ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE INGENIERÍA

DISEÑO Y CONSTRUCCIÓN DE UN SISTEMA PORTÁTIL DE TRANSMISIÓN Y RECEPCIÓN PARA CODIFICACIÓN DE LÍNEA, CONTROLADO POR PC VÍA PUERTO USB

PROYECTO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

WASHINGTON LEONARDO CHUQUI CHICAIZA
GUIDO FERNANDO IBARRA PASTÁS

DIRECTOR: ING. FERNANDO FLORES

Quito, Septiembre 2004

DECLARACIÓN

Nosotros, Washington Leonardo Chuqui Chicaiza y Guido Fernando Ibarra Pastás, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedemos el derecho de propiedad intelectual correspondiente a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su reglamento y por la normativa institucional vigente.

Washington Leonardo Chuqui Chicaiza

Guido Fernando Ibarra Pastás

CERTIFICACIÓN

Certifico que el siguiente trabajo fue realizado por Washington Leonardo Chuqui Chicaiza y Guido Fernando Ibarra Pastás, bajo mi supervisión.

Ing. Fernando Flores
DIRECTOR DEL PROYECTO

AGRADECIMIENTO

Gracias a mi familia, quienes de una u otra forma me ayudarón en la terminación de mis estudios universitarios; a la Escuela Politécnica Nacional por los conocimientos adquiridos que me ayudarán en mi vida profesional.

A mis amigos, Fabián Bautista, Xavier Zurita, Juan Carlos Lugmaña, Patricio Usiña, Nancy González, Nubia Pazmiño, Tannia Castillo y Adriana Molina con quienes compartí momentos inolvidables de mi vida.

A quienes nos ayudaron en el desarrollo del proyecto como lo es Mery González, Mélida Masabanda, Ing. Fernando Vásquez e Ing. Pablo Salinas.

A Vilma Montenegro, una persona especial en mi vida, quien estuvo a mi lado en los buenos y en los malos momentos.

Gracias al Ing. Fernando Flores, director del proyecto, que bajo su dirección se llegó a la culminación del mismo.

Guido Fernando Ibarra Pastás

AGRADECIMIENTO

Te agradezco Dios por todo lo vivido, pues gracias a tí tengo la mejor familia que un ser humano puede tener, me permitiste conocer a tantas personas que ayudaron en mi formación; por lo que viviré, pues cada día representa un nuevo reto y del cual se aprenderá algo nuevo que engrandecerá mi espíritu.

A mis amigos, Javier, Juan Carlos, Fabián, Patricio, Adriana, Nancy, Tannia, Nubia amigos que siempre estuvieron junto a mi en las buenas y en las malas, en la etapa más bonita de estudiante, la etapa de la Universidad. Como olvidarme de Mery y Mélida, amigas que nos han ayudado de distintas formas en el avance del presente proyecto. Fernando un agradecimiento especial por ser un ejemplo de paciencia y un buen Amigo.

Gracias a nuestro Director Fernando Flores por su acertada dirección en la culminación de este proyecto.

Washington Leonardo Chuqui Chicaiza

DEDICATORIA

Dedico este trabajo a Dios por darme unos Padres como José Laureano Ibarra Ortiz y María Elena Pastás Gutierrez, quienes supieron guiarme para alcanzar cada uno de los objetivos trazados en mi vida; a unas hermanas como Alexandra y Miriam quienes me ayudaron en cada una de mis estapas de estudiante; y por darme la fuerza necesaria para superar aquellas trabas que la vida presenta.

Guido Fernando Ibarra Pastás

DEDICATORIA

Este trabajo esta dedicado a las personas más importantes en mi vida, por todo el amor que me han brindado, por ser mis amigos, consejeros, un ejemplo de fortaleza, dedicación y optimismo, a mis padres *Washington Leonardo Chuqui Toapanta* y *Magdalena del Consuelo Chicaiza Morales.*

Washington Leonardo Chuqui Chicaiza

ÍNDICE PÁG.

