para calcular el periodo
        // time-out para las operaciones de lectura.
                DWORD ReadTotalTimeoutConstant;
        // Idem pero constante.
        // Los dos anteriores especifican que la operación de lectura vuelve
        // inmediatamente con el carácter que acaba de ser recibido,
        // incluso si el carácter no ha sido recibido
                DWORD WriteTotalTimeoutMultiplier;
        // Idem pero en operación de escritura.
                DWORD WriteTotalTimeoutConstant;
        } COMMTIMEOUTS,*LPCOMMTIMEOUTS;
        */
        HANDLE hEventEscritura;
        // Los HANDLER se utilizan en las aplicaciones de Windows para
        // referirse a un recurso que ha sido cargado en memoria
        OVERLAPPED overEscritura;
        // OVERLAPPED es una estructura de Win32 que contiene información usada
        // en entradas/salidas asincrónas
        /*
        typedef struct _OVERLAPPED { // o
                DWORD  Internal;              // Reservado para el sistema operativo
                DWORD  InternalHigh;  // Reservado para el sistema operativo
                DWORD  Offset;        // Especifica una posición de fichero
                                              // en la que empieza la transferencia
                DWORD  OffsetHigh;            // Especifica la palabra alta del byte
offset
                                              // que empieza la transferencia
                HANDLE hEvent;        // Identifica un evento estableciendo a estado
                                // señalizando cuando la transferencia ha sido completada.
        } OVERLAPPED;
        */
        BOOL    EspiaHabilitat;
};
```

```
////////////////////////////////////////////////////////////////////////////
//                      Implementacio Clase RS232                         //
////////////////////////////////////////////////////////////////////////////
///-------------------------------------------------------------------------

CRS232::CRS232(int LBufRX, int LBufTX)
{
        TRACE("En CRS232::CRS232(int LBufRX, int LBufTX)\n");
        // Punter a aquest objecte
        pAquestObjecte = this;

        // Port tancat
        PortObert=FALSE;
        EspiaHabilitat =FALSE;

        // buffers (construir i dimensionar)
        LongBufRX = LBufRX;
        LongBufTX = LBufTX;

        pBufRX = new BYTE[LongBufRX];
        pBufTX = new BYTE[LongBufTX];
        NumBytesBufRX= 0;
        NumBytesBufTX= 0;

        // overlapped
        hEventEscritura = CreateEvent(
                                    NULL,   // El handler no puede ser heredado
                                    TRUE,   // manual-reset event=>usar ResetEvent
                                    FALSE,  // flag for initial state
                                    NULL);  // pointer to event-object name. Sin nombre
        // La función CreateEvent crea un objeto evento con nombre o anónimo
        /*
        HANDLE CreateEvent(
        LPSECURITY_ATTRIBUTES lpEventAttributes, // pointer to security attributes
        // Puntero a una estructura SSCURITY_ATTRIBUTE que determina si el handle
        // retornado puede ser heredado por el proceso hijo. Si IpEventAttributes es
        // NULL, el handle no puede ser heredado
        BOOL bManualReset,    // flag for manual-reset event
        // Especifica si un objeto evento creado es manual-reset o auto-reset.
        // Si TRUE entonces debes usar la función ResetEvent para resetear
        // manualmenteel estado. Si FALSE, Windows automáticamente resetea
        BOOL bInitialState,  // flag for initial state
        // Especifica el estado inicial del objeto evento.
        // Si TRUE, el estado inicial es señalizado, en caso cantrario es no señalizado
        LPCTSTR lpName          // pointer to event-object name
        // Puntero a un string terminado con null especificando el nombre del objeto
        // evento. Si es NULL el objeto evento es creado sin un nombre
        );
        */
        overEscritura.Offset= 0;
        overEscritura.OffsetHigh=    0;
        overEscritura.hEvent= hEventEscritura;

        // Taules per a l'obrir el port
                // Tabla de baudios obtenida de la estructura DCB, miembro BaudRate
        m_wTablaBaudios[0]   = CBR_1200;
        m_wTablaBaudios[1]   = CBR_2400;
        m_wTablaBaudios[2]   = CBR_4800;
        m_wTablaBaudios[3]   = CBR_9600;
        m_wTablaBaudios[4]   = CBR_19200;
        m_wTablaBaudios[5]   = CBR_38400;
        m_wTablaBaudios[6]   = CBR_115200;
                // Tabla de paridad obtenida de la estructura DCB, miembro Parity
        m_TablaParidad[0]    = NOPARITY;
        m_TablaParidad[1]    = EVENPARITY;
        m_TablaParidad[2]    = ODDPARITY;
        m_TablaParidad[3]    = MARKPARITY;
        m_TablaParidad[4]    = SPACEPARITY;


                // Tabla de paridad obtenida de la estructura DCB, miembro StopBits
        m_TablaBitsParada[0] = ONESTOPBIT;
        m_TablaBitsParada[1] = ONE5STOPBITS;
        m_TablaBitsParada[2] = TWOSTOPBITS;
        /*
        // The DCB structure defines the control setting for
```

```
        // a serial communications device.
        typedef struct _DCB { // dcb
                DWORD DCBlength;           // sizeof(DCB)
                DWORD BaudRate;            // current baud rate
                DWORD fBinary: 1;          // binary mode, no EOF check
                DWORD fParity: 1;          // enable parity checking
                DWORD fOutxCtsFlow:1;      // CTS output flow control
                DWORD fOutxDsrFlow:1;      // DSR output flow control
                DWORD fDtrControl:2;       // DTR flow control type
                DWORD fDsrSensitivity:1;   // DSR sensitivity
                DWORD fTXContinueOnXoff:1; // XOFF continues Tx
                DWORD fOutX: 1;            // XON/XOFF out flow control
                DWORD fInX: 1;             // XON/XOFF in flow control
                DWORD fErrorChar: 1;       // enable error replacement
                DWORD fNull: 1;            // enable null stripping
                DWORD fRtsControl:2;       // RTS flow control
                DWORD fAbortOnError:1;     // abort reads/writes on error
                DWORD fDummy2:17;          // reserved
                WORD wReserved;            // not currently used
                WORD XonLim;               // transmit XON threshold
                WORD XoffLim;              // transmit XOFF threshold
                BYTE ByteSize;             // number of bits/byte, 4-8
                BYTE Parity;               // 0-4=no,odd,even,mark,space
                BYTE StopBits;             // 0,1,2 = 1, 1.5, 2
                char XonChar;              // Tx and Rx XON character
                char XoffChar;             // Tx and Rx XOFF character
                char ErrorChar;            // error replacement character
                char EofChar;              // end of input character
                char EvtChar;              // received event character
                WORD wReserved1;           // reserved; do not use
        } DCB;
        // BaudRate
        // Specifies the baud rate at which the communications device operates.
        // This member can be an actual baud rate value,
        // or one of the following baud rate indexes:
        // CBR_110    CBR_300    CBR_600    CBR_1200  CBR_2400  CBR_4800  CBR_9600
        // CBR_14400  CBR_19200  CBR_38400 CBR_56000 CBR_57600 CBR_115200
        // CBR_128000 CBR_256000

        // Parity
        // Specifies the parity scheme to be used.
        // This member can be one of the following values:
        // Value              Meaning
        // EVENPARITY         Even
        // MARKPARITY         Mark
        // NOPARITY           No parity
        // ODDPARITY          Odd
        // SPACEPARITY        Space

        // StopBits
        // Specifies the number of stop bits to be used.
        // This member can be one of the following values:
        // Value              Meaning
        // ONESTOPBIT         1 stop bit
        // ONE5STOPBITS       1.5 stop bits
        // TWOSTOPBITS 2 stop bits
        */
}
```

```
///--------------------------------------------------------------------------
void CRS232::ConfigurarPort(
                      int    Port,
                      int    Baudis,
                      int    BitsCar,
                      int    BitsParada,
                      int    Paritat,
                      int    ControlFlux,
                      HWND   hFinestra,
                      void   (*pRxApli)(int,CRS232*))      // adreça aten. apli Tx
{
       // Parametres port
       m_nPort=              Port;
       m_nBaudis=            Baudis;
       m_nBitsCar=           BitsCar;
       m_nBitsParada= BitsParada;
       m_nParitat=           Paritat;
       m_nControlFlux=       ControlFlux;
       // handler per a l'emissio de missatges WM_...
       DestiMsg = hFinestra;
       // adreça funciós d'atencio
       pRutinaAtencio = pRxApli;
}
///--------------------------------------------------------------------------
BOOL CRS232::ObrirPort()
{
       char   szPort[10];
       BOOL   a;

       // Formar la cadena "COMx"
       wsprintf(szPort, "COM%d", m_nPort + 1);
       // La función wsprintf formatea y almacena una serie de caracteres
       // y valores en un buffer

       // Obrir el port de comunicacions
       m_idDisCom=CreateFile(
              szPort,                        // COM1 ó COM2 ...
              GENERIC_READ|GENERIC_WRITE,    // RD/WR
              0,                                    // obert en mode exclussiu
              NULL,                                 // Sense atributs de seguretat
              OPEN_EXISTING,                 // Comm -> sempre així
              FILE_FLAG_OVERLAPPED,          // Overlapped(assíncron en background)
              NULL);                                // Comm -> sempre així
       // La funsión CreateFile crea o abre los siguientes objetos y devuelve un
       // handle que puede ser usado para acceder al objeto:
       //· files
       //· pipes
       //· mailslots
       //· communications resources
       //· disk devices (Windows NT only)
       //· consoles
       //· directories (open only)
       /*
       HANDLE CreateFile(
       LPCTSTR lpFileName,                           // pointer to name of the file
       DWORD dwDesiredAccess,                               // access (read-write) mode
       DWORD dwShareMode,                            // share mode
       LPSECURITY_ATTRIBUTES lpSecurityAttributes,          // pointer to security
attributes
       DWORD dwCreationDistribution,                        // how to create
       // OPEN_EXISTING Opens the file. The function fails if the file does not exist.
       DWORD dwFlagsAndAttributes,                   // file attributes
       // FILE_FLAG_OVERLAPPED
       // Informa al sistema para inicializar el objeto, así esa operación que toma
       // una significante cantidad de tiempo para procesar el retorno
       // ERROR_IO_PENDING.
       // Cuando la operación está finalizada, el evento especificado es establecido a
       // estado señalizado. Cuando especificas FILE_FLAG_OVERLAPPED, las funciones
       // ReadFile y WriteFile deben especificar una estructura OVERLAPPED.
       // Que es, cuando FILE_FLAG_OVERLAPPED está especificado, una aplicación debe

       // caracterizar lectura y escritura solapada (overlapped).
       // Esta bandera también habilita más de una operación para ser
       // caracterizada simultáneamente con un handle (una operación de lectura y
       // escritura simltanea, por ejemplo).
       HANDLE hTemplateFile          // handle to file with attributes to copy
       );
```

```
        */

        // Si es detecta error a l'obrir
        if (m_idDisCom==INVALID_HANDLE_VALUE) return FALSE;

        // Buffers de Win95
        a = SetupComm(m_idDisCom, 1024/*bufIn*/, 1024/*bufOut*/);
        // La función SetupComm inicializa los parámetros de comunicaciones
        // para la comunicación específica de un dispositivo.

        if (a==FALSE) return FALSE;

        // Carregar paràmetres al DCB
        if (!CarregarDCB()) return FALSE;

        // Establir Timeouts
        COMMTIMEOUTS_1.ReadIntervalTimeout=         MAXDWORD;
        COMMTIMEOUTS_1.ReadTotalTimeoutMultiplier=  0;
        COMMTIMEOUTS_1.ReadTotalTimeoutConstant=    0;
        COMMTIMEOUTS_1.WriteTotalTimeoutMultiplier= 0;
        COMMTIMEOUTS_1.WriteTotalTimeoutConstant=   0;
        SetCommTimeouts(m_idDisCom, &COMMTIMEOUTS_1);
        // Esta función establece los parámetros time-out para todas las operaciones de
        // lectura y escritura en un dispositivo de comunicaciones específico.
        /*
        BOOL SetCommTimeouts(
        HANDLE hFile,              // handle of communications device
        LPCOMMTIMEOUTS lpCommTimeouts  // address of communications time-out structure
        );
        */

        // Indicador per al thread y funcions externes
        PortObert=TRUE;

        return TRUE;
}
///-------------------------------------------------------------------------
BOOL CRS232::CarregarDCB()
{
        DCB            dcb;

        GetCommState(m_idDisCom, &dcb);
        // Esta función llena en un bloque de control-dispositivo (una estructura DCB)
        // con el control actual para un dispositivo de comunicaciones concreto.
        /*
        BOOL GetCommState( HANDLE hFile, // handle of communications device
                LPDCB lpDCB                  // address of device-control block structure
                );
        */

        dcb.BaudRate    = m_wTablaBaudios[m_nBaudis];
        dcb.Parity      = m_TablaParidad[m_nParitat];
        dcb.ByteSize    = (BYTE)(m_nBitsCar + 4);
        dcb.StopBits    = m_TablaBitsParada[m_nBitsParada];

        dcb.fBinary     = TRUE;
        dcb.fParity     = TRUE;
        dcb.fErrorChar= FALSE;
        dcb.EvtChar     = EVENT;
        dcb.EofChar     = EOT;




        // Control de flux
        switch(m_nControlFlux) {
        case (0):                       // Sense ==> Sortides: DTR, RTS disable
                dcb.fDsrSensitivity = FALSE;
                dcb.fDtrControl       = DTR_CONTROL_DISABLE;
                dcb.fRtsControl       = RTS_CONTROL_DISABLE;
                dcb.fOutxDsrFlow      = FALSE;
                dcb.fOutxCtsFlow      = FALSE;
                break;
        case (1):                       // DTR/DSR ==> Sortides: RTS=DISABLE, No importa
CTS.
                dcb.fDsrSensitivity = TRUE;
                dcb.fDtrControl       = DTR_CONTROL_HANDSHAKE;
```

```
                dcb.fRtsControl       = RTS_CONTROL_DISABLE;
                dcb.fOutxDsrFlow      = TRUE;
                dcb.fOutxCtsFlow      = FALSE;
                break;
        case (2):                     // RTS/CTS ==> Sortides: DTR=DISABLE, No importa
DSR.
                dcb.fDsrSensitivity = FALSE;
                dcb.fDtrControl       = DTR_CONTROL_DISABLE;
                dcb.fRtsControl       = RTS_CONTROL_HANDSHAKE;
                dcb.fOutxDsrFlow      = FALSE;
                dcb.fOutxCtsFlow      = TRUE;
                break;
        case (3):                     // Control de flux software Xon/Xoff
                dcb.fInX              = TRUE;
                dcb.fOutX             = TRUE;
                dcb.XonChar           = XON;
                dcb.XoffChar          = XOFF;
                dcb.XonLim            = 20;
                dcb.XoffLim           = 20;
                dcb.fDsrSensitivity = FALSE;
                dcb.fDtrControl       = DTR_CONTROL_DISABLE;
                dcb.fRtsControl       = RTS_CONTROL_DISABLE;
                dcb.fOutxDsrFlow      = FALSE;
                dcb.fOutxCtsFlow      = FALSE;
                break;
                }

        if(SetCommState(m_idDisCom, &dcb) < 0) return FALSE;
        return TRUE;
}
///-------------------------------------------------------------------------
void CRS232::TancarPort()
{

        // Indicador per al thread y funcions externes
        PortObert = FALSE;
        // Tancar handle
        CloseHandle(m_idDisCom);             // El thread es dona compte
        // Esta función cierra un handle de objeto abierto
}
///-------------------------------------------------------------------------
void CRS232::Thread_Att_RS232()
{
        #define LON_BUF_RX232_TEMP    1024
        OVERLAPPED     over1;
        HANDLE         hEvent1;
        BOOL           bitResult;
        DWORD          dwResult;
        DWORD          BytesLlegits,events232;
        int            Bytes,x;
        BYTE           pBufRxTemp[LON_BUF_RX232_TEMP];
```

```
        // Event
        hEvent1 = CreateEvent(
                                    NULL,   // pointer to security attributes
                                    TRUE,   // manual-reset event
                                    FALSE,  // flag for initial state
                                    NULL);  // pointer to event-object name
        // overlapped
        over1.Offset=         0;
        over1.OffsetHigh=     0;
        over1.hEvent=         hEvent1;

        while(TRUE) {//-------------------------------------------------
                if (PortObert==FALSE) return;

                // Preparar màscara event.............................
                bitResult= SetCommMask(m_idDisCom, EV_RXCHAR);
                // Esta función especifica un juego de eventos para ser monitorizado
                // para una comunicación de dispositivo.
                /*
                BOOL SetCommMask( HANDLE hFile,  // handle of communications device
                        DWORD dwEvtMask              // mask that identifies enabled
events
                        );
                EV_RXCHAR  A character was received and placed in the input buffer.
                */

                if (bitResult==FALSE) {
                        dwResult=GetLastError();
                        if (dwResult != ERROR_IO_PENDING) {
                                ::PostMessage(DestiMsg,WM_ERROR_ACCES_PORT,1,dwResult);
                                return;
                                }
                        }

                // Indicar espera d'event.............................
                bitResult= WaitCommEvent(
                                    m_idDisCom,    // handle dispos. comunicacions
                                    &events232,    // adreça per events rebuts
                                    &over1);       // adreça estruct. overlapped
                // Esta función espera a un evento que ocurre al dispositivo
                // especificado de comunicaciones. El juego de eventos que son
                // monitorizados por esta función está contenido en la máscara
                // de eventos asociados con el dispositivo handle.
                /*
                BOOL WaitCommEvent( HANDLE hFile, // handle of communications device
                        LPDWORD lpEvtMask,  // address of variable for event that occurred
                        LPOVERLAPPED lpOverlapped,  // address of overlapped structure
                        );
                */

                if (bitResult==FALSE) {
                        dwResult=GetLastError();
                        if (dwResult != ERROR_IO_PENDING) {
                                ::PostMessage(DestiMsg,WM_ERROR_ACCES_PORT,2,dwResult);
                                return;
                                }
                        }

                // Iniciar espera .....................................
                dwResult= WaitForSingleObject(hEvent1,INFINITE);

                ResetEvent(hEvent1);
                if ((dwResult!= WAIT_OBJECT_0)||(PortObert==FALSE)) {
                        return;
                        }




                // Determinar numero de bytes rebuts ..................
                bitResult= ClearCommError(m_idDisCom, &dwResult, &COMSTAT_1);

                if (bitResult==FALSE) {
                        ::PostMessage(DestiMsg,WM_ERROR_ACCES_PORT,3,dwResult);
                        return;
```

```
        }

    // Bytes a llegir (observar posible overflow).............
    Bytes=COMSTAT_1.cbInQue;
    if (Bytes > LON_BUF_RX232_TEMP) {
        ::PostMessage(DestiMsg,WM_ERROR_ACCES_PORT,5,0);
        return;
        }

    // Llegir bytes...........................................
    bitResult = ReadFile(
                        m_idDisCom,           // handle al port
                        pBufRxTemp,           // buffer
                        Bytes,                // bytes a llegir
                        &BytesLlegits,
                        &over1);              // adreça estruct. overlapped
    // Espia..............................
    if (EspiaHabilitat==TRUE) {
        (*pEspAtenRX)(BytesLlegits, pBufRxTemp);
        }
    // Tractament buffer.....................................

    if (bitResult==TRUE) {// buscar un EOF
        for (x=0; x<(int)BytesLlegits; x++) {
            if (x == (int)(BytesLlegits - 1)) {
                pBufRX[NumBytesBufRX++]=pBufRxTemp[x];
                                // guarda último Byte
                if (pRutinaAtencio!=0) {
                                // si existeix rutina d'atencio:
                (*pRutinaAtencio)(NumBytesBufRX,pAquestObjecte);
                        }
                NumBytesBufRX=0;
            }
            else {
                pBufRX[NumBytesBufRX++]=pBufRxTemp[x];
                if (NumBytesBufRX>LongBufRX) {
                ::PostMessage(DestiMsg,WM_ERROR_ACCES_PORT,5,0);
                        return;
                        }
            }
        }
    }
    else {
        dwResult=GetLastError();
        if (dwResult != ERROR_IO_PENDING) {
            ::PostMessage(DestiMsg,WM_ERROR_ACCES_PORT,4,dwResult);
            return;
            }
        }
    //.....................................................
    }
}
```

```
///--------------------------------------------------------------------------
BOOL CRS232::TX(BYTE* pCadena,int NumBytes/*,BOOL transparencia*/)
{
    BOOL       result;
       DWORD  bytes_escritos,error;
       int            x;

       // Copiar buffer.....................
       for (x=0;x<NumBytes;x++) pBufTX[x]= pCadena[x];
              NumBytesBufTX= NumBytes;

       // Espia.............................
       if (EspiaHabilitat==TRUE) {
              (*pEspAtenTX)(NumBytesBufTX, pBufTX);
              }

       // Escriure al RS232.................
       result = WriteFile(   m_idDisCom,
                                          pBufTX,
                                          NumBytesBufTX,
                                          &bytes_escritos,
                                          &overEscritura);
       if (result==FALSE) {
              error=GetLastError();
              if (error==ERROR_IO_PENDING) result=TRUE;
              }
       return result;
}
///--------------------------------------------------------------------------
int CRS232::RXnumBytes()
{
    return NumBytesBufRX;
}
///--------------------------------------------------------------------------
int CRS232::RXcopia(BYTE* pCadena)
{
       int            x;

       for (x=0;x<NumBytesBufRX;x++) pCadena[x]= pBufRX[x];

       return NumBytesBufRX;
}
///--------------------------------------------------------------------------
BOOL CRS232::TXacabat()
{
       DWORD  dwResult;

       dwResult= WaitForSingleObject(hEventEscritura,0/*retorn inmediat*/);
       ResetEvent(hEventEscritura);
       //     WAIT_OBJECT_0  The state of the specified object is signaled.
       //     WAIT_TIMEOUT   The time-out elapsed, and the event is nonsignaled.
       //     WAIT_ABANDONED The specified object that was not released by the
       //     thread has terminated. The object is nonsignaled.
       if (dwResult==WAIT_OBJECT_0) return TRUE;    // enviament acabat
       return FALSE;
}
///--------------------------------------------------------------------------
void CRS232::EspiaHab(void (*pEspTx)(int, BYTE*),// adreça aten. esp Tx
                      void (*pEspRx)(int, BYTE*))// adreça aten. esp Rx
{
       pEspAtenTX = pEspTx;
       pEspAtenRX = pEspRx;
       EspiaHabilitat = TRUE;
}
///--------------------------------------------------------------------------
void CRS232::EspiaDeshab()
{
       EspiaHabilitat = FALSE;
}
```

6.2.2 CLASE CSCOPEDOC

```
// scopeDoc.h : interface of the CScopeDoc class
//
/////////////////////////////////////////////////////////////////////////////

#if !defined(AFX_SCOPEDOC_H__DA985EE4_C77B_11D3_BB39_ACF5B442421C__INCLUDED_)
#define AFX_SCOPEDOC_H__DA985EE4_C77B_11D3_BB39_ACF5B442421C__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000


class CScopeDoc : public CDocument
{
// Variables que almacenan los datos recibidos de la tarjeta de adquisición de datos
public:
	double* m_pVin;                     // Para la vista. En voltios. ¡¡ 8KBy !!
	int m_nVinPuntos;                   // Número de puntos de m_pVin
	CByteArray m_arrayMuestras;  // Almacena las muestras recibidas
	int m_nArrayPuntos;                 // Número de muestras
private:
	static double s_dValueGanancia[];    // V / div
	static double s_dValueDivisor[];            // ms / div

public:
	typedef struct {             // Almacena el estado de un canal
		int nGanancia;
		int nDivisor;
		BOOL bACDCneg;
		BOOL bEnable;
	} SCanalDoc;

	SCanalDoc* m_pCanalDoc;      // Almacena los estados de los canales

public:
	void ScopeCrearVin(); // Crea el vector para la vista
	void ScopeDecodificarTramaRX(int nCanal);

protected: // create from serialization only
	CScopeDoc();
	DECLARE_DYNCREATE(CScopeDoc)

// Attributes
public:

// Operations
public:

// Overrides
	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CScopeDoc)
	public:
	virtual BOOL OnNewDocument();
	virtual void Serialize(CArchive& ar);
	virtual BOOL OnOpenDocument(LPCTSTR lpszPathName);
	//}}AFX_VIRTUAL
// Implementation
public:
	virtual ~CScopeDoc();
#ifdef _DEBUG
	virtual void AssertValid() const;
	virtual void Dump(CDumpContext& dc) const;
#endif

protected:
// Generated message map functions
protected:
	//{{AFX_MSG(CScopeDoc)
	afx_msg void OnUpdateFileSave(CCmdUI* pCmdUI);
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};
```

```
/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately before the
previous line.

#endif // !defined(AFX_SCOPEDOC_H__DA985EE4_C77B_11D3_BB39_ACF5B442421C__INCLUDED_)
```

```
// scopeDoc.cpp : implementation of the CScopeDoc class
//

#include "stdafx.h"
#include "scope.h"

#include "scopeDoc.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CScopeDoc
//int  CScopeDoc::s_nMaxPoint = 1024;
double CScopeDoc::s_dValueDivisor[2] = {0.012, 1};
double CScopeDoc::s_dValueGanancia[8] = {0.5, 1, 2, 5, 10, 20, 50, 100};

IMPLEMENT_DYNCREATE(CScopeDoc, CDocument)

BEGIN_MESSAGE_MAP(CScopeDoc, CDocument)
	//{{AFX_MSG_MAP(CScopeDoc)
	ON_UPDATE_COMMAND_UI(ID_FILE_SAVE, OnUpdateFileSave)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CScopeDoc construction/destruction

CScopeDoc::CScopeDoc()
{
}

CScopeDoc::~CScopeDoc()
{
}

BOOL CScopeDoc::OnNewDocument()
{
	TRACE("En CScopeV1Doc::OnNewDocument()\n");

	if (!CDocument::OnNewDocument())
		return FALSE;
	// (SDI documents will reuse this document)

	m_nArrayPuntos = 1024;                      //Valor máximo
	m_arrayMuestras.SetSize(m_nArrayPuntos);
	// Se utiliza una plantilla que tiene la propiedad de
	// ser serializada para posteriormente guardarla o cargarla del disco.

	m_nVinPuntos = 0;
	m_pVin = new double [m_nArrayPuntos - 12];// Puntero para la vista. En voltios

	m_pCanalDoc = new SCanalDoc [4];

	// Valores por omisión de los canales
	m_pCanalDoc[0].nGanancia = 0;
	m_pCanalDoc[0].nDivisor =    0;
	m_pCanalDoc[0].bACDCneg =    FALSE;
	m_pCanalDoc[0].bEnable =     FALSE;

	m_pCanalDoc[1].nGanancia = 0;
	m_pCanalDoc[1].nDivisor =    0;
	m_pCanalDoc[1].bACDCneg =    FALSE;
	m_pCanalDoc[1].bEnable =     FALSE;



	m_pCanalDoc[2].nGanancia = 0;
	m_pCanalDoc[2].nDivisor =    0;
	m_pCanalDoc[2].bACDCneg =    FALSE;
	m_pCanalDoc[2].bEnable =     FALSE;

	m_pCanalDoc[3].nGanancia = 0;
	m_pCanalDoc[3].nDivisor =    0;
```

```
	m_pCanalDoc[3].bACDCneg =	FALSE;
	m_pCanalDoc[3].bEnable =	FALSE;

	return TRUE;
}



/////////////////////////////////////////////////////////////////////////////
// CScopeDoc serialization

void CScopeDoc::Serialize(CArchive& ar)
{
	m_arrayMuestras.Serialize(ar);
	// Es la clase base de la plantilla la que se encarga
	// de la serialización
}

/////////////////////////////////////////////////////////////////////////////
// CScopeDoc diagnostics

#ifdef _DEBUG
void CScopeDoc::AssertValid() const
{
	CDocument::AssertValid();
}

void CScopeDoc::Dump(CDumpContext& dc) const
{
	CDocument::Dump(dc);
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CScopeDoc commands

void CScopeDoc::OnUpdateFileSave(CCmdUI* pCmdUI)
{
	pCmdUI->Enable(IsModified());
	// Habilita el botón guardar si el documento ha sufrido cambios
}

BOOL CScopeDoc::OnOpenDocument(LPCTSTR lpszPathName)
{
	if (!CDocument::OnOpenDocument(lpszPathName))
		return FALSE;

	//By guardados = 4 By de estado+(m_nIndiceRX - 7) By de datos+4 By trama final
	m_nVinPuntos = m_arrayMuestras.GetUpperBound() - 12;
	ScopeCrearVin();
	UpdateAllViews(NULL); // Actualiza la vista
	return TRUE;
}
```

```
////////////////////////////////////////////////////////////////////////////
// Codificación y creación del vector de la vista

void CScopeDoc::ScopeCrearVin()
{
	// Crea el vector con el puntero m_pVin con doubles.
	// Los valores están en V / ms y los utilizara la vista
	// para dibujar los canales en OnDraw y OnPrint
	ScopeDecodificarTramaRX(0);   // Se decodifican los bytes
	ScopeDecodificarTramaRX(1);   // de estado de cada uno de
	ScopeDecodificarTramaRX(2);   // los canales
	ScopeDecodificarTramaRX(3);

	int nNumCanales = 1;
	if (m_pCanalDoc[1].bEnable) {
		nNumCanales = 2;
	}
	if (m_pCanalDoc[3].bEnable) {
		nNumCanales = 4;
	}

	if (nNumCanales == 1) {
		for (int i = 0; i < m_nVinPuntos; i++) {
			m_pVin[i] = ((((double) m_arrayMuestras[i + 4]) * 5 / 255) - 2.5)/
				(s_dValueGanancia[(m_pCanalDoc[0].nGanancia)] *
				 s_dValueDivisor[(m_pCanalDoc[0].nDivisor)]);
		}
	}
	else {
		if (nNumCanales == 2) {
			int nVinPuntos = m_nVinPuntos / 2;
			for (int i = 0; i < nVinPuntos; i++) {
			m_pVin[2 * i] = ((((double) m_arrayMuestras[2*i+4])*5/255)-
				2.5) / (s_dValueGanancia[(m_pCanalDoc[0].nGanancia)] *
				s_dValueDivisor[(m_pCanalDoc[0].nDivisor)]);
			m_pVin[2 * i + 1] = ((((double) m_arrayMuestras[2*i+5])*5/255)-
				2.5) / (s_dValueGanancia[(m_pCanalDoc[1].nGanancia)] *
				s_dValueDivisor[(m_pCanalDoc[1].nDivisor)]);
			}
		}
		else {
			int nVinPuntos = m_nVinPuntos / 4;
			for (int i = 0; i < nVinPuntos; i++) {
			m_pVin[4 * i] = ((((double) m_arrayMuestras[4*i+4])*5 / 255)-
				2.5) / (s_dValueGanancia[(m_pCanalDoc[0].nGanancia)] *
				s_dValueDivisor[(m_pCanalDoc[0].nDivisor)]);
			m_pVin[4 * i + 1] = ((((double) m_arrayMuestras[4*i+5])*5/ 255)-
				2.5) / (s_dValueGanancia[(m_pCanalDoc[1].nGanancia)] *
				s_dValueDivisor[(m_pCanalDoc[1].nDivisor)]);
			m_pVin[4 * i + 2] = ((((double) m_arrayMuestras[4*i+6])*5/ 255)-
				2.5) / (s_dValueGanancia[(m_pCanalDoc[2].nGanancia)] *
				s_dValueDivisor[(m_pCanalDoc[2].nDivisor)]);
			m_pVin[4 * i + 3] = ((((double) m_arrayMuestras[4*i+7])*5/ 255)-
				2.5) / (s_dValueGanancia[(m_pCanalDoc[3].nGanancia)] *
				s_dValueDivisor[(m_pCanalDoc[3].nDivisor)]);
			}
		}
	}
}
```

```
void CScopeDoc::ScopeDecodificarTramaRX(int nCanal)
{
	// Decodifica los Bytes de estado situados después
	// de la trama de inicio
	BYTE byteEstado;

	byteEstado = m_arrayMuestras[nCanal];		//	Miro ACDC
	if (byteEstado & 0x01) {			//	0000 0001
		m_pCanalDoc[nCanal].bACDCneg = TRUE;
	}
	else {
		m_pCanalDoc[nCanal].bACDCneg = FALSE;
	}

	byteEstado = m_arrayMuestras[nCanal];
	BYTE byteAux;					// Miro G3
	if (byteEstado & 0x02) {			// 0000 0010
							// G3 = 1
		byteAux = 0x04;				// 0000 0100
	}
	else {						// G3 = 0
		byteAux = 0x00;
	}
	byteEstado = m_arrayMuestras[nCanal];		// Miro G2
	if (byteEstado & 0x04) {			// 0000 0100
							// G2 = 1
		byteAux = byteAux | 0x02;		// 0000 0010
	}
	else {						// G2 = 0
		byteAux = byteAux & 0xFD;		// 1111 1101
	}
	byteEstado = m_arrayMuestras[nCanal];		// Miro G1
	if (byteEstado & 0x08) {			// 0000 1000
							// G1 = 1
		byteAux = byteAux | 0x01;		// 0000 0001
	}
	else {
							// G1 = 0
		byteAux = byteAux & 0xFE;		// 1111 1110
	}
	m_pCanalDoc[nCanal].nGanancia = (int) byteAux;

	byteEstado = m_arrayMuestras[nCanal];		// Miro FAC!
	if (byteEstado & 0x10) {			// 0001 0000
							// FAC! = 1 => * 1
		m_pCanalDoc[nCanal].nDivisor = 1;    // Sin divisor
	}
	else {
							// FAC! = 0 => * 0.012
		m_pCanalDoc[nCanal].nDivisor = 0;    // Con divisor
	}

	byteEstado = m_arrayMuestras[nCanal];		// Miro CANAL
	if (byteEstado & 0x20) {			// 0010 0000
		m_pCanalDoc[nCanal].bEnable = TRUE;
	}
	else {
		m_pCanalDoc[nCanal].bEnable = FALSE;
	}

}
```

6.2.3 CLASE CPERSISTENTFRAME

// Persist.h

```
#ifndef _INSIDE_VISUAL_CPP_PERSISTENT_FRAME
#define _INSIDE_VISUAL_CPP_PERSISTENT_FRAME

class CPersistentFrame : public CFrameWnd
{ // recuerda dónde estaba en el área de trabajo
        DECLARE_DYNAMIC(CPersistentFrame)
private:
        static const CRect s_rectDefault;
        static const char s_profileHeading[];
        static const char s_profileRect[];
        static const char s_profileIcon[];
        static const char s_profileMax[];
        static const char s_profileTool[];
        static const char s_profileStatus[];
        BOOL m_bFirstTime;
protected:     // Crea sólo a partir de la serialización
        CPersistentFrame();
        ~CPersistentFrame();

        //{{AFX_VIRTUAL(CPersistentFrame)
        public:
        virtual void ActivateFrame(int nCmdShow = -1);
        protected:
        //}}AFX_VIRTUAL

        //{{AFX_MSG(CPersistentFrame)
        afx_msg void OnDestroy();
        //}}AFX_MSG

        DECLARE_MESSAGE_MAP()
};

#endif // _INSIDE_VISUAL_CPP_PERSISTENT_FRAME
```

```
// Persist.cpp Clase de marco persistente para aplicaciones SDI

#include "stdafx.h"
#include "persist.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////
// CPersistentFrame

const CRect CPersistentFrame::s_rectDefault(10,  10, 500, 400);
const char CPersistentFrame::s_profileHeading[] =   "Tamaño ventana";
const char CPersistentFrame::s_profileRect[] =      "Rect";
const char CPersistentFrame::s_profileIcon[] =      "icono";
const char CPersistentFrame::s_profileMax[] =       "max";
const char CPersistentFrame::s_profileTool[] =      "herramienta";
const char CPersistentFrame::s_profileStatus[] =    "estado";
IMPLEMENT_DYNAMIC(CPersistentFrame, CFrameWnd)