RESUMEN xi

PRESENTACIÓN xii

CAPÍTULO I LA INTERFAZ USB Y LOS CÓDIGOS DE LÍNEA 1

1.1 INTRODUCCIÓN 2

1.2 INTERFAZ USB 2

1.2.1 VENTAJAS 4

1.2.2 DESVENTAJAS 7

1.2.3 REQUERIMIENTOS MÍNIMOS DEL PC 8

1.2.4 LOS COMPONENTES 8

1.2.5 EL PUERTO USB 9

1.2.6 TOPOLOGÍA DEL BUS 10

1.2.7 VELOCIDAD 10

1.3 FUNDAMENTOS DE LA TRANSFERENCIA 11

1.3.1 MANEJO DE LOS DATOS EN EL BUS 11

1.3.2 VELOCIDAD DEL HOST Y VELOCIDAD DEL BUS 12

1.3.3 ELEMENTOS DE UNA TRANSFERENCIA 13

1.3.4 TIPOS DE TRANSFERENCIAS 14

1.3.4.1 Transferencias de Control 14

1.3.4.2 Transferencias Bulk 14

1.3.4.3 Transferencias de Interrupción 15

1.3.4.4 Transferencias Isócronas 15

1.3.5 INICIO DE UNA TRANSFERENCIA 16

1.3.6 TRANSACCIONES 16

1.3.6.1 Fases de una Transacción 17

1.3.7 SINCRONIZACIÓN 20

1.3.8 TRANSACCIONES DIVIDIDAS 20

1.3.9 CONTROL DE FLUJO Y CHEQUEO DE ERRORES 22

1.3.9.1 Handshaking 22

1.3.9.2 Chequeo de errores 24

1.4 ENUMERACIÓN 25

1.4.1 PROCESO 25

1.4.2 PASOS DE LA ENUMERACIÓN 26

1.4.3 DESCONEXIÓN DEL DISPOSITIVO 30

1.4.4 TIPOS DE DESCRIPTORES 30
1.4.4.1 Descriptor del Dispositivo 30
1.4.4.2 Descriptor Device_Qualifier 30
1.4.4.3 Descriptor de Configuración 31
1.4.4.4 Descriptor Other_Speed_Configuration 31
1.4.4.5 Descriptor de la Interfaz 32
1.4.4.6 Descriptor del Endpoint 32
1.4.4.7 Descriptor de Secuencia 32
1.4.5 LAS PETICIONES 32
1.5 ADMINISTRACIÓN DE LA ENERGÍA 34
1.5.1 OPCIONES DE ENERGÍA 34
1.5.2 VOLTAJES 34
1.5.3 PERIFÉRICOS QUE PUEDEN UTILIZAR ENERGÍA DEL BUS 35
1.5.4 NECESIDADES DE ENERGÍA 35
1.6 CODIFICACIÓN DE LOS DATOS 35
1.6.1 SINCRONIZACIÓN PERMANENTE 36
1.6.1.1 Bit de relleno 36
1.6.1.2 Campo SYNC 36
1.6.1.3 Señal de fin de paquete 37
1.7 FORMATO DEL PAQUETE 37
1.7.1 CAMPO SYNC 37
1.7.2 CAMPO DE IDENTIFICADOR DEL PAQUETE 37
1.7.3 CAMPO DE DIRECCIÓN 38
1.7.4 CAMPO DEL ENDPOINT 38
1.7.5 CAMPO DEL NÚMERO DE TRAMA 38
1.7.6 CAMPO DE DATOS 39
1.7.7 CAMPO CRC 39
1.7.8 RETARDO ENTRE PAQUETES 39
1.8 TRANSMISORES-RECEPTORES Y SEÑALES 39
1.8.1 SEGMENTOS DE CABLE 39
1.8.2 CABLES 41
1.8.2.1 Conductores 41
1.8.2.2 Conectores 43
1.8.2.3 Cables desmontables y prisioneros 45
1.8.2.4 Longitud del cable 45

1.9 APLICACIONES DEL INTERFAZ USB 46
1.9.1 TECLADOS, RATONES Y JOYSTICKS 46
1.9.2 IMPRESORAS 46
1.9.3 CÁMARAS FOTOGRÁFICAS Y SCANNERS 46
1.9.4 APLICACIONES DE AUDIO 46
1.9.5 MÓDEMS 47
1.10 CÓDIGOS DE LÍNEA 47
1.10.1 PARÁMETROS 47
1.10.1.1 Eficiencia espectral 47
1.10.1.2 Sincronismo y Transparencia 47
1.10.1.3 Capacidad de detección de errores 48
1.10.1.4 Baja probabilidad de errores 48
1.10.2 CLASES DE CÓDIGOS DE LÍNEA 48
1.11 DEFINICIÓN DE TÉRMINOS 51

CAPÍTULO II DISEÑO Y CONSTRUCCIÓN DEL DISPOSITIVO 53

2.1 REQUERIMIENTOS 53
2.1.1 CÓDIGO DE LÍNEA 54
2.1.2 VELOCIDAD DE TRANSMISIÓN 54
2.2 DISEÑO 54
2.2.1 BLOQUE DE TRANSMISIÓN Y RECEPCIÓN DE LA SEÑAL DE DATOS 55
2.2.2 BLOQUE DE TRANSMISIÓN Y RECEPCIÓN DE LA CONFIGURACIÓN DEL DISPOSITIVO 57
2.2.3 BLOQUE DE PROCESAMIENTO DE LA SEÑAL DE DATOS CODIFICADOS 60
2.2.4 BLOQUE DE PROCESAMIENTO DE LA SEÑAL DE DATOS A DECODIFICAR 61
2.2.5 BLOQUE GENERADOR DEL RELOJ DE SINCRONISMO Y SINCRONIZACIÓN DE LOS DATOS 63
2.2.6 UNIDAD CENTRAL DE PROCESOS 71
2.2.6.1 Código AMI 74
2.2.6.2 Código NRZ Polar 75
2.2.6.3 Código RZ Polar 76
2.2.6.4 Código Manchester Diferencial 77
2.2.6.5 Código Bifase M 79
2.2.6.6 Código CMI 80