BEGIN_MESSAGE_MAP(CPersistentFrame, CFrameWnd)
        //{{AFX_MSG_MAP(CPersistentFrame)
        ON_WM_DESTROY()
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////
CPersistentFrame::CPersistentFrame()
{
        m_bFirstTime = TRUE;
}

/////////////////////////////////////////////////////////////
CPersistentFrame::~CPersistentFrame()
{
}

/////////////////////////////////////////////////////////////
void CPersistentFrame::OnDestroy()
{
        CString strText;
        BOOL bIconic, bMaximized;

        WINDOWPLACEMENT wndpl;
        wndpl.length = sizeof(WINDOWPLACEMENT);
        // Obtiene la posición actual de la ventana y
        // su estado (iconizada/maximizada)
        BOOL bRet = GetWindowPlacement(&wndpl);
        if (wndpl.showCmd == SW_SHOWNORMAL) {
                bIconic = FALSE;
                bMaximized = FALSE;
        }
        else if (wndpl.showCmd == SW_SHOWMAXIMIZED) {
                bIconic = FALSE;
                bMaximized = TRUE;
        }
        else if (wndpl.showCmd == SW_SHOWMINIMIZED) {
                bIconic = TRUE;
                if (wndpl.flags) {
                        bMaximized = TRUE;
                }
                else {
                        bMaximized = FALSE;
                }
        }



        strText.Format("%04d %04d %04d %04d",
                        wndpl.rcNormalPosition.left,
                        wndpl.rcNormalPosition.top,
                        wndpl.rcNormalPosition.right,
                        wndpl.rcNormalPosition.bottom);
        AfxGetApp()->WriteProfileString(s_profileHeading,
                                        s_profileRect, strText);
```

```
        AfxGetApp()->WriteProfileInt(s_profileHeading,
                                     s_profileIcon, bIconic);
        AfxGetApp()->WriteProfileInt(s_profileHeading,
                                     s_profileMax, bMaximized);
        SaveBarState(AfxGetApp()->m_pszProfileName);
        CFrameWnd::OnDestroy();
}

//////////////////////////////////////////////////////////////
void CPersistentFrame::ActivateFrame(int nCmdShow)
{
        CString strText;
        BOOL bIconic, bMaximized;
        UINT flags;
        WINDOWPLACEMENT wndpl;
        CRect rect;

        if (m_bFirstTime) {
                m_bFirstTime = FALSE;
                strText = AfxGetApp()->GetProfileString(s_profileHeading,
                                                        s_profileRect);
                if (!strText.IsEmpty()) {
                        rect.left = atoi((const char*) strText);
                        rect.top = atoi((const char*) strText + 5);
                        rect.right = atoi((const char*) strText + 10);
                        rect.bottom = atoi((const char*) strText + 15);
                }
                else {
                        rect = s_rectDefault;
                }

                bIconic = AfxGetApp()->GetProfileInt(s_profileHeading,
                                                     s_profileIcon, 0);
                bMaximized = AfxGetApp()->GetProfileInt(s_profileHeading,
                                                        s_profileMax, 0);
                if (bIconic) {
                        nCmdShow = SW_SHOWMINNOACTIVE;
                        if (bMaximized) {
                                flags = WPF_RESTORETOMAXIMIZED;
                        }
                        else {
                                flags = WPF_SETMINPOSITION;
                        }
                }
                else {
                        if (bMaximized) {
                                nCmdShow = SW_SHOWMAXIMIZED;
                                flags = WPF_RESTORETOMAXIMIZED;
                        }
                        else {
                                nCmdShow = SW_NORMAL;
                                flags = WPF_SETMINPOSITION;
                        }
                }
                wndpl.length = sizeof(WINDOWPLACEMENT);
                wndpl.showCmd = nCmdShow;
                wndpl.flags = flags;
                wndpl.ptMinPosition = CPoint(0, 0);
                wndpl.ptMaxPosition =
                        CPoint(-::GetSystemMetrics(SM_CXBORDER),
                               -::GetSystemMetrics(SM_CYBORDER));
                wndpl.rcNormalPosition = rect;

                LoadBarState(AfxGetApp()->m_pszProfileName);
                // dispone la posición de la ventana, y su estado
                // (minimizado/maximizado)
                BOOL bRet = SetWindowPlacement(&wndpl);
        }
        CFrameWnd::ActivateFrame(nCmdShow);
}
```

6.2.4 CLASE CMAINFRAME

```
// MainFrm.h : interface of the CMainFrame class
//
/////////////////////////////////////////////////////////////////////////////

#if !defined(AFX_MAINFRM_H__DA985EE2_C77B_11D3_BB39_ACF5B442421C__INCLUDED_)
#define AFX_MAINFRM_H__DA985EE2_C77B_11D3_BB39_ACF5B442421C__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

#include "persist.h"

class CMainFrame : public CPersistentFrame
{
protected: // create from serialization only
        CMainFrame();
        DECLARE_DYNCREATE(CMainFrame)

// Attributes
public:

// Operations
public:

// Overrides
        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CMainFrame)
        virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
        //}}AFX_VIRTUAL

// Implementation
public:
        virtual ~CMainFrame();
#ifdef _DEBUG
        virtual void AssertValid() const;
        virtual void Dump(CDumpContext& dc) const;
#endif

protected:  // control bar embedded members
        CStatusBar  m_wndStatusBar;
        CToolBar    m_wndToolBar;

// Generated message map functions
protected:
        //{{AFX_MSG(CMainFrame)
        afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
                // NOTE - the ClassWizard will add and remove member functions here.
                //    DO NOT EDIT what you see in these blocks of generated code!
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately before the
previous line.

#endif // !defined(AFX_MAINFRM_H__DA985EE2_C77B_11D3_BB39_ACF5B442421C__INCLUDED_)
```

```
// MainFrm.cpp : implementation of the CMainFrame class
//
#include "stdafx.h"
#include "scope.h"

#include "MainFrm.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CMainFrame

IMPLEMENT_DYNCREATE(CMainFrame, CPersistentFrame)

BEGIN_MESSAGE_MAP(CMainFrame, CPersistentFrame)
	//{{AFX_MSG_MAP(CMainFrame)
		// NOTE - the ClassWizard will add and remove mapping macros here.
		//    DO NOT EDIT what you see in these blocks of generated code !
	ON_WM_CREATE()
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

static UINT indicators[] =
{
	ID_SEPARATOR,           // status line indicator
	ID_INDICATOR_CAPS,
	ID_INDICATOR_NUM,
	ID_INDICATOR_SCRL,
};

/////////////////////////////////////////////////////////////////////////////
// CMainFrame construction/destruction

CMainFrame::CMainFrame()
{
}

CMainFrame::~CMainFrame()
{
}

int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
	if (CPersistentFrame::OnCreate(lpCreateStruct) == -1)
		return -1;

	if (!m_wndToolBar.Create(this) ||
		!m_wndToolBar.LoadToolBar(IDR_MAINFRAME))
	{
		TRACE0("Failed to create toolbar\n");
		return -1;      // fail to create
	}

	if (!m_wndStatusBar.Create(this) ||
		!m_wndStatusBar.SetIndicators(indicators,
		  sizeof(indicators)/sizeof(UINT)))
	{
		TRACE0("Failed to create status bar\n");
		return -1;      // fail to create
	}

	// TODO: Remove this if you don't want tool tips or a resizeable toolbar
	m_wndToolBar.SetBarStyle(m_wndToolBar.GetBarStyle() |
		CBRS_TOOLTIPS | CBRS_FLYBY | CBRS_SIZE_DYNAMIC);


	// TODO: Delete these three lines if you don't want the toolbar to
	//  be dockable
	m_wndToolBar.EnableDocking(CBRS_ALIGN_ANY);
	EnableDocking(CBRS_ALIGN_ANY);
	DockControlBar(&m_wndToolBar);

	return 0;
```

```
}

BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
	// TODO: Modify the Window class or styles here by modifying
	//  the CREATESTRUCT cs

	return CPersistentFrame::PreCreateWindow(cs);
}

/////////////////////////////////////////////////////////////////////////////
// CMainFrame diagnostics

#ifdef _DEBUG
void CMainFrame::AssertValid() const
{
	CPersistentFrame::AssertValid();
}

void CMainFrame::Dump(CDumpContext& dc) const
{
	CPersistentFrame::Dump(dc);
}

#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CMainFrame message handlers
```

6.2.5 CLASE CSCOPEVIEW

```
// scopeView.h : interface of the CScopeView class
//
/////////////////////////////////////////////////////////////////////////////

#if !defined(AFX_SCOPEVIEW_H__DA985EE6_C77B_11D3_BB39_ACF5B442421C__INCLUDED_)
#define AFX_SCOPEVIEW_H__DA985EE6_C77B_11D3_BB39_ACF5B442421C__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

class CControlDlg;
class CActuarDlg;

class CRS232;

//-----------------------------GLOBALES-------------------------
//Declaración del hilo global
UINT MiThread(LPVOID);

//Declaración de servicio interrupción a RX global
void OnRxChar(int NumBytes, CRS232* pRs232);
//-----------------------------Fin GLOBALES--------------------

#define WM_DATA_RX WM_USER + 12

class CScopeView : public CView
{
	//Para el interface de comuinicaciones
private:
	CRS232* m_pRs232;

	typedef struct {	// Almacena estado RS232
		int nPuerto;
		int nBaudios;
		int nParidad;
		int nBitsCar;
		int nBitsParada;
		int nControlFlujo;
	}SCom;
	SCom m_comRs232;

	typedef struct {	// Para detectar la trama de inicio y final
		BOOL bByte1;
		BOOL bByte2;
		BOOL bByte3;
		BOOL bByte4;
		BYTE bytePrimero;
		BYTE byteSegundo;
		BYTE byteTercero;
		BYTE byteCuarto;
	}STrama;
	STrama m_tramaInicio, m_tramaFinal;
	// Detecta la trama de inicio y final
	void ScopeDetectarTrama(STrama* pTrama, BYTE byteValor);

	int m_nIndiceRX;	// Índice de los datos recibidos
	// si está habilitada la detección de trama inicial y final
```

```
        //Para visualización del osciloscopio
private:
        typedef struct {              // Almacena el modo de representación
                int nCanal;           // de uno de los canales
                int nFactorX;
                int nFactorY;
                int nWidthC;
                BYTE bRedC;
                BYTE bGreenC;
                BYTE bBlueC;
        } SCanal;

        SCanal m_aPaintCanal[4];      // Modo representación canales monitor
        SCanal m_aPrintCanal[4];      // Modo representación canales impresora

        static double s_dValueCanal[];              // V / div
        static double s_dValueTimeDiv[];            // ms / div

        CRect m_rectClientHimetric;          // Coordenadas ventana cliente en modo
Himetric
        static CSize s_sizeRangoHimetricPaint;
                        // Tamaño logico de ventana cliente en modo Himetric 0.01 mm

        //Dibuja un canal
        void ScopeDrawChanel(CDC* pDC,       // Puntero a contexto de dispositivo
                CScopeDoc* pDoc,             // Puntero a documento
                SCanal* pCanal,              // Puntero a modo representación canal
                int nMux,                    // Modo de multiplexación
                CSize sizeRango,             // Rango de visualización
                double dTMuestreo);          // Periodo de muestreo en ms

        //Dibuja el fondo
        void ScopeBackground(CDC* pDC,       // Puntero a contexto de dispositivo
                CSize sizeBackground, // Tamaño del fondo
                BYTE bRedB,                  // Colores del marco
                BYTE bGreenB,
                BYTE bBlueB,
                int nWidthP,                 // Grosor de las divisiones
                BYTE bRedP,                  // Color de las divisiones
                BYTE bGreenP,
                BYTE bBlueP);

        //Para los diálodos no modales
private:
        BOOL m_bOnControlDlg;
        CControlDlg* m_pControlDlg;

        BOOL m_bOnActuarDlg;
        CActuarDlg* m_pActuarDlg;

        // Para el "diálogo" de abrir/cerrar puerto
        BOOL m_bOnComunicacionDlg;

        //Funciones y variables para el registro de Windows
private:
        void ScopeGuardarFicheroIni();
        void ScopeAbrirFicheroIni();

        static const char s_profileHeadingCom[];
        static const char s_profileCom[];
        static const char s_profileBaudios[];

protected: // create from serialization only
        CScopeView();
        DECLARE_DYNCREATE(CScopeView)

// Attributes
public:
        CScopeDoc* GetDocument();

// Operations
public:

// Overrides
        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CScopeView)
        public:
```

```
        virtual void OnDraw(CDC* pDC);  // overridden to draw this view
        virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
        virtual void OnPrepareDC(CDC* pDC, CPrintInfo* pInfo = NULL);
        virtual void OnInitialUpdate();
        protected:
        virtual BOOL OnPreparePrinting(CPrintInfo* pInfo);
        virtual void OnBeginPrinting(CDC* pDC, CPrintInfo* pInfo);
        virtual void OnEndPrinting(CDC* pDC, CPrintInfo* pInfo);
        virtual void OnPrint(CDC* pDC, CPrintInfo* pInfo);
        //}}AFX_VIRTUAL

// Implementation
public:
        virtual ~CScopeView();
#ifdef _DEBUG
        virtual void AssertValid() const;
        virtual void Dump(CDumpContext& dc) const;
#endif

protected:

// Generated message map functions
protected:
        //{{AFX_MSG(CScopeView)
        afx_msg void OnScopeActuar();
        afx_msg void OnUpdateScopeActuar(CCmdUI* pCmdUI);
        afx_msg void OnScopeControl();
        afx_msg void OnUpdateScopeControl(CCmdUI* pCmdUI);
        afx_msg void OnScopeComunicacionConfigurar();
        afx_msg void OnScopeComunicacion();
        afx_msg void OnUpdateScopeComunicacion(CCmdUI* pCmdUI);
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
        afx_msg LRESULT OnScopeControlGoodbye(WPARAM wParam, LPARAM lParam);
        afx_msg LRESULT OnScopeActuarGoodbye(WPARAM wParam, LPARAM lParam);
        afx_msg LRESULT OnScopeErrorAccesPort(WPARAM wParam, LPARAM lParam);
        afx_msg LRESULT OnScopeDataRX(WPARAM wParam, LPARAM lParam);
};

#ifndef _DEBUG  // debug version in scopeView.cpp
inline CScopeDoc* CScopeView::GetDocument()
   { return (CScopeDoc*)m_pDocument; }
#endif

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately before the
previous line.
```

```
#endif // !defined(AFX_SCOPEVIEW_H__DA985EE6_C77B_11D3_BB39_ACF5B442421C__INCLUDED_)
// scopeView.cpp : implementation of the CScopeView class
//

#include "stdafx.h"
#include "scope.h"

#include "scopeDoc.h"
#include "scopeView.h"

#include "ControlDlg.h"
#include "ActuarDlg.h"

#include "Rs232v3.h"
#include "ConfigComDlg.h"


#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CScopeView

double CScopeView::s_dValueCanal[14] = {0.005, 0.01, 0.02, 0.05,
                      0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50, 100};   // En V/div
double CScopeView::s_dValueTimeDiv[7] = {0.05, 0.1, 0.2,
                      0.5, 1, 2, 5};                              // En ms/div

//Tamaño lógico visualización
CSize CScopeView::s_sizeRangoHimetricPaint = CPoint(1000, 1000);

//Para el registro
const char CScopeView::s_profileHeadingCom[] =      "Configuracion RS232";
const char CScopeView::s_profileCom[] =             "Puerto";
const char CScopeView::s_profileBaudios[] =         "Baudios";

//Para Interface comunicación----------------------------------------GLOBAL
//Definición del puntero pOnRxChar
void (*g_pOnRxChar) (int, CRS232*) = &OnRxChar;

//Creación del objeto RS232 y puntero
CRS232* g_pRs232 = new CRS232(1024, 1024);

//Definición de puntero a el Thread
CWinThread* g_pMiThread;

//Definición de puntero a la vista
CScopeView* g_pView;

//Definición del hilo
UINT MiThread(LPVOID pParam)
{
       CRS232* punteroRs232 = (CRS232*) pParam;

       punteroRs232->Thread_Att_RS232();
       return 0;
}
//-----------------------------------------------------------------Fin GOOBAL
```

```
IMPLEMENT_DYNCREATE(CScopeView, CView)

BEGIN_MESSAGE_MAP(CScopeView, CView)
	ON_MESSAGE(WM_GOODBYE_CONTROL, OnScopeControlGoodbye)
	ON_MESSAGE(WM_GOODBYE_ACTUAR, OnScopeActuarGoodbye)
	ON_MESSAGE(WM_ERROR_ACCES_PORT, OnScopeErrorAccesPort)
	ON_MESSAGE(WM_DATA_RX, OnScopeDataRX)
	//{{AFX_MSG_MAP(CScopeView)
	ON_COMMAND(ID_SCOPE_ACTUAR, OnScopeActuar)
	ON_UPDATE_COMMAND_UI(ID_SCOPE_ACTUAR, OnUpdateScopeActuar)
	ON_COMMAND(ID_SCOPE_CONTROL, OnScopeControl)
	ON_UPDATE_COMMAND_UI(ID_SCOPE_CONTROL, OnUpdateScopeControl)
	ON_COMMAND(ID_SCOPE_COMUNICACION_CONFIGURAR, OnScopeComunicacionConfigurar)
	ON_COMMAND(ID_SCOPE_COMUNICACION, OnScopeComunicacion)
	ON_UPDATE_COMMAND_UI(ID_SCOPE_COMUNICACION, OnUpdateScopeComunicacion)
	//}}AFX_MSG_MAP
	// Standard printing commands
	ON_COMMAND(ID_FILE_PRINT, CView::OnFilePrint)
	ON_COMMAND(ID_FILE_PRINT_DIRECT, CView::OnFilePrint)
	ON_COMMAND(ID_FILE_PRINT_PREVIEW, CView::OnFilePrintPreview)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CScopeView construction/destruction

CScopeView::CScopeView()
{
	g_pView = this;

	ScopeAbrirFicheroIni();

	m_bOnComunicacionDlg = FALSE;
	m_pRs232 = g_pRs232;

	// Estado e inicialización de la trama de inicio a detectar
	m_tramaInicio.bByte1 = FALSE;
	m_tramaInicio.bByte2 = FALSE;
	m_tramaInicio.bByte3 = FALSE;
	m_tramaInicio.bByte4 = FALSE;
	m_tramaInicio.bytePrimero = 0xFB;
	m_tramaInicio.byteSegundo = 0x04;
	m_tramaInicio.byteTercero = 0xFB;
	m_tramaInicio.byteCuarto =  0x04;

	// Estado e inicialización de la trama final a detectar
	m_tramaFinal.bByte1 = FALSE;
	m_tramaFinal.bByte2 = FALSE;
	m_tramaFinal.bByte3 = FALSE;
	m_tramaFinal.bByte4 = FALSE;
	m_tramaFinal.bytePrimero = 0x04;
	m_tramaFinal.byteSegundo = 0xFB;
	m_tramaFinal.byteTercero = 0x04;
	m_tramaFinal.byteCuarto =  0xFB;

	m_nIndiceRX = 0;	// By recibidos

	m_pControlDlg = new CControlDlg(this);
	m_bOnControlDlg = FALSE;

	m_pActuarDlg = new CActuarDlg(this);
	m_bOnActuarDlg = FALSE;
```

```
        //Anchos de pluma y colores de los canales para la pantalla
        m_aPaintCanal[0].nCanal =           0;
        m_aPaintCanal[0].nFactorX =   0;
        m_aPaintCanal[0].nFactorY =   0;
        m_aPaintCanal[0].nWidthC =    4;
        m_aPaintCanal[0].bRedC =            255;
        m_aPaintCanal[0].bGreenC =    255;
        m_aPaintCanal[0].bBlueC =           255;

        m_aPaintCanal[1].nCanal =           1;
        m_aPaintCanal[1].nFactorX =   0;
        m_aPaintCanal[1].nFactorY =   0;
        m_aPaintCanal[1].nWidthC =    4;
        m_aPaintCanal[1].bRedC =            255;
        m_aPaintCanal[1].bGreenC =    150;
        m_aPaintCanal[1].bBlueC =           150;

        m_aPaintCanal[2].nCanal =           2;
        m_aPaintCanal[2].nFactorX =   0;
        m_aPaintCanal[2].nFactorY =   0;
        m_aPaintCanal[2].nWidthC =    4;
        m_aPaintCanal[2].bRedC =            150;
        m_aPaintCanal[2].bGreenC =    255;
        m_aPaintCanal[2].bBlueC =           150;

        m_aPaintCanal[3].nCanal =           3;
        m_aPaintCanal[3].nFactorX =   0;
        m_aPaintCanal[3].nFactorY =   0;
        m_aPaintCanal[3].nWidthC =    4;
        m_aPaintCanal[3].bRedC =            150;
        m_aPaintCanal[3].bGreenC =    150;
        m_aPaintCanal[3].bBlueC =           255;

        //Anchos de pluma y colores de los canales para la impresora
        m_aPrintCanal[0].nCanal =           0;
        m_aPrintCanal[0].nFactorX =   0;
        m_aPrintCanal[0].nFactorY =   0;
        m_aPrintCanal[0].nWidthC =    4;
        m_aPrintCanal[0].bRedC =            0;
        m_aPrintCanal[0].bGreenC =    0;
        m_aPrintCanal[0].bBlueC =           0;

        m_aPrintCanal[1].nCanal =           1;
        m_aPrintCanal[1].nFactorX =   0;
        m_aPrintCanal[1].nFactorY =   0;
        m_aPrintCanal[1].nWidthC =    4;
        m_aPrintCanal[1].bRedC =            255;
        m_aPrintCanal[1].bGreenC =    0;
        m_aPrintCanal[1].bBlueC =           0;

        m_aPrintCanal[2].nCanal =           2;
        m_aPrintCanal[2].nFactorX =   0;
        m_aPrintCanal[2].nFactorY =   0;
        m_aPrintCanal[2].nWidthC =    4;
        m_aPrintCanal[2].bRedC =            0;
        m_aPrintCanal[2].bGreenC =    255;
        m_aPrintCanal[2].bBlueC =           0;

        m_aPrintCanal[3].nCanal =           3;
        m_aPrintCanal[3].nFactorX =   0;
        m_aPrintCanal[3].nFactorY =   0;
        m_aPrintCanal[3].nWidthC =    4;
        m_aPrintCanal[3].bRedC =            0;
        m_aPrintCanal[3].bGreenC =    0;
        m_aPrintCanal[3].bBlueC =           255;
}
```

```
CScopeView::~CScopeView()
{
	ScopeGuardarFicheroIni();
	if (m_pRs232->PortObert) {
		m_pRs232->TancarPort();
	}

	delete m_pControlDlg;
	delete m_pActuarDlg;
}

void CScopeView::OnInitialUpdate()
{
	// Se inicializan los datos miembros de los diálogos
	// Se entra también después de nuevo documento
	m_pControlDlg->m_nTrackbarCanal1 =  0;
	m_pControlDlg->m_nTrackbarCanal2 =  0;
	m_pControlDlg->m_nTrackbarCanal3 =  0;
	m_pControlDlg->m_nTrackbarCanal4 =  0;
	m_pControlDlg->m_nTrackbarTimeDiv = 0;
	m_pControlDlg->m_nCanales = 0;
	m_pControlDlg->m_bControlManual = FALSE;
	m_pControlDlg->m_bInteractivo =   FALSE;
	m_pControlDlg->m_nAcDcC1 = 0;
	m_pControlDlg->m_nAcDcC2 = 0;
	m_pControlDlg->m_nAcDcC3 = 0;
	m_pControlDlg->m_nAcDcC4 = 0;

	m_pActuarDlg->m_nCanal =    0;
	m_pActuarDlg->m_nDivisor =  0;
	m_pActuarDlg->m_nGanancia = 0;
	m_pActuarDlg->m_nDCAC = 0;
	m_pActuarDlg->m_nEnable = FALSE;
	m_pActuarDlg->m_bSimula = FALSE;
	m_pActuarDlg->m_bConTramas = TRUE;
	m_pActuarDlg->m_bEnableIn =  FALSE;
	m_pActuarDlg->m_bEnableOut = FALSE;
	UpdateData(FALSE);

	if (m_bOnControlDlg==FALSE) {
		OnScopeControl();     //Si el diálogo de control esta cerrado lo abro
	}

	m_pControlDlg->OnApplyPublic();
}

BOOL CScopeView::PreCreateWindow(CREATESTRUCT& cs)
{
	// TODO: Modify the Window class or styles here by modifying
	//  the CREATESTRUCT cs

	return CView::PreCreateWindow(cs);
}
```

```
////////////////////////////////////////////////////////////////////////////
// CScopeView apoyo a la visualización

void CScopeView::ScopeBackground(CDC* pDC,
                                 CSize sizeBackground,
                                 BYTE bRedB,
                                 BYTE bGreenB,
                                 BYTE bBlueB,
                                 int nWidthP,
                                 BYTE bRedP,
                                 BYTE bGreenP,
                                 BYTE bBlueP)
{       //En unidades lógicas
        //En Origen de las coordenadas(0,0) en la izquierda, al medio
        //Crea un rejilla de 10 x 10 cuadros
        CPen newPen(PS_SOLID, nWidthP, RGB(bRedP, bGreenP, bBlueP));
        CPen* pOldPen = pDC->SelectObject(&newPen);

        CBrush newBrush(RGB(bRedB, bGreenB, bBlueB));
        CBrush* pOldBrush = pDC->SelectObject(&newBrush);

        pDC->Rectangle(CRect(0, sizeBackground.cy / 2,
                sizeBackground.cx, - sizeBackground.cy / 2));//En unidades lógicas

        //Lineas Horizontales centradas con origen y
        pDC->MoveTo(0,              4 * sizeBackground.cy / 10);
        pDC->LineTo(sizeBackground.cx, 4 * sizeBackground.cy / 10);
        pDC->MoveTo(0,              3 * sizeBackground.cy / 10);
        pDC->LineTo(sizeBackground.cx, 3 * sizeBackground.cy / 10);
        pDC->MoveTo(0,              2 * sizeBackground.cy / 10);
        pDC->LineTo(sizeBackground.cx, 2 * sizeBackground.cy / 10);
        pDC->MoveTo(0,              1 * sizeBackground.cy / 10);
        pDC->LineTo(sizeBackground.cx, 1 * sizeBackground.cy / 10);
        pDC->MoveTo(0,              0);
        pDC->LineTo(sizeBackground.cx, 0);
        pDC->MoveTo(0,              -1 * sizeBackground.cy / 10);
        pDC->LineTo(sizeBackground.cx, -1 * sizeBackground.cy / 10);
        pDC->MoveTo(0,              -2 * sizeBackground.cy / 10);
        pDC->LineTo(sizeBackground.cx, -2 * sizeBackground.cy / 10);
        pDC->MoveTo(0,              -3 * sizeBackground.cy / 10);
        pDC->LineTo(sizeBackground.cx, -3 * sizeBackground.cy / 10);
        pDC->MoveTo(0,              -4 * sizeBackground.cy / 10);
        pDC->LineTo(sizeBackground.cx, -4 * sizeBackground.cy / 10);

        //Vertical situada a la derecha del origen x
        pDC->MoveTo(    sizeBackground.cx / 10,  sizeBackground.cy / 2);
        pDC->LineTo(    sizeBackground.cx / 10, -sizeBackground.cy / 2);
        pDC->MoveTo(2 * sizeBackground.cx / 10,  sizeBackground.cy / 2);
        pDC->LineTo(2 * sizeBackground.cx / 10, -sizeBackground.cy / 2);
        pDC->MoveTo(3 * sizeBackground.cx / 10,  sizeBackground.cy / 2);
        pDC->LineTo(3 * sizeBackground.cx / 10, -sizeBackground.cy / 2);
        pDC->MoveTo(4 * sizeBackground.cx / 10,  sizeBackground.cy / 2);
        pDC->LineTo(4 * sizeBackground.cx / 10, -sizeBackground.cy / 2);
        pDC->MoveTo(5 * sizeBackground.cx / 10,  sizeBackground.cy / 2);
        pDC->LineTo(5 * sizeBackground.cx / 10, -sizeBackground.cy / 2);
        pDC->MoveTo(6 * sizeBackground.cx / 10,  sizeBackground.cy / 2);
        pDC->LineTo(6 * sizeBackground.cx / 10, -sizeBackground.cy / 2);
        pDC->MoveTo(7 * sizeBackground.cx / 10,  sizeBackground.cy / 2);
        pDC->LineTo(7 * sizeBackground.cx / 10, -sizeBackground.cy / 2);
        pDC->MoveTo(8 * sizeBackground.cx / 10,  sizeBackground.cy / 2);
        pDC->LineTo(8 * sizeBackground.cx / 10, -sizeBackground.cy / 2);
        pDC->MoveTo(9 * sizeBackground.cx / 10,  sizeBackground.cy / 2);
        pDC->LineTo(9 * sizeBackground.cx / 10, -sizeBackground.cy / 2);

        pDC->SelectObject(pOldPen);
        pDC->SelectObject(pOldBrush);
}
```

```
void CScopeView::ScopeDrawChanel(CDC* pDC,
                                 CScopeDoc* pDoc,
                                 SCanal* pCanal,
                                 int nMux,
                                 CSize sizeRango,
                                 double dTMuestreo)

{
        // Dibuja un canal. En pantalla o en presentación
        // preliminar según quien la llame
        CPen newPen(PS_SOLID,
                pCanal->nWidthC,
                RGB(pCanal->bRedC,
                pCanal->bGreenC,
                pCanal->bBlueC));
        CPen* pOldPen = pDC->SelectObject(&newPen);

        int i, j, x, y, nYmax;

        nYmax = sizeRango.cy / 2;

        // Primer punto
        pDC->MoveTo(CPoint(pCanal->nCanal * dTMuestreo * sizeRango.cx * 0.1 /
                                                s_dValueTimeDiv[(pCanal->nFactorX)],
                pDoc->m_pVin[pCanal->nCanal] * sizeRango.cy * 0.1  /
                                                s_dValueCanal[(pCanal->nFactorY)]));

        for (i = 1; i < ((pDoc->m_nVinPuntos / nMux)); i++) {
                j = (i * nMux) + pCanal->nCanal;
                x = (int) j * dTMuestreo * sizeRango.cx * 0.1 /
                                        s_dValueTimeDiv[(pCanal->nFactorX)]; //En ms
                y = (int) (pDoc->m_pVin[j] * sizeRango.cy * 0.1  /
                                        s_dValueCanal[(pCanal->nFactorY)]);  //En V

                // Para no salirse del tamaño del fondo
                if (y > nYmax) {
                        y = nYmax;
                }
                if (y < -nYmax) {
                        y = -nYmax;
                }
                if (x > sizeRango.cx) {
                        x = sizeRango.cx;
                }
                if (x < -sizeRango.cx) {
                        x = -sizeRango.cx;
                }
                pDC->LineTo(CPoint(x, y));    //En coordenadas lógicas
        }
        pDC->SelectObject(pOldPen);
}
```

```
/////////////////////////////////////////////////////////////////////////////
// CScopeView drawing

void CScopeView::OnDraw(CDC* pDC)
{
	GetClientRect(m_rectClientHimetric);		// 0, 0, Ancho, Alto
	pDC->SetMapMode(MM_ISOTROPIC);				// Relación 1:1
	pDC->SetWindowExt(s_sizeRangoHimetricPaint);//  Rango de visualización
	pDC->SetViewportExt(m_rectClientHimetric.right, - m_rectClientHimetric.bottom);
	pDC->SetViewportOrg(0, m_rectClientHimetric.bottom / 2);   // Origen.
	pDC->DPtoLP(m_rectClientHimetric);// De unidades lógica a dispositivo (pixels)
	m_rectClientHimetric.NormalizeRect();		// Normalización del
rectángulo

	CSize sizeClientHimetric = m_rectClientHimetric.Size();    // Obtiene el tamaño

	int nSizeMinLogical;

	// Se obtiene el tamaño mínimo
	if (sizeClientHimetric.cy > sizeClientHimetric.cx) {
		nSizeMinLogical = sizeClientHimetric.cx;
	}
	else {
		nSizeMinLogical = sizeClientHimetric.cy;
	}

	CScopeDoc* pDoc = GetDocument();			// Se obtiene el puntero a documento

	int nCanalMux = 0;							// Canal 1 siempre activo
	if (pDoc->m_pCanalDoc[1].bEnable) {
		nCanalMux = 1;
	}
	if (pDoc->m_pCanalDoc[3].bEnable) {
		nCanalMux = 3;
	}

	// Se dibuja el fondo
	ScopeBackground(pDC,
		CSize(nSizeMinLogical, nSizeMinLogical)/*Unidades lógicas*/,
		0	/*RedB*/,
		0	/*GreenB*/,
		0	/*BlueB*/,
		1	/*WidthP*/,
		100	/*RedP*/,
		100	/*GreenP*/,
		100	/*BlueP*/);

	// Se dibuja el/los canales en función del estado recibido de las
	// muestras, que fue procesado por el documento
	if (nCanalMux == 0) {
		ScopeDrawChanel(pDC,
			pDoc,
			&m_aPaintCanal[0],
			1,
			s_sizeRangoHimetricPaint,
			0.019/*En ms*/);
	}
```

```
	if (nCanalMux == 1) {
		ScopeDrawChanel(pDC,
			pDoc,
			&m_aPaintCanal[0],
			2,
			s_sizeRangoHimetricPaint,
			0.034/*En ms*/);

		ScopeDrawChanel(pDC,
			pDoc,
			&m_aPaintCanal[1],
			2,
			s_sizeRangoHimetricPaint,
			0.034/*En ms*/);
		}

	if (nCanalMux == 3) {
		ScopeDrawChanel(pDC,
			pDoc,
			&m_aPaintCanal[0],
			4,
			s_sizeRangoHimetricPaint,
			0.044/*En ms*/);

		ScopeDrawChanel(pDC,
			pDoc,
			&m_aPaintCanal[1],
			4,
			s_sizeRangoHimetricPaint,
			0.044/*En ms*/);

		ScopeDrawChanel(pDC,
			pDoc,
			&m_aPaintCanal[2],
			4,
			s_sizeRangoHimetricPaint,
			0.044/*En ms*/);

		ScopeDrawChanel(pDC,
			pDoc,
			&m_aPaintCanal[3],
			4,
			s_sizeRangoHimetricPaint,
			0.044/*En ms*/);
		}
}

/////////////////////////////////////////////////////////////////////////////
// CScopeView printing

void CScopeView::OnPrepareDC(CDC* pDC, CPrintInfo* pInfo)
{
	pDC->SetMapMode(MM_HIMETRIC); //Modo de 0,01mm
}

void CScopeView::OnPrint(CDC* pDC, CPrintInfo* pInfo)
{
	// Momentos previos a lo presentación preliminar
	UpdateData(TRUE);     // Se obtiene el estado del diálogo de control
	m_aPrintCanal[0].nFactorY = (int) m_pControlDlg->m_nTrackbarCanal1;
	m_aPrintCanal[0].nFactorX = (int) m_pControlDlg->m_nTrackbarTimeDiv;
	m_aPrintCanal[1].nFactorY = (int) m_pControlDlg->m_nTrackbarCanal2;
	m_aPrintCanal[1].nFactorX = (int) m_pControlDlg->m_nTrackbarTimeDiv;
	m_aPrintCanal[2].nFactorY = (int) m_pControlDlg->m_nTrackbarCanal3;
	m_aPrintCanal[2].nFactorX = (int) m_pControlDlg->m_nTrackbarTimeDiv;
	m_aPrintCanal[3].nFactorY = (int) m_pControlDlg->m_nTrackbarCanal4;
	m_aPrintCanal[3].nFactorX = (int) m_pControlDlg->m_nTrackbarTimeDiv;



	pDC->SetMapMode(MM_HIMETRIC);               //Modo de 0,01mm
	CRect rectPrintLogical = pInfo->m_rectDraw;
	//Obtiene el tamaño de impresión escogido por la impresora
	//En unidades lógicas => las escogidas po OnPrepareDC. 0, 0, alto, -ancho

	CRect rectPrintDevice = rectPrintLogical;
	int nSizeMinPrintDevice;
```

```
rectPrintDevice.NormalizeRect();

pDC->LPtoDP(rectPrintDevice);
CSize sizePrintDevice = rectPrintDevice.Size();
// Se obtiene el tamaño mínimo
if (sizePrintDevice.cx > - sizePrintDevice.cy) {
	nSizeMinPrintDevice = - sizePrintDevice.cy; // en unidades dispositivo
}
else {
	nSizeMinPrintDevice = sizePrintDevice.cx;
}
pDC->SetViewportOrg(0, nSizeMinPrintDevice / 2);    // Se establece el origen

rectPrintLogical.NormalizeRect();
CSize sizePrintLogical = rectPrintLogical.Size();

int nSizeMinPrintLogical;
// Se establece el tamaño mínimo
if (sizePrintLogical.cy > sizePrintLogical.cx) {
	nSizeMinPrintLogical = sizePrintLogical.cx; // en unidades lógicas
}
else {
	nSizeMinPrintLogical = sizePrintLogical.cy;
}

CSize sizeRangoHimetricPrint = CSize(nSizeMinPrintLogical,
                                                    nSizeMinPrintLogical);

CScopeDoc* pDoc = GetDocument();     // Se obtiene el puntero del documento

int nCanalMux = 0;    //Canal 1 siempre activo
if (pDoc->m_pCanalDoc[1].bEnable) {
	nCanalMux = 1;
}
if (pDoc->m_pCanalDoc[3].bEnable) {
	nCanalMux = 3;
}

// Se dibuja el fondo
ScopeBackground(pDC,
	CSize(nSizeMinPrintLogical, nSizeMinPrintLogical)/*Unidades lógicas*/,
	255            /*RedB*/,
	255            /*GreenB*/,
	255            /*BlueB*/,
	1              /*WidthP*/,
	100            /*RedP*/,
	100            /*GreenP*/,
	100            /*BlueP*/);

// Se dibujan el/los canal/es
if (nCanalMux == 0) {
	ScopeDrawChanel(pDC,
		pDoc,
		&m_aPrintCanal[0],
		1,
		sizeRangoHimetricPrint,
		0.019/*En ms*/);
}
if (nCanalMux == 1) {
	ScopeDrawChanel(pDC,
		pDoc,
		&m_aPrintCanal[0],
		2,
		sizeRangoHimetricPrint,
		2 * 0.034/*En ms*/);

	ScopeDrawChanel(pDC,
		pDoc,
		&m_aPrintCanal[1],
		2,
		sizeRangoHimetricPrint,
		2 * 0.034/*En ms*/);
	}
```

```
        if (nCanalMux == 3) {
                ScopeDrawChanel(pDC,
                        pDoc,
                        &m_aPrintCanal[0],
                        4,
                        sizeRangoHimetricPrint,
                        4 * 0.044/*En ms*/);

                ScopeDrawChanel(pDC,
                        pDoc,
                        &m_aPrintCanal[1],
                        4,
                        sizeRangoHimetricPrint,
                        4 * 0.044/*En ms*/);

                ScopeDrawChanel(pDC,
                        pDoc,
                        &m_aPrintCanal[2],
                        4,
                        sizeRangoHimetricPrint,
                        4 * 0.044/*En ms*/);

                ScopeDrawChanel(pDC,
                        pDoc,
                        &m_aPrintCanal[3],
                        4,
                        sizeRangoHimetricPrint,
                        4 * 0.044/*En ms*/);
                }
}

BOOL CScopeView::OnPreparePrinting(CPrintInfo* pInfo)
{
        pInfo->SetMaxPage(1); // Solo se imprime una página
        return DoPreparePrinting(pInfo);
}

void CScopeView::OnBeginPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
}

void CScopeView::OnEndPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
}

/////////////////////////////////////////////////////////////////////////////
// CScopeView diagnostics

#ifdef _DEBUG
void CScopeView::AssertValid() const
{
        CView::AssertValid();
}

void CScopeView::Dump(CDumpContext& dc) const
{
        CView::Dump(dc);
}

CScopeDoc* CScopeView::GetDocument() // non-debug version is inline
{
        ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CScopeDoc)));
        return (CScopeDoc*)m_pDocument;
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CScopeView message handlers
```

```
////////////////////////////////////////////////////////////////////////////
// Diálogo de control

void CScopeView::OnScopeControl()
{
	// Crea el diálogo si todavía no está creado
	if (m_bOnControlDlg) {// No se pierden los datos miembro, sólo se elimina
		m_pControlDlg->DestroyWindow();	// la ventana
		m_bOnControlDlg = FALSE;
	}
	else {
		if (m_pControlDlg->GetSafeHwnd() == 0) {
			m_pControlDlg->Create();	//Muestra la ventana del diálogo
			m_bOnControlDlg = TRUE;
		}
	}
}

void CScopeView::OnUpdateScopeControl(CCmdUI* pCmdUI)
{
	if (m_bOnControlDlg) {
		pCmdUI->SetCheck(1);
	}
	else {
		pCmdUI->SetCheck(0);
	}
}


LRESULT CScopeView::OnScopeControlGoodbye(WPARAM wParam, LPARAM lParam)
{
	//Mensaje recibido en respuesta a la acción del diáologo Control
	//En wParam se almacena el ID que lo llamaó
	if (wParam == IDCANCEL) {
		m_pControlDlg->DestroyWindow();
		m_bOnControlDlg = FALSE;
	}
	if (wParam == IDOK) {
		m_pControlDlg->DestroyWindow();
		m_bOnControlDlg = FALSE;

		m_aPaintCanal[0].nFactorY = (int) m_pControlDlg->m_nTrackbarCanal1;
		m_aPaintCanal[0].nFactorX = (int) m_pControlDlg->m_nTrackbarTimeDiv;

		m_aPaintCanal[1].nFactorY = (int) m_pControlDlg->m_nTrackbarCanal2;
		m_aPaintCanal[1].nFactorX = (int) m_pControlDlg->m_nTrackbarTimeDiv;

		m_aPaintCanal[2].nFactorY = (int) m_pControlDlg->m_nTrackbarCanal3;
		m_aPaintCanal[2].nFactorX = (int) m_pControlDlg->m_nTrackbarTimeDiv;

		m_aPaintCanal[3].nFactorY = (int) m_pControlDlg->m_nTrackbarCanal4;
		m_aPaintCanal[3].nFactorX = (int) m_pControlDlg->m_nTrackbarTimeDiv;

		InvalidateRect(m_rectClientHimetric);	// En coordenadas lógicas
	}

	if (wParam == IDC_MIAPPLY) {  //Envía la información a la tarjeta
		UpdateData(TRUE);
		if (m_pControlDlg->m_bInteractivo) {
			if (m_pActuarDlg->m_bEnableOut) {
				char pcTX[10];

				_itoa((BYTE) m_pControlDlg->m_pTXCadena[0], pcTX, 16);
				m_pActuarDlg->m_strEditTX += pcTX;
				m_pActuarDlg->m_strEditTX += " ";
				m_pActuarDlg->UpdateData(FALSE);
			}
			BOOL bTransmit = m_pRs232->TX(
				m_pControlDlg->m_pTXCadena,1);
			if (bTransmit == 0) {
			char pcMensaje[100];

			wsprintf(pcMensaje, "Error al escribir en el puerto serie");
			MessageBox(pcMensaje, AfxGetAppName(), MB_OK | MB_ICONSTOP);
			}
		}
	}
```

```
	if (wParam == IDAPPLY) {			//Actualiza la vista
		m_aPaintCanal[0].nFactorY = (int) m_pControlDlg->m_nTrackbarCanal1;
		m_aPaintCanal[0].nFactorX = (int) m_pControlDlg->m_nTrackbarTimeDiv;

		m_aPaintCanal[1].nFactorY = (int) m_pControlDlg->m_nTrackbarCanal2;
		m_aPaintCanal[1].nFactorX = (int) m_pControlDlg->m_nTrackbarTimeDiv;

		m_aPaintCanal[2].nFactorY = (int) m_pControlDlg->m_nTrackbarCanal3;
		m_aPaintCanal[2].nFactorX = (int) m_pControlDlg->m_nTrackbarTimeDiv;

		m_aPaintCanal[3].nFactorY = (int) m_pControlDlg->m_nTrackbarCanal4;
		m_aPaintCanal[3].nFactorX = (int) m_pControlDlg->m_nTrackbarTimeDiv;

		InvalidateRect(m_rectClientHimetric);	//En coordenadas lógicas
	}
	return 0L;
}



////////////////////////////////////////////////////////////////////////////
// Diálogo de control avanzado

void CScopeView::OnScopeActuar()
{
	if (m_bOnActuarDlg) {
		m_pActuarDlg->DestroyWindow();
		m_bOnActuarDlg = FALSE;
	}
	else {
		if (m_pActuarDlg->GetSafeHwnd() == 0) {
			m_pActuarDlg->Create(); // muestra la ventana del diálogo
			m_bOnActuarDlg = TRUE;
		}
	}
}

void CScopeView::OnUpdateScopeActuar(CCmdUI* pCmdUI)
{
	if (m_bOnActuarDlg) {
		pCmdUI->SetCheck(1);
	}
	else {
		pCmdUI->SetCheck(0);
	}
}

LRESULT CScopeView::OnScopeActuarGoodbye(WPARAM wParam, LPARAM lParam)
{
	//Mensaje recibido en respuesta a los mensajes del diálogo avanzado
	if (wParam == IDCANCEL) {
		m_pActuarDlg->DestroyWindow();
		m_bOnActuarDlg = FALSE;
	}

	if (wParam == IDOK) {
		m_pActuarDlg->DestroyWindow();
		m_bOnActuarDlg = FALSE;
	}

	if (wParam == IDENVIAR) {
		BOOL bTransmit = m_pRs232->TX(
			m_pActuarDlg->m_pCadenaTX,
			m_pActuarDlg->m_nNumBytesTX);
		if (bTransmit == 0) {
			char pcMensaje[100];
			wsprintf(pcMensaje, "Error al escribir en el puerto serie");
			MessageBox(pcMensaje, AfxGetAppName(), MB_OK | MB_ICONSTOP);
		}
	}
	return 0L;
}
```

```
////////////////////////////////////////////////////////////////////////////
// Diálogos de comunicación
void CScopeView::OnScopeComunicacion()
{
	//Abrir o cerrar el puerto serie
	char pcMensaje[100];
	if (m_bOnComunicacionDlg) {
		UpdateData(TRUE);
		if (!m_pControlDlg->m_bInteractivo) {
		m_pRs232->TancarPort();
		m_bOnComunicacionDlg = FALSE;
		wsprintf(pcMensaje, "Puerto serie cerrado");
		MessageBox(pcMensaje, AfxGetAppName(), MB_OK | MB_ICONINFORMATION);
		}
		else {
		//Para evitar "infinitos" mensajes de error al escribir en el puerto
		wsprintf(pcMensaje, "Inhabilitar antes el control interactivo");
		MessageBox(pcMensaje, AfxGetAppName(), MB_OK | MB_ICONINFORMATION);
		}
	}
	else {
		m_pRs232->ConfigurarPort(
			m_comRs232.nPuerto,
			m_comRs232.nBaudios,
			m_comRs232.nBitsCar,
			m_comRs232.nBitsParada,
			m_comRs232.nParidad,
			m_comRs232.nControlFlujo,
			GetSafeHwnd(),        //Ventana a la que enviaré los mensajes
			g_pOnRxChar);                //Puntero a función de servicio
		BOOL bAbrir = m_pRs232->ObrirPort();
		if (bAbrir == FALSE) {
			wsprintf(pcMensaje, "Error al abrir el puerto serie");
			MessageBox(pcMensaje, AfxGetAppName(), MB_OK | MB_ICONSTOP);
		}
		else {
			//Puesto abierto. Puesta en marcha del Thread
			g_pMiThread = AfxBeginThread(
				MiThread,
				m_pRs232,
				THREAD_PRIORITY_TIME_CRITICAL);
		m_bOnComunicacionDlg = TRUE;
		wsprintf(pcMensaje, "Puerto serie abierto satisfactoriamente");
		MessageBox(pcMensaje, AfxGetAppName(), MB_OK | MB_ICONINFORMATION);
		}
	}
}

void CScopeView::OnScopeComunicacionConfigurar()
{
	//Configurar puerto
	if (!m_bOnComunicacionDlg) {
		CConfigComDlg dlg;
		dlg.m_nCom = m_comRs232.nPuerto;
		dlg.m_nBaudios = m_comRs232.nBaudios;

		if (dlg.DoModal() != IDOK) return;

		m_comRs232.nPuerto = dlg.m_nCom;
		m_comRs232.nBaudios = dlg.m_nBaudios;
	}
	else {
		char Mensaje[100];
		wsprintf(Mensaje, "Cerrar antes el puerto serie");
		MessageBox(Mensaje, AfxGetAppName(), MB_OK | MB_ICONSTOP);
	}
}
void CScopeView::OnUpdateScopeComunicacion(CCmdUI* pCmdUI)
{
	if (m_bOnComunicacionDlg) {
		pCmdUI->SetCheck(1);
	}
	else {
		pCmdUI->SetCheck(0);
	}
}
```

```
////////////////////////////////////////////////////////////////////////////
// Funciones de recepción

void OnRxChar(int nNumBytes, CRS232* pRs232)
{
	// Rutina de atención al canal serie. Es global
	// Manda un mensaje a la vista por cada BYTE recibido
	BYTE pCadena[1024];

	pRs232->RXcopia(pCadena);
	for (int j = 0; j < nNumBytes; j++) {
		g_pView->PostMessage(WM_DATA_RX, pCadena[j], j);
	}
}

LRESULT CScopeView::OnScopeDataRX(WPARAM wParam, LPARAM lParam)
{
	//Mensaje recibido por cada recepción
	BYTE byteValor = (BYTE) wParam;
	char pcRX[10];
	CScopeDoc* pDoc = GetDocument();

	UpdateData(TRUE);
	if (m_pActuarDlg->m_bConTramas) {		//Con trama de inicio y final
		if (m_tramaInicio.bByte4) {		//Trama inicio detectada,
			ScopeDetectarTrama(&m_tramaFinal, byteValor);
			if (m_tramaFinal.bByte4) {		//Trama final detecatda =>
fin
				m_tramaInicio.bByte1 = FALSE; // Reset
				m_tramaInicio.bByte2 = FALSE;
				m_tramaInicio.bByte3 = FALSE;
				m_tramaInicio.bByte4 = FALSE;

				m_tramaFinal.bByte1 = FALSE;
				m_tramaFinal.bByte2 = FALSE;
				m_tramaFinal.bByte3 = FALSE;
				m_tramaFinal.bByte4 = FALSE;

				if (m_bOnActuarDlg) {
					if (m_pActuarDlg->m_bEnableIn) {
						_itoa(m_nIndiceRX + 1, pcRX, 16);
						m_pActuarDlg->m_strEditRX += " fb n=";
						m_pActuarDlg->m_strEditRX += pcRX;
					}
				}
			//By guardados = (m_nIndiceRX + 1) By (por empezar en 0) =
			//       4 By de estado + X By de datos + 4 By trama final
			//By recibidos reales = 4 By trama inicial + (m_nIndiceRX + 1) By
				pDoc->m_nVinPuntos = m_nIndiceRX - 7;
				// Se habilita botón guardar
				GetDocument()->SetModifiedFlag();
				// Se crea el nuevo vector que usará la vista
				pDoc->ScopeCrearVin();
				InvalidateRect(m_rectClientHimetric);
				// Actualiza la vista
				m_nIndiceRX = 0;
			}
			else {  //Cargar buffer. Se detectó la trama de inicio



				if (m_bOnActuarDlg) {
					if (m_pActuarDlg->m_bEnableIn) {
						_itoa(byteValor, pcRX, 16);
						if (m_nIndiceRX == 0) {
							m_pActuarDlg->m_strEditRX = pcRX;
						}
						else {
							m_pActuarDlg->m_strEditRX += " ";
							m_pActuarDlg->m_strEditRX += pcRX;
						}
					}
				}
				pDoc->m_arrayMuestras[m_nIndiceRX] = byteValor;
							// Se actualiza el documento
				m_nIndiceRX++;
			}
```

```
                }
                else {                          // Se está detectando la trama de inicio
                                ScopeDetectarTrama(&m_tramaInicio, byteValor);
                                m_nIndiceRX = 0;
                }
        }
        else {  //Sin detectar tramas. Sólo se aprecia el el diálogo Control avanzado
                if (m_bOnActuarDlg) {
                        if (m_pActuarDlg->m_bEnableIn) {
                                _itoa(byteValor, pcRX, 16);
                                m_pActuarDlg->m_strEditRX += pcRX;
                                m_pActuarDlg->m_strEditRX += " ";
                        }
                }
        }

        if (m_bOnActuarDlg) {
                        if (m_pActuarDlg->m_bEnableIn) {
                                m_pActuarDlg->UpdateData(FALSE);
                        }
        }
        return 0L;
}

LRESULT CScopeView::OnScopeErrorAccesPort(WPARAM wParam, LPARAM lParam)
{
        // Mensaje recibido por la existencia de errores procedentes de
        // la clase RS232
        char pcMensaje[100];

        wsprintf(pcMensaje,"Error de CRS232 (%d, %d)",wParam, lParam);
        MessageBox(pcMensaje, AfxGetAppName(), MB_OK | MB_ICONSTOP);

        return 0L;
}
```

```
void CScopeView::ScopeDetectarTrama(STrama* pTrama, BYTE byteValor)
{
	//Detecta la trama de inicio o final
	if ((byteValor == pTrama->bytePrimero) || pTrama->bByte1) {
		if (!pTrama->bByte1) {//1ª vez
				pTrama->bByte1 = TRUE;
			}
			else { //2ª, 3ª o 4ª vez
				if ((byteValor == pTrama->byteSegundo) || pTrama->bByte2) {
					if (!pTrama->bByte2) {//1ª vez
						pTrama->bByte2 = TRUE;
					}
					else {  //2º o 3ª vez
						if ((byteValor == pTrama->byteTercero) ||
                                                                pTrama->bByte3)
                                                                {
                                                        if (!pTrama->bByte3) {//1ª vez
                                                                pTrama->bByte3 = TRUE;
                                                        }
                                                        else {//2ª vez
                                                                if (byteValor ==
                                                                        pTrama->byteCuarto) {
                                                                        pTrama->bByte4 = TRUE;
                                                                //Trama inicio o final
                                                                detectada
                                                                        }
                                                                else {
                                                                      pTrama->bByte1 = FALSE;
                                                                      pTrama->bByte2 = FALSE;
                                                                      pTrama->bByte3 = FALSE;
                                                                }
                                                        }
                                                }
                                                else {
                                                        pTrama->bByte1 = FALSE;
                                                        pTrama->bByte2 = FALSE;
                                                }
                                        }
                                }
                                else {
                                        pTrama->bByte1 = FALSE;
                                }
                        }
                }
}

//////////////////////////////////////////////////////////////////////////
// Funciones para el registro de Windows

void CScopeView::ScopeAbrirFicheroIni()
{
	CScopeApp* pApp = (CScopeApp*) AfxGetApp();

	m_comRs232.nPuerto  = pApp->GetProfileInt(s_profileHeadingCom,
		s_profileCom, 0);     //Com1, valor por defecto si no encuentra el otro
	m_comRs232.nBaudios = pApp->GetProfileInt(s_profileHeadingCom,
		s_profileBaudios, 3); //9600baudios
	m_comRs232.nParidad = 0;      //Sin paridad
	m_comRs232.nBitsCar = 4;      //8Bits/Byte
	m_comRs232.nBitsParada =      0;      //1bit
	m_comRs232.nControlFlujo =    0;      //Sin control de flujo
}

void CScopeView::ScopeGuardarFicheroIni()
{
	CScopeApp* pApp = (CScopeApp*) AfxGetApp();
	pApp->WriteProfileInt(s_profileHeadingCom, s_profileCom, m_comRs232.nPuerto);
       pApp->WriteProfileInt(s_profileHeadingCom, s_profileBaudios, m_comRs232.nBaudios);

}
```

6.2.6 CLASE CSCOPEDOC

```
// scopeDoc.h : interface of the CScopeDoc class
//
/////////////////////////////////////////////////////////////////////////////

#if !defined(AFX_SCOPEDOC_H__DA985EE4_C77B_11D3_BB39_ACF5B442421C__INCLUDED_)
#define AFX_SCOPEDOC_H__DA985EE4_C77B_11D3_BB39_ACF5B442421C__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000


class CScopeDoc : public CDocument
{
// Variables que almacenan los datos recibidos de la tarjeta de adquisición de datos
public:
	double* m_pVin;                     // Para la vista. En voltios. ¡¡ 8KBy !!
	int m_nVinPuntos;                   // Número de puntos de m_pVin

	CByteArray m_arrayMuestras;   // Almacena las muestras recibidas
	int m_nArrayPuntos;                 // Número de muestras

private:
	static double s_dValueGanancia[];    // V / div
	static double s_dValueDivisor[];            // ms / div

public:
	typedef struct {              // Almacena el estado de un canal
		int nGanancia;
		int nDivisor;
		BOOL bACDCneg;
		BOOL bEnable;
	} SCanalDoc;

	SCanalDoc* m_pCanalDoc;       // Almacena los estados de los canales

public:
	void ScopeCrearVin(); // Crea el vector para la vista
	void ScopeDecodificarTramaRX(int nCanal);

protected: // create from serialization only
	CScopeDoc();
	DECLARE_DYNCREATE(CScopeDoc)

// Attributes
public:

// Operations
public:

// Overrides
	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CScopeDoc)
	public:
	virtual BOOL OnNewDocument();
	virtual void Serialize(CArchive& ar);
	virtual BOOL OnOpenDocument(LPCTSTR lpszPathName);
	//}}AFX_VIRTUAL
```

```
// Implementation
public:
	virtual ~CScopeDoc();
#ifdef _DEBUG
	virtual void AssertValid() const;
	virtual void Dump(CDumpContext& dc) const;
#endif

protected:

// Generated message map functions
protected:
	//{{AFX_MSG(CScopeDoc)
	afx_msg void OnUpdateFileSave(CCmdUI* pCmdUI);
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately before the
previous line.

#endif // !defined(AFX_SCOPEDOC_H__DA985EE4_C77B_11D3_BB39_ACF5B442421C__INCLUDED_)
```

// scopeDoc.cpp : implementation of the CScopeDoc class

```
//

#include "stdafx.h"
#include "scope.h"

#include "scopeDoc.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CScopeDoc
//int  CScopeDoc::s_nMaxPoint = 1024;
double CScopeDoc::s_dValueDivisor[2] = {0.012, 1};
double CScopeDoc::s_dValueGanancia[8] = {0.5, 1, 2, 5, 10, 20, 50, 100};

IMPLEMENT_DYNCREATE(CScopeDoc, CDocument)

BEGIN_MESSAGE_MAP(CScopeDoc, CDocument)
	//{{AFX_MSG_MAP(CScopeDoc)
	ON_UPDATE_COMMAND_UI(ID_FILE_SAVE, OnUpdateFileSave)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CScopeDoc construction/destruction

CScopeDoc::CScopeDoc()
{
}

CScopeDoc::~CScopeDoc()
{
}
```

```
BOOL CScopeDoc::OnNewDocument()
{
	TRACE("En CScopeV1Doc::OnNewDocument()\n");

	if (!CDocument::OnNewDocument())
		return FALSE;
	// (SDI documents will reuse this document)

	m_nArrayPuntos = 1024;                      //Valor máximo
	m_arrayMuestras.SetSize(m_nArrayPuntos);
	// Se utiliza una plantilla que tiene la propiedad de
	// ser serializada para posteriormente guardarla o cargarla del disco.

	m_nVinPuntos = 0;
	m_pVin = new double [m_nArrayPuntos - 12];// Puntero para la vista. En voltios

	m_pCanalDoc = new SCanalDoc [4];

	// Valores por omisión de los canales
	m_pCanalDoc[0].nGanancia = 0;
	m_pCanalDoc[0].nDivisor =     0;
	m_pCanalDoc[0].bACDCneg =     FALSE;
	m_pCanalDoc[0].bEnable =      FALSE;

	m_pCanalDoc[1].nGanancia = 0;
	m_pCanalDoc[1].nDivisor =     0;
	m_pCanalDoc[1].bACDCneg =     FALSE;
	m_pCanalDoc[1].bEnable =      FALSE;



	m_pCanalDoc[2].nGanancia = 0;
	m_pCanalDoc[2].nDivisor =     0;
	m_pCanalDoc[2].bACDCneg =     FALSE;
	m_pCanalDoc[2].bEnable =      FALSE;

	m_pCanalDoc[3].nGanancia = 0;
	m_pCanalDoc[3].nDivisor =     0;
	m_pCanalDoc[3].bACDCneg =     FALSE;
	m_pCanalDoc[3].bEnable =      FALSE;

	return TRUE;
}

/////////////////////////////////////////////////////////////////////////////
// CScopeDoc serialization

void CScopeDoc::Serialize(CArchive& ar)
{
	m_arrayMuestras.Serialize(ar);
	// Es la clase base de la plantilla la que se encarga
	// de la serialización
}

/////////////////////////////////////////////////////////////////////////////
// CScopeDoc diagnostics

#ifdef _DEBUG
void CScopeDoc::AssertValid() const
{
	CDocument::AssertValid();
}

void CScopeDoc::Dump(CDumpContext& dc) const
{
	CDocument::Dump(dc);
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CScopeDoc commands

void CScopeDoc::OnUpdateFileSave(CCmdUI* pCmdUI)
{
	pCmdUI->Enable(IsModified());
	// Habilita el botón guardar si el documento ha sufrido cambios
}
```

```
BOOL CScopeDoc::OnOpenDocument(LPCTSTR lpszPathName)
{
	if (!CDocument::OnOpenDocument(lpszPathName))
		return FALSE;

	//By guardados = 4 By de estado+(m_nIndiceRX - 7) By de datos+4 By trama final
	m_nVinPuntos = m_arrayMuestras.GetUpperBound() - 12;
	ScopeCrearVin();
	UpdateAllViews(NULL); // Actualiza la vista
	return TRUE;
}


////////////////////////////////////////////////////////////////////////////
// Codificación y creación del vector de la vista

void CScopeDoc::ScopeCrearVin()
{
	// Crea el vector con el puntero m_pVin con doubles.
	// Los valores están en V / ms y los utilizara la vista
	// para dibujar los canales en OnDraw y OnPrint
	ScopeDecodificarTramaRX(0);   // Se decodifican los bytes
	ScopeDecodificarTramaRX(1);   // de estado de cada uno de
	ScopeDecodificarTramaRX(2);   // los canales
	ScopeDecodificarTramaRX(3);

	int nNumCanales = 1;
	if (m_pCanalDoc[1].bEnable) {
		nNumCanales = 2;
	}
	if (m_pCanalDoc[3].bEnable) {
		nNumCanales = 4;
	}

	if (nNumCanales == 1) {
		for (int i = 0; i < m_nVinPuntos; i++) {
			m_pVin[i] = ((((double) m_arrayMuestras[i + 4]) * 5 / 255) - 2.5)/
				(s_dValueGanancia[(m_pCanalDoc[0].nGanancia)] *
				 s_dValueDivisor[(m_pCanalDoc[0].nDivisor)]);
		}
	}
	else {
		if (nNumCanales == 2) {
			int nVinPuntos = m_nVinPuntos / 2;
			for (int i = 0; i < nVinPuntos; i++) {
			m_pVin[2 * i] = ((((double) m_arrayMuestras[2*i+4])*5/255)-
				2.5) / (s_dValueGanancia[(m_pCanalDoc[0].nGanancia)] *
				s_dValueDivisor[(m_pCanalDoc[0].nDivisor)]);
			m_pVin[2 * i + 1] = ((((double) m_arrayMuestras[2*i+5])*5/255)-
				2.5) / (s_dValueGanancia[(m_pCanalDoc[1].nGanancia)] *
				s_dValueDivisor[(m_pCanalDoc[1].nDivisor)]);
			}
		}
		else {
			int nVinPuntos = m_nVinPuntos / 4;
			for (int i = 0; i < nVinPuntos; i++) {
			m_pVin[4 * i] = ((((double) m_arrayMuestras[4*i+4])*5 / 255)-
				2.5) / (s_dValueGanancia[(m_pCanalDoc[0].nGanancia)] *
				s_dValueDivisor[(m_pCanalDoc[0].nDivisor)]);
			m_pVin[4 * i + 1] = ((((double) m_arrayMuestras[4*i+5])*5/ 255)-
				2.5) / (s_dValueGanancia[(m_pCanalDoc[1].nGanancia)] *
				s_dValueDivisor[(m_pCanalDoc[1].nDivisor)]);
			m_pVin[4 * i + 2] = ((((double) m_arrayMuestras[4*i+6])*5/ 255)-
				2.5) / (s_dValueGanancia[(m_pCanalDoc[2].nGanancia)] *
				s_dValueDivisor[(m_pCanalDoc[2].nDivisor)]);
			m_pVin[4 * i + 3] = ((((double) m_arrayMuestras[4*i+7])*5/ 255)-
				2.5) / (s_dValueGanancia[(m_pCanalDoc[3].nGanancia)] *
				s_dValueDivisor[(m_pCanalDoc[3].nDivisor)]);
			}
		}
	}
}
```

```
void CScopeDoc::ScopeDecodificarTramaRX(int nCanal)
{
	// Decodifica los Bytes de estado situados después
	// de la trama de inicio
	BYTE byteEstado;

	byteEstado = m_arrayMuestras[nCanal];		//	Miro ACDC
	if (byteEstado & 0x01) {			//	0000 0001
		m_pCanalDoc[nCanal].bACDCneg = TRUE;
	}
	else {
		m_pCanalDoc[nCanal].bACDCneg = FALSE;
	}

	byteEstado = m_arrayMuestras[nCanal];
	BYTE byteAux;					// Miro G3
	if (byteEstado & 0x02) {			// 0000 0010
							// G3 = 1
		byteAux = 0x04;				// 0000 0100
	}
	else {						// G3 = 0
		byteAux = 0x00;
	}
	byteEstado = m_arrayMuestras[nCanal];		// Miro G2
	if (byteEstado & 0x04) {			// 0000 0100
							// G2 = 1
		byteAux = byteAux | 0x02;		// 0000 0010
	}
	else {						// G2 = 0
		byteAux = byteAux & 0xFD;		// 1111 1101
	}
	byteEstado = m_arrayMuestras[nCanal];		// Miro G1
	if (byteEstado & 0x08) {			// 0000 1000
							// G1 = 1
		byteAux = byteAux | 0x01;		// 0000 0001
	}
	else {
							// G1 = 0
		byteAux = byteAux & 0xFE;		// 1111 1110
	}
	m_pCanalDoc[nCanal].nGanancia = (int) byteAux;

	byteEstado = m_arrayMuestras[nCanal];		// Miro FAC!
	if (byteEstado & 0x10) {			// 0001 0000
							// FAC! = 1 => * 1
		m_pCanalDoc[nCanal].nDivisor = 1;    // Sin divisor
	}
	else {
							// FAC! = 0 => * 0.012
		m_pCanalDoc[nCanal].nDivisor = 0;    // Con divisor
	}

	byteEstado = m_arrayMuestras[nCanal];		// Miro CANAL
	if (byteEstado & 0x20) {			// 0010 0000
		m_pCanalDoc[nCanal].bEnable = TRUE;
	}
	else {
		m_pCanalDoc[nCanal].bEnable = FALSE;
	}

}
```

6.2.7 CLASE CCONTROLDLG

```
#if !defined(AFX_CONTROLDLG_H__DA985EF0_C77B_11D3_BB39_ACF5B442421C__INCLUDED_)
#define AFX_CONTROLDLG_H__DA985EF0_C77B_11D3_BB39_ACF5B442421C__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// ControlDlg.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// CControlDlg dialog

#define WM_GOODBYE_CONTROL WM_USER + 5

class CControlDlg : public CDialog
{
// Construction
public:
        CControlDlg(CWnd* pParent = NULL);          // standard constructor
        CControlDlg(CView* pView);
        BOOL Create();
private:
        CView* m_pView;

// Para las barras de deslizamiento
public:
        int m_nTrackbarCanal1;
        int m_nTrackbarCanal2;
        int m_nTrackbarCanal3;
        int m_nTrackbarCanal4;
        int m_nTrackbarTimeDiv;
private:
        static double s_dValueCanal[];              // V / div
        static double s_dValueTimeDiv[];            // ms /div

// Para la transmisión
public:
        BYTE* m_pTXCadena;                          // Puntero para transmisión
private:
        void MiApply();                             // Realiza una transmisión
        void ScopeCodificarTramaTX(int nCanal);
        BOOL* m_pTXCambio;                          // Cambios en el diálogo
        int m_nEstadoAnteriorCanales;
        int m_nTimer;                               // Identificador del timer
        static BYTE s_byteMascara[];         // Para codificar V / div
        static BYTE s_byteCanal[];                  // Para codificar el canal

// Para actualizar la vista después de OpenDocument
public:
        void OnApplyPublic();

// Dialog Data
        //{{AFX_DATA(CControlDlg)
        enum { IDD = IDD_CONTROL_DLG };
        int             m_nCanales;
        int             m_nAcDcC1;
        int             m_nAcDcC2;
        int             m_nAcDcC3;
        int             m_nAcDcC4;
        BOOL    m_bInteractivo;
        BOOL    m_bControlManual;
        //}}AFX_DATA
// Overrides
        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CControlDlg)
        protected:
        virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
        //}}AFX_VIRTUAL
```

```
// Implementation
protected:

	// Generated message map functions
	//{{AFX_MSG(CControlDlg)
	afx_msg void OnApply();
	afx_msg void OnCanal1();
	afx_msg void OnCanal1_2();
	afx_msg void OnCanal1_2_3_4();
	virtual void OnCancel();
	virtual void OnOK();
	afx_msg void OnHScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar);
	afx_msg void OnVScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar);
	virtual BOOL OnInitDialog();
	afx_msg void OnScopeDisparo();
	afx_msg void OnScopeInteractivo();
	afx_msg void OnTimer(UINT nIDEvent);
	afx_msg void OnAcdcC1();
	afx_msg void OnAcdcC2();
	afx_msg void OnAcdcC3();
	afx_msg void OnAcdcC4();
	afx_msg void OnControlManual();
	afx_msg void OnAcC1();
	afx_msg void OnAcC2();
	afx_msg void OnAcC3();
	afx_msg void OnAcC4();
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()

};

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately before the
previous line.

#endif // !defined(AFX_CONTROLDLG_H__DA985EF0_C77B_11D3_BB39_ACF5B442421C__INCLUDED_)
```

```
// ControlDlg.cpp : implementation file
//

#include "stdafx.h"
#include "scope.h"
#include "ControlDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CControlDlg dialog

double CControlDlg::s_dValueCanal[14] = {0.005, 0.01, 0.02, 0.05, 0.1,
                              0.2, 0.5, 1, 2, 5, 10, 20, 50, 100};                // En V
double CControlDlg::s_dValueTimeDiv[7] = {0.05, 0.1, 0.2, 0.5, 1, 2, 5};  // En ms

BYTE CControlDlg::s_byteMascara[14] = {0xFF, 0xF7, 0xFB, 0xF3, 0xFD, 0xF5, 0xF9,
                              0xF1, 0xEB, 0xE3, 0xED, 0xE5, 0xE9, 0xE1};
BYTE CControlDlg::s_byteCanal[4] = {0x3F, 0x7F, 0xBF, 0xFF};

CControlDlg::CControlDlg(CWnd* pParent /*=NULL*/)
        : CDialog(CControlDlg::IDD, pParent)
{
        //{{AFX_DATA_INIT(CControlDlg)
        m_nCanales = -1;
        m_nAcDcC1 = -1;
        m_nAcDcC2 = -1;
        m_nAcDcC3 = -1;
        m_nAcDcC4 = -1;
        m_bInteractivo = FALSE;
        m_bControlManual = FALSE;
        //}}AFX_DATA_INIT
        m_pView = NULL;
}

CControlDlg::CControlDlg(CView* pView)
{
        // Constructor no modal
        m_pView = pView;

        m_pTXCambio = new BOOL[4];
        m_pTXCambio[0] = FALSE;
        m_pTXCambio[1] = FALSE;
        m_pTXCambio[2] = FALSE;
        m_pTXCambio[3] = FALSE;

        m_pTXCadena = new BYTE[2];

        m_nEstadoAnteriorCanales = 0;
}
```

```
void CControlDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CControlDlg)
	DDX_Radio(pDX, IDC_CANALES, m_nCanales);
	DDX_Radio(pDX, IDC_ACDC_C1, m_nAcDcC1);
	DDX_Radio(pDX, IDC_ACDC_C2, m_nAcDcC2);
	DDX_Radio(pDX, IDC_ACDC_C3, m_nAcDcC3);
	DDX_Radio(pDX, IDC_ACDC_C4, m_nAcDcC4);
	DDX_Check(pDX, IDC_INTERACTIVO, m_bInteractivo);
	DDX_Check(pDX, IDC_CONTROL_MANUAL, m_bControlManual);
	//}}AFX_DATA_MAP
	if (pDX->m_bSaveAndValidate) {
	CSliderCtrl* pSlideCanal1 =(CSliderCtrl*) GetDlgItem(IDC_TRACKBAR_CANAL1);
	m_nTrackbarCanal1 = pSlideCanal1->GetPos();

	CSliderCtrl* pSlideCanal2 =(CSliderCtrl*) GetDlgItem(IDC_TRACKBAR_CANAL2);
	m_nTrackbarCanal2 = pSlideCanal2->GetPos();

	CSliderCtrl* pSlideCanal3 =(CSliderCtrl*) GetDlgItem(IDC_TRACKBAR_CANAL3);
	m_nTrackbarCanal3 = pSlideCanal3->GetPos();

	CSliderCtrl* pSlideCanal4 =(CSliderCtrl*) GetDlgItem(IDC_TRACKBAR_CANAL4);
	m_nTrackbarCanal4 = pSlideCanal4->GetPos();

	CSliderCtrl* pSlideTimeDiv =(CSliderCtrl*) GetDlgItem(IDC_TRACKBAR_TIMEDIV);
	m_nTrackbarTimeDiv = pSlideTimeDiv->GetPos();
	}
}

BOOL CControlDlg::Create()
{
	return CDialog::Create(CControlDlg::IDD);
}

BEGIN_MESSAGE_MAP(CControlDlg, CDialog)
	//{{AFX_MSG_MAP(CControlDlg)
	ON_BN_CLICKED(IDAPPLY, OnApply)
	ON_BN_CLICKED(IDC_CANALES, OnCanal1)
	ON_BN_CLICKED(IDC_RADIO2, OnCanal1_2)
	ON_BN_CLICKED(IDC_RADIO3, OnCanal1_2_3_4)
	ON_WM_HSCROLL()
	ON_WM_VSCROLL()
	ON_BN_CLICKED(IDC_DISPARO, OnScopeDisparo)
	ON_BN_CLICKED(IDC_INTERACTIVO, OnScopeInteractivo)
	ON_WM_TIMER()
	ON_BN_CLICKED(IDC_ACDC_C1, OnAcdcC1)
	ON_BN_CLICKED(IDC_ACDC_C2, OnAcdcC2)
	ON_BN_CLICKED(IDC_ACDC_C3, OnAcdcC3)
	ON_BN_CLICKED(IDC_ACDC_C4, OnAcdcC4)
	ON_BN_CLICKED(IDC_CONTROL_MANUAL, OnControlManual)
	ON_BN_CLICKED(IDC_RADIO23, OnAcC1)
	ON_BN_CLICKED(IDC_RADIO24, OnAcC2)
	ON_BN_CLICKED(IDC_RADIO25, OnAcC3)
	ON_BN_CLICKED(IDC_RADIO26, OnAcC4)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()
```

```
/////////////////////////////////////////////////////////////////////////////
// CControlDlg message handlers

BOOL CControlDlg::OnInitDialog()
{
	// Inicialización del diálogo
	CString strTextCanal1;
	CSliderCtrl* pSlideCanal1 = (CSliderCtrl*) GetDlgItem(IDC_TRACKBAR_CANAL1);
	pSlideCanal1->SetRange(0, 13);
	pSlideCanal1->SetPos(m_nTrackbarCanal1);
	strTextCanal1.Format("%4.3f", s_dValueCanal[pSlideCanal1->GetPos()]);
	SetDlgItemText(IDC_STATIC_TRACK_CANAL1, strTextCanal1);

	CString strTextCanal2;
	CSliderCtrl* pSlideCanal2 = (CSliderCtrl*) GetDlgItem(IDC_TRACKBAR_CANAL2);
	pSlideCanal2->SetRange(0, 13);
	pSlideCanal2->SetPos(m_nTrackbarCanal2);
	strTextCanal2.Format("%4.3f", s_dValueCanal[pSlideCanal2->GetPos()]);
	SetDlgItemText(IDC_STATIC_TRACK_CANAL2, strTextCanal2);

	CString strTextCanal3;
	CSliderCtrl* pSlideCanal3 = (CSliderCtrl*) GetDlgItem(IDC_TRACKBAR_CANAL3);
	pSlideCanal3->SetRange(0, 13);
	pSlideCanal3->SetPos(m_nTrackbarCanal3);
	strTextCanal3.Format("%4.3f", s_dValueCanal[pSlideCanal3->GetPos()]);
	SetDlgItemText(IDC_STATIC_TRACK_CANAL3, strTextCanal3);

	CString strTextCanal4;
	CSliderCtrl* pSlideCanal4 = (CSliderCtrl*) GetDlgItem(IDC_TRACKBAR_CANAL4);
	pSlideCanal4->SetRange(0, 13);
	pSlideCanal4->SetPos(m_nTrackbarCanal4);
	strTextCanal4.Format("%4.3f", s_dValueCanal[pSlideCanal4->GetPos()]);
	SetDlgItemText(IDC_STATIC_TRACK_CANAL4, strTextCanal4);

	CString strTextTimeDiv;
	CSliderCtrl* pSlideTimeDiv = (CSliderCtrl*) GetDlgItem(IDC_TRACKBAR_TIMEDIV);
	pSlideTimeDiv->SetRange(0, 6);
	pSlideTimeDiv->SetPos(m_nTrackbarTimeDiv);
	strTextTimeDiv.Format("%3.2f", s_dValueTimeDiv[pSlideTimeDiv->GetPos()]);
	SetDlgItemText(IDC_STATIC_TRACK_TIMEDIV, strTextTimeDiv);

	return CDialog::OnInitDialog();
}

void CControlDlg::OnCancel()
{
	if (m_pView != NULL) {
			// caso no modal -- no llamar a OnCancel de la clase de base
		m_pView->PostMessage(WM_GOODBYE_CONTROL, IDCANCEL);
	}
	else {
		CDialog::OnCancel(); // caso modal
	}
}

void CControlDlg::OnOK()
{
	if (m_pView != NULL) {
			// Caso no modal -- no llamar a OnOK de la clase de base
		UpdateData(TRUE);
		m_pView->PostMessage(WM_GOODBYE_CONTROL, IDOK);
	}
	else {
		CDialog::OnOK();	// Caso modal
	}
}

void CControlDlg::OnApply()
{
	// Envia un mensaje a la vista para que actualice los
	// datos de las barras de deslizamiento
	if (m_pView != NULL) {// Caso no modal --
		UpdateData(TRUE);
		m_pView->PostMessage(WM_GOODBYE_CONTROL, IDAPPLY);
	}
}
```

```
void CControlDlg::OnCanal1()
{
	// Al pulsar boton de radio Canal 1
	if (m_nEstadoAnteriorCanales == 1) {		// Inhabilitar Canal 2
		m_pTXCadena[0] = 0x40;			// 0100 0000
		MiApply();
	}
	else {
		if (m_nEstadoAnteriorCanales == 3) { // Inhabilitar Canales 2, 3 y 4
			m_pTXCadena[0] = 0xC0;		// 1100 0000
			MiApply();
		}
	}
	m_pTXCambio[0] = TRUE;
	m_nEstadoAnteriorCanales = 0;
}

void CControlDlg::OnCanal1_2()
{
	// Al pulsar boton de radio Canal 1 y 2
	if (m_nEstadoAnteriorCanales == 3) {		//Inhabilitar Canales 2, 3 y 4
		m_pTXCadena[0] = 0xC0;			//1100 0000
		MiApply();
	}
	m_nEstadoAnteriorCanales = 1;
	m_pTXCambio[1] = TRUE;
}

void CControlDlg::OnCanal1_2_3_4()
{
	// Al pulsar boton de radio Canal 1, 2, 3 y 4
	m_nEstadoAnteriorCanales = 3;
	m_pTXCambio[3] = TRUE;
}

void CControlDlg::OnAcdcC1()
{
	m_pTXCambio[0] = TRUE;
}

void CControlDlg::OnAcdcC2()
{
	m_pTXCambio[1] = TRUE;
}

void CControlDlg::OnAcdcC3()
{
	m_pTXCambio[2] = TRUE;
}

void CControlDlg::OnAcdcC4()
{
	m_pTXCambio[3] = TRUE;
}
void CControlDlg::OnAcC1()
{
	m_pTXCambio[0] = TRUE;
}


void CControlDlg::OnAcC2()
{
	m_pTXCambio[1] = TRUE;
}

void CControlDlg::OnAcC3()
{
	m_pTXCambio[2] = TRUE;
}

void CControlDlg::OnAcC4()
{
	m_pTXCambio[3] = TRUE;
}
```

```
void CControlDlg::OnHScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar)
{
	// Para la barra de deslizamiento ms / div
	CSliderCtrl* pSlide = (CSliderCtrl*) pScrollBar;
	CString strText;

	strText.Format("%3.2f", s_dValueTimeDiv[pSlide->GetPos()]);
	SetDlgItemText(IDC_STATIC_TRACK_TIMEDIV, strText);

	OnApply();
}

void CControlDlg::OnVScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar)
{
	// Para la barras de deslizamiento V / div
	CSliderCtrl* pSlide = (CSliderCtrl*) pScrollBar;
	CString strText;

	//Cuatro barras de seguimiento están enviando mensajes
	switch(pScrollBar->GetDlgCtrlID()) {
	case IDC_TRACKBAR_CANAL1:
		strText.Format("%4.3f", s_dValueCanal[pSlide->GetPos()]);
		SetDlgItemText(IDC_STATIC_TRACK_CANAL1, strText);

		m_pTXCambio[0] = TRUE;
		OnApply();
		break;
	case IDC_TRACKBAR_CANAL2:
		strText.Format("%4.3f", s_dValueCanal[pSlide->GetPos()]);
		SetDlgItemText(IDC_STATIC_TRACK_CANAL2, strText);

		m_pTXCambio[1] = TRUE;
		OnApply();
		break;
	case IDC_TRACKBAR_CANAL3:
		strText.Format("%4.3f", s_dValueCanal[pSlide->GetPos()]);
		SetDlgItemText(IDC_STATIC_TRACK_CANAL3, strText);

		m_pTXCambio[2] = TRUE;
		OnApply();
		break;
	case IDC_TRACKBAR_CANAL4:
		strText.Format("%4.3f", s_dValueCanal[pSlide->GetPos()]);
		SetDlgItemText(IDC_STATIC_TRACK_CANAL4, strText);

		m_pTXCambio[3] = TRUE;
		OnApply();
		break;
	}
}

void CControlDlg::OnControlManual()
{
	// Habilita el control manual
	UpdateData(TRUE);
	if (m_bControlManual) {
		m_pTXCadena[0] = 0x10;
	}
	else {
		m_pTXCadena[0] = 0x00;
	}
	MiApply();
}

void CControlDlg::OnScopeDisparo()
{
	// Para realizar el muestreo
	UpdateData(TRUE);
	if (m_bControlManual) {
		m_pTXCadena[0] = 0x18;
		MiApply();
	}
}
```

```
////////////////////////////////////////////////////////////////////////////
// Funciones para la transmisión
void CControlDlg::OnScopeInteractivo()
{
	// Cuando esta habilitado este diálogo a la vez que
	// controla la vista hace lo mismo con la tarjeta de adquisición
	// de datos
	UpdateData(TRUE);
	if (m_bInteractivo) {
		// Se grea un temporizador que en cada time-out enviara
		// a la tarjeta una By para cambiar su estado si es necesario
		m_nTimer = SetTimer(1, 1000 /*ms*/, NULL);
		ASSERT(m_nTimer != 0);
	}
	else {  // Destruye el timer
		KillTimer(m_nTimer);
	}
}
void CControlDlg::MiApply()
{
	// Envía el mensaje a la vista para que
	// ésta realice una transmisión
	if (m_pView != NULL) {// Caso no modal --
		UpdateData(TRUE);
		m_pView->PostMessage(WM_GOODBYE_CONTROL, IDC_MIAPPLY);
	}
}
void CControlDlg::OnTimer(UINT nIDEvent)
{
	// Se entra aquí cada Time-out, de esta forma se evita
	// Enviar una cantidad excesiva de By.
	UpdateData(TRUE);
	if (m_nCanales ==0) {
		if (m_pTXCambio[0]){
			ScopeCodificarTramaTX(0);		// Codificar el By a enviar
			MiApply();				// Se envía el By
		}
	}
	if (m_nCanales == 1) {
		if (m_pTXCambio[0]){
			ScopeCodificarTramaTX(0);
			MiApply();
		}
		if (m_pTXCambio[1]) {
			ScopeCodificarTramaTX(1);
			MiApply();
		}
	}
	if (m_nCanales == 2) {
		if (m_pTXCambio[0]){
			ScopeCodificarTramaTX(0);
			MiApply();
		}
		else {
			if (m_pTXCambio[1]) {
				ScopeCodificarTramaTX(1);
				MiApply();
			}
			else {
				if (m_pTXCambio[2]) {
					ScopeCodificarTramaTX(2);
					MiApply();
				}
				else {
					if (m_pTXCambio[3]) {
						ScopeCodificarTramaTX(3);
						MiApply();
					}
				}
			}
		}
	}
	m_pTXCambio[0] = FALSE;		// Se resetean los cambios
	m_pTXCambio[1] = FALSE;
	m_pTXCambio[2] = FALSE;
	m_pTXCambio[3] = FALSE;
}
```

```
void CControlDlg::ScopeCodificarTramaTX(int nCanal)
{
	// Se codifica un By que posteriormente se enviara,
	// Para ello se leen los estados de las barras de
	// deslizamiento
	BYTE cByteTX = 0xFF;

	UpdateData(TRUE);

	int* pAcDc = new int[4];
	int* pTrackbar = new int[4];

	pAcDc[0] = m_nAcDcC1;
	pAcDc[1] = m_nAcDcC2;
	pAcDc[2] = m_nAcDcC3;
	pAcDc[3] = m_nAcDcC4;

	pTrackbar[0] = m_nTrackbarCanal1;
	pTrackbar[1] = m_nTrackbarCanal2;
	pTrackbar[2] = m_nTrackbarCanal3;
	pTrackbar[3] = m_nTrackbarCanal4;

	if (pAcDc[nCanal] == 0) {		//DC--- .0 = 0
		cByteTX = cByteTX & 0xFE;	//FE == 1111 1110;
	}
	else {							//AC--- .0 = 1
		cByteTX = cByteTX | 0x01;
	}

	cByteTX = cByteTX | 0x1E;		//0001 1110, FAC! ,G1, G2, G3 = 1
	cByteTX = cByteTX & s_byteMascara[pTrackbar[nCanal]];

	cByteTX = cByteTX | 0xE0;		//1110 0000, CH1 = CH2 = E =1
	cByteTX = cByteTX & s_byteCanal[nCanal];

	m_pTXCadena[0] = cByteTX;

	// Seguidamente se presentan las máscaras de codificación
	// BYTE CControlDlg::s_byteMascara[14] = {		0xFF/*1111 1111*/,
	//										0xF7/*1111 0111*/,
	//										0xFB/*1111 1011*/,
	//										0xF3/*1111 0011*/,
	//										0xFD/*1111 1101*/,
	//										0xF5/*1111 0101*/,
	//										0xF9/*1111 1001*/,
	//										0xF1/*1111 0001*/,
	//										0xEB/*1110 1011*/,
	//										0xE3/*1110 0011*/,
	//										0xED/*1110 1101*/,
	//										0xE5/*1110 0101*/,
	//										0xE9/*1110 1001*/,
	//										0xE1/*1110 0001*/};
	// BYTE CControlDlg::s_byteCanal[4] = {		0x3F/*0011 1111*/,
	//										0x7F/*0111 1111*/,
	//										0xBF/*1011 1111*/,
	//										0xFF/*1111 1111*/};
}

//////////////////////////////////////////////////////////////////////////
// Otra función

void CControlDlg::OnApplyPublic()
{
	// La utiliza el documento para actualizar la vista
	OnApply();
}
```

6.2.8 CLASE CACTUARDLG

```
#if !defined(AFX_ACTUARDLG_H__DA985EF1_C77B_11D3_BB39_ACF5B442421C__INCLUDED_)
#define AFX_ACTUARDLG_H__DA985EF1_C77B_11D3_BB39_ACF5B442421C__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// ActuarDlg.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// CActuarDlg dialog

#define WM_GOODBYE_ACTUAR WM_USER + 6

class CActuarDlg : public CDialog
{
// Construction
public:
	CActuarDlg(CWnd* pParent = NULL);	// Standard constructor

	CActuarDlg(CView* pView);				// COnstructor no modal
	BOOL Create();					// Para crear el diálogo no modal
// Para la transmisión de datos
public:
	BYTE* m_pCadenaTX;				// Vector para la transmisión
	int m_nNumBytesTX;				// Número de By máximos a transmitir

private:
	void ScopeCodificarTramaTX();		// Codifica la trama a enciar
	void ScopeSimular();			// Para la simulación
	CView* m_pView;					// Puntero de la vista

	BYTE* m_pEstadoCH;				// Estado de los canales
	BOOL* m_pEnableCH;				// Enable de los canales
public:
// Dialog Data
	//{{AFX_DATA(CActuarDlg)
	enum { IDD = IDD_ACTUAR_DLG };
	int		m_nCanal;
	int		m_nDivisor;
	int		m_nGanancia;
	int		m_nDCAC;
	CString	m_strEditRX;
	CString	m_strEditTX;
	BOOL		m_nEnable;
	BOOL		m_bConTramas;
	BOOL		m_bEnableIn;
	BOOL		m_bEnableOut;
	BOOL		m_bSimula;
	//}}AFX_DATA


// Overrides
	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CActuarDlg)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);	// DDX/DDV support
	//}}AFX_VIRTUAL
```

```
// Implementation
protected:

	// Generated message map functions
	//{{AFX_MSG(CActuarDlg)
	afx_msg void OnApply();
	virtual void OnCancel();
	virtual void OnOK();
	afx_msg void OnEnviar();
	afx_msg void OnLimpiarin();
	afx_msg void OnLimpiarout();
	afx_msg void OnEnablein();
	afx_msg void OnEnableout();
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately before //
the previous line.

#endif // !defined(AFX_ACTUARDLG_H__DA985EF1_C77B_11D3_BB39_ACF5B442421C__INCLUDED_)

// ActuarDlg.cpp : implementation file
//

#include "stdafx.h"
#include "scope.h"
#include "ActuarDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CActuarDlg dialog

CActuarDlg::CActuarDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CActuarDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CActuarDlg)
	m_nCanal = -1;
	m_nDivisor = -1;
	m_nGanancia = -1;
	m_nDCAC = -1;
	m_strEditRX = _T("");
	m_strEditTX = _T("");
	m_nEnable = FALSE;
	m_bConTramas = FALSE;
	m_bEnableIn = FALSE;
	m_bEnableOut = FALSE;
	m_bSimula = FALSE;
	//}}AFX_DATA_INIT
	m_pView = NULL;
}
```

```
CActuarDlg::CActuarDlg(CView* pView)
{
	m_pView = pView;

	m_nNumBytesTX = 1024;
	m_pCadenaTX = new BYTE[m_nNumBytesTX];

	m_pEstadoCH = new BYTE[4];
	m_pEstadoCH[0] = 0x00;// Estado de reset de los canales
	m_pEstadoCH[1] = 0x00;
	m_pEstadoCH[2] = 0x00;
	m_pEstadoCH[3] = 0x00;

	m_pEnableCH = new BOOL[4];
	m_pEnableCH[0] = FALSE;	// Canales inhabilitados
	m_pEnableCH[1] = FALSE;
	m_pEnableCH[2] = FALSE;
	m_pEnableCH[3] = FALSE;
}
BOOL CActuarDlg::Create()
{
	return CDialog::Create(CActuarDlg::IDD);
}

void CActuarDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CActuarDlg)
	DDX_Radio(pDX, IDC_CANAL, m_nCanal);
	DDX_Radio(pDX, IDC_DIVISOR, m_nDivisor);
	DDX_Radio(pDX, IDC_GANANCIA, m_nGanancia);
	DDX_Radio(pDX, IDC_DCAC, m_nDCAC);
	DDX_Text(pDX, IDC_EDIT_RX, m_strEditRX);
	DDX_Text(pDX, IDC_EDIT_TX, m_strEditTX);
	DDX_Check(pDX, IDC_ENABLE, m_nEnable);
	DDX_Check(pDX, IDC_CONTRAMAS, m_bConTramas);
	DDX_Check(pDX, IDC_ENABLEIN, m_bEnableIn);
	DDX_Check(pDX, IDC_ENABLEOUT, m_bEnableOut);
	DDX_Check(pDX, IDC_SIMULA, m_bSimula);
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CActuarDlg, CDialog)
	//{{AFX_MSG_MAP(CActuarDlg)
	ON_BN_CLICKED(IDAPPLY, OnApply)
	ON_BN_CLICKED(IDENVIAR, OnEnviar)
	ON_BN_CLICKED(IDC_LIMPIARIN, OnLimpiarin)
	ON_BN_CLICKED(IDC_LIMPIAROUT, OnLimpiarout)
	ON_BN_CLICKED(IDC_ENABLEIN, OnEnablein)
	ON_BN_CLICKED(IDC_ENABLEOUT, OnEnableout)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CActuarDlg message handlers
void CActuarDlg::OnCancel()
{
	if (m_pView != NULL) {
			// Caso no modal -- no llamar a OnCancel de la clase de base
		m_pView->PostMessage(WM_GOODBYE_ACTUAR, IDCANCEL);
	}
	else {
		CDialog::OnCancel(); // Caso modal
	}
}
void CActuarDlg::OnOK()
{
	if (m_pView != NULL) {
			// Caso no modal -- no llamar a OnOK de la clase de base
		UpdateData(TRUE);
		m_pView->PostMessage(WM_GOODBYE_ACTUAR, IDOK);
	}
	else {
		CDialog::OnOK();	// Caso modal
	}
}
```

```
void CActuarDlg::OnLimpiarin()
{
	// Limpiar cuadro de edición de entrada
	m_strEditRX = "";
	UpdateData(FALSE);
}
void CActuarDlg::OnLimpiarout()
{
	// Limpiar cuadro de edición de salida
	m_strEditTX = "";
	UpdateData(FALSE);
}
void CActuarDlg::OnEnablein()
{
	// Actualiza los datos del diálogo
	UpdateData(TRUE);
}
void CActuarDlg::OnEnableout()
{
	// Actualiza los datos del diálogo
	UpdateData(TRUE);
}
void CActuarDlg::OnEnviar()
{
	// Prepara los datos que serán enviados
	if (m_pView != NULL) {		// Caso no modal
		char pcTX[10];

		UpdateData(TRUE);			// Recoger datos del diálogo
		if (m_bSimula) {			// Simulación
			m_nNumBytesTX = 1024; // Envio 1KBy
			ScopeSimular();
			if (m_bEnableOut) {
				_itoa(m_pCadenaTX[0], pcTX, 16);
				m_strEditTX = pcTX;
				m_strEditTX += " ";
				for (int i = 1; i < m_nNumBytesTX; i++) {
					_itoa(m_pCadenaTX[i], pcTX, 16);
					m_strEditTX += pcTX;
					m_strEditTX += " ";
				}
				m_strEditTX += "n=";
				_itoa(m_nNumBytesTX, pcTX, 16);
				m_strEditTX += pcTX;
			}
		}
		else {				// Sin simulación
			m_nNumBytesTX = 1;		// Envio un By
			if (m_nCanal == 0) {
				m_pCadenaTX[0] = m_pEstadoCH[0];
			}
			else {
				if (m_nCanal == 1) {
					m_pCadenaTX[0] = m_pEstadoCH[1];
				}
				else {
					if (m_nCanal == 2) {
						m_pCadenaTX[0] = m_pEstadoCH[2];
					}
					else {
						m_pCadenaTX[0] = m_pEstadoCH[3];
					}
				}
			}
			if (m_bEnableOut) {
				_itoa(m_pCadenaTX[0], pcTX, 16);
				m_strEditTX = pcTX;
				m_strEditTX += " ";
			}
		}
		UpdateData(FALSE);		// Actualizar diálogo y variables mienbro
					// para la vista
		m_pView->PostMessage(WM_GOODBYE_ACTUAR, IDENVIAR);
	}
}
}
```

```
/////////////////////////////////////////////////////////////////////////////
// CActuarDlg funciones de apoyo

void CActuarDlg::OnApply()
{
	// Crea la trama a enviar.
	// Si se presiona el botón enviar, se transmite lo visualizado en salida
	if (m_pView != NULL) {		// Caso no modal
		char pcTX[10];

		UpdateData(TRUE);
		ScopeCodificarTramaTX();

		if (m_bEnableOut) {		// Habilitada la ventana de salida
			if (m_bSimula) {	// Caso simulacion
				_itoa(m_pEstadoCH[0], pcTX, 16);
				m_strEditTX = pcTX;
				m_strEditTX += " ";

				for (int i = 1; i < 4;i++) {
					_itoa(m_pEstadoCH[i], pcTX, 16);
					m_strEditTX += pcTX;
					m_strEditTX += " ";
				}
			}
			else {			// Inhabilitada la ventana de salida
				if (m_nCanal == 0) {
					_itoa(m_pEstadoCH[0], pcTX, 16);
				}
				else {
					if (m_nCanal == 1) {
						_itoa(m_pEstadoCH[1], pcTX, 16);
					}
					else {
						if (m_nCanal == 2) {
							_itoa(m_pEstadoCH[2], pcTX, 16);
						}
						else {
							_itoa(m_pEstadoCH[3], pcTX, 16);
						}
					}
				}
			m_strEditTX = pcTX;
			m_strEditTX += " ";
			}
		}
		UpdateData(FALSE);		// Actualizar diálogo
	}
}
```

```
void CActuarDlg::ScopeCodificarTramaTX()
{
	// Recoge los datos del diálogo y los codifica en un By
	BYTE cByteTX = 0xFF;
	if (m_nDCAC == 0) {                          // DC--- .0 = 0
		cByteTX = cByteTX & 0xFE;            // FE == 1111 1110;
	}
	else {                                       // AC--- .0 = 1
		cByteTX = cByteTX | 0x01;
	}
                                                     // 100--- .3 = 1, .2 = 1, .1 = 1
	cByteTX = cByteTX | 0x0E;                    // 0000 1110;
	if (m_nGanancia == 0) {                      // 0.5--- .3 = 0, .2 = 0, .1 = 0
		cByteTX = cByteTX & 0xF1;            // 1111 0001;
	}
	else {
		if (m_nGanancia == 1) {              // 1--- .3 = 1, .2 = 0, .1 = 0
			cByteTX = cByteTX & 0xF9;    // 1111 1001;
		}
		else {
			if (m_nGanancia == 2) {              // 2--- .3 = 0, .2 = 1, .1 =
0
				cByteTX = cByteTX & 0xF5;    // 1111 0101;
			}
			else {
				if (m_nGanancia == 3) {      // 5--- .3 = 1, .2 = 1, .1 =
0
					cByteTX = cByteTX & 0xFD;    // 1111 1101;
				}
				else {
					if (m_nGanancia == 4) {//10--- .3 = 0, .2 = 0, .1 =
1
						cByteTX = cByteTX & 0xF3;    // 1111 0011;
					}
					else {
						if (m_nGanancia ==5) {//20---.3=1,.2=0, .1 =
1
							cByteTX = cByteTX & 0xFB;    // 1111
1011;
						}
						else {
							if (m_nGanancia == 6) {
								// 50--- .3 = 0, .2 = 1, .1 =
1
								cByteTX = cByteTX & 0xF7;
								// 1111 0111;
							}
						}
					}
				}
			}
		}
	}

	if (m_nDivisor == 0) {                       // Con divisor
		cByteTX = cByteTX & 0xEF;            // 1110 1111
	}
	else {                                       // Sin divisor
		cByteTX = cByteTX | 0x10;            // 0001 0000
	}

	if (m_nEnable == TRUE) {
		cByteTX = cByteTX | 0x20;            // 0010 0000
	}
	else {
		cByteTX = cByteTX & 0xDF;            // 1101 1111
	}

	cByteTX = cByteTX | 0xC0;                    // 1100 0000
	if (m_nCanal == 0) {
		cByteTX = cByteTX & 0x3F;            // 0011 1111
		m_pEstadoCH[0] = cByteTX;
		if (m_nEnable) {
			m_pEnableCH[0] = TRUE;
		}

		else {
```

```
                        m_pEnableCH[0] = FALSE;
                }
        }
        else {
                if (m_nCanal == 1) {
                        cByteTX = cByteTX & 0x7F;            // 0111 1111
                        m_pEstadoCH[1] = cByteTX;
                        if (m_nEnable) {
                                m_pEnableCH[1] = TRUE;
                        }
                        else {
                                m_pEnableCH[1] = FALSE;
                        }
                }
                else {
                        if (m_nCanal == 2) {
                                cByteTX = cByteTX & 0xBF;     // 1011 1111
                                m_pEstadoCH[2] = cByteTX;
                                if (m_nEnable) {
                                        m_pEnableCH[2] = TRUE;
                                }
                                else {
                                        m_pEnableCH[2] = FALSE;
                                }
                        }
                        else {
                                m_pEstadoCH[3] = cByteTX;
                                if (m_nEnable) {
                                        m_pEnableCH[3] = TRUE;
                                }
                                else {
                                        m_pEnableCH[3] = FALSE;
                                }
                        }
                }
        }
}
```

```
void CActuarDlg::ScopeSimular()
{
	// Función para crear un trama si se desea simular.
	// Para observar el resultado es necesario cortocircuitar
	// los pines TX y RX del puerto serie configurado
	m_pCadenaTX[0] = 0xFB;		// Trama de inicio
	m_pCadenaTX[1] = 0x04;
	m_pCadenaTX[2] = 0xFB;
	m_pCadenaTX[3] = 0x04;
	m_pCadenaTX[4] = m_pEstadoCH[0];	// By de estado
	m_pCadenaTX[5] = m_pEstadoCH[1];
	m_pCadenaTX[6] = m_pEstadoCH[2];
	m_pCadenaTX[7] = m_pEstadoCH[3];

	double pi = 3.1415926535;

	if (m_pEnableCH[3]) {		// 4 canales
		BYTE byteAux = 0x00;
		BYTE byteAux2 = 0x0a;
		int nContador = 0;
		for (int i = 0; i < ((m_nNumBytesTX - 12) / 4); i++){
			m_pCadenaTX[4 * i + 8] = (BYTE) (128 + 120 * sin(10 * i * 2 * pi /
				(m_nNumBytesTX - 12)) );
			m_pCadenaTX[4 * i + 9] = (BYTE) (128 + 60 * sin(25 * i * 2 * pi /
				 (m_nNumBytesTX - 12)) );
			m_pCadenaTX[4 * i + 10] = byteAux;
			byteAux += 3;
			m_pCadenaTX[4 * i + 11] = byteAux2;
			if (nContador == 2) {
				nContador = 0;
				if (byteAux2 == 0x0a) {
					byteAux2 = 0xF0;
				}
				else {
					byteAux2 = 0x0a;
				}
			}
			nContador++;
		}
	}
	else {
		if (m_pEnableCH[1]) { // 2 canal
			for (int i = 0; i < ((m_nNumBytesTX - 12) / 2); i++){
				m_pCadenaTX[2 * i + 8] = (BYTE) (128 + 120 * sin(10 * i * 2
					* pi / (m_nNumBytesTX - 12)) );
				m_pCadenaTX[2 * i + 9] = (BYTE) (128 + 120 * sin(15 * i * 2
					* pi / (m_nNumBytesTX - 12)) );
			}
		}
		else {				// 1 canal
			for (int i = 0; i < (m_nNumBytesTX - 12); i++){
				m_pCadenaTX[i + 8] = (BYTE) (128 + 110 * sin(50 * i * 2
					* pi / (m_nNumBytesTX - 12)) );
			}
		}
	}
	m_pCadenaTX[m_nNumBytesTX - 4] = 0x04;	// Trama final
	m_pCadenaTX[m_nNumBytesTX - 3] = 0xFB;
	m_pCadenaTX[m_nNumBytesTX - 2] = 0x04;
	m_pCadenaTX[m_nNumBytesTX - 1] = 0xFB;
}
```

6.2.9 CLASE CCONFIGCOMDLG

```
#if !defined(AFX_CONFIGCOMDLG_H__4AFC49A5_CC0F_11D3_BB39_96F49A117913__INCLUDED_)
#define AFX_CONFIGCOMDLG_H__4AFC49A5_CC0F_11D3_BB39_96F49A117913__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// ConfigComDlg.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// CConfigComDlg dialog

class CConfigComDlg : public CDialog
{
// Construction
public:
	CConfigComDlg(CWnd* pParent = NULL);   // standard constructor

// Dialog Data
	//{{AFX_DATA(CConfigComDlg)
	enum { IDD = IDD_CONFIGCOM_DLG };
	int		m_nBaudios;
	int		m_nCom;
	//}}AFX_DATA


// Overrides
	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CConfigComDlg)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
	//}}AFX_VIRTUAL

// Implementation
protected:

	// Generated message map functions
	//{{AFX_MSG(CConfigComDlg)
	virtual void OnOK();
	virtual void OnCancel();
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately before the
previous line.

#endif // !defined(AFX_CONFIGCOMDLG_H__4AFC49A5_CC0F_11D3_BB39_96F49A117913__INCLUDED_)

// ConfigComDlg.cpp : implementation file
//

#include "stdafx.h"
#include "scope.h"
#include "ConfigComDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
```

```
/////////////////////////////////////////////////////////////////////////////
// CConfigComDlg dialog


CConfigComDlg::CConfigComDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CConfigComDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CConfigComDlg)
	m_nBaudios = -1;
	m_nCom = -1;
	//}}AFX_DATA_INIT
}


void CConfigComDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CConfigComDlg)
	DDX_Radio(pDX, IDC_BAUDIOS, m_nBaudios);
	DDX_Radio(pDX, IDC_COM, m_nCom);
	//}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(CConfigComDlg, CDialog)
	//{{AFX_MSG_MAP(CConfigComDlg)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CConfigComDlg message handlers

void CConfigComDlg::OnOK()
{
	CDialog::OnOK();
}

void CConfigComDlg::OnCancel()
{
	CDialog::OnCancel();
}
```

7 BIBLIOGRAFÍA

Título: **CIRCUITOS Y SEÑALES:**
Introducción a los circuitos lineales y de acoplamiento
Autores: R. E. Thomas
A. J. Rosa
Editorial: REVERTÉ

Título: **Diseño electrónico. CIRCUITOS Y SISTEMAS**
Autores: C. J. Savant, Jr
Martin S. Roden
Gordon L. Carpenter
Editorial: ADDISON-WESLEY IBEROAMERICANA

Título: **EL LENGUAJE DE PROGRAMACIÓN C**
Autores: Brian W. Kernighan
Dennis M. Ritchie
Editorial: Prentice Hall

Título: **C++ Guía de autoenseñanza**
Autor: Herbert Schildt
Editorial: Osborne/McGraw-Hill

Título: **Programación avanzada con Microsoft Visual C++**
Autor: David J. Kruglinski
Editorial: McGraw-Hill

Apuntes de las siguientes asignaturas:

Informática Industrial I
Informática Industrial II
Tecnología Electrónica I
Ingeniería de Equipos Electrónicos I
Sistemas de Telecomando
Sistemas Digitales II