2.2.6.7 Código Miller..... 82
2.2.6.8 Código MLT3..... 84
2.2.6.9 Código B3ZS..... 86
2.2.6.10 Código MS43..... 88
2.2.6.11 Subrutinas..... 94

CAPÍTULO III DESARROLLO DEL SOFTWARE PARA LA INTERFAZ GRÁFICA..... 100

3.1 DESCRIPCIÓN Y REQUERIMIENTOS DEL SOFTWARE..... 101
3.2 RUTINAS DE INICIALIZACIÓN Y VERIFICACIÓN DE ESTADO DE LA CONEXIÓN CON EL CODEC..... 104
3.2.1 PRESENTACIÓN DEL SOFTWARE DESARROLLADO..... 104
3.2.2 VERIFICACIÓN DEL ESTADO DE LA CONEXIÓN..... 105
3.3 RUTINA DE CONFIGURACIÓN DEL DISPOSITIVO..... 110
3.4 RUTINA PARA ENVIAR LOS DATOS A SER CODIFICADOS Y GRAFICAR LA SEÑAL DE RELOJ, SEÑAL DE DATOS Y SEÑAL CODIFICADA..... 112
3.5 RUTINAS DE INFORMACIÓN DE LOS CÓDIGOS DE LÍNEA..... 115
3.5.1 ALGORITMO DE GENERACIÓN Y CARACTERÍSTICAS..... 116
3.5.2 APLICACIÓN VISUALIZAR LOS ESPECTROS DE POTENCIA APROXIMADOS..... 118
3.5.3 APLICACIÓN PARA CODIFICAR..... 118
3.5.4 APLICACIÓN PARA DECODIFICAR..... 119
3.6 RUTINAS DE CODIFICACIÓN Y DECODIFICACIÓN PARA LOS CÓDIGOS DE LÍNEA..... 120
3.6.1 RUTINA PARA EL CÓDIGO AMI..... 121
3.6.2 RUTINA PARA EL CÓDIGO B3ZS..... 121
3.6.3 RUTINA PARA EL CÓDIGO BIFASE M..... 122
3.6.4 RUTINA PARA EL CÓDIGO CMI..... 129
3.6.5 RUTINA PARA EL CÓDIGO HDB3..... 129
3.6.6 RUTINA PARA EL CÓDIGO MANCHESTER DIFERENCIAL..... 130
3.6.7 RUTINA PARA EL CÓDIGO MILLER..... 130
3.6.8 RUTINA PARA EL CÓDIGO MLT-3..... 131
3.6.9 RUTINA PARA EL CÓDIGO MS43..... 137
3.6.10 RUTINA PARA EL CÓDIGO NRZ POLAR..... 143
3.6.11 RUTINA PARA EL CÓDIGO RZ POLAR..... 143
3.6.12 RUTINA PARA EL CÓDIGO 4B3T..... 146

CAPÍTULO IV EVALUACIÓN DEL CODIFICADOR / DECODIFICADOR DE CÓDIGOS DE LÍNEA..... 151

4.1 DESCRIPCIÓN DE LA PARTE EXTERNA DEL DISPOSITIVO..... 152
4.2 HERRAMIENTAS UTILIZADAS EN LAS PRUEBAS DEL DISPOSITIVO..... 155
4.3 PRUEBAS DEL FUNCIONAMIENTO DE LOS CÓDIGOS DE LÍNEA..... 157
4.3.1 CÓDIGO AMI..... 157
4.3.2 CÓDIGO NRZ POLAR..... 159
4.3.3 CÓDIGO HDB3..... 160
4.3.4 CÓDIGO B3ZS..... 162
4.3.5 CÓDIGO MLT3..... 163
4.3.6 CÓDIGO RZ POLAR..... 164
4.3.7 CÓDIGO BIFASE M..... 165
4.3.8 CÓDIGO MANCHESTER DIFERENCIAL..... 167
4.3.9 CÓDIGO CMI..... 168
4.3.10 CÓDIGO MILLER..... 170
4.3.11 CÓDIGO MS43..... 171
4.3.12 CÓDIGO 4B3T..... 171
4.3.13 SEÑALES DE RELOJ..... 172
4.4 PRUEBAS DE VELOCIDADES DE TRANSMISIÓN..... 174
4.5 TIEMPOS PROMEDIOS DE PROCESAMIENTO..... 175
4.6 LIMITACIONES DEL SISTEMA..... 175
4.6.1 LIMITACIONES DEL SOFTWARE PARA CONFIGURAR EL CODEC PORTÁTIL V1.0..... 175
4.6.2 LIMITACIONES DEL HARDWARE DEL CODEC PORTÁTIL..... 176
4.7 COSTOS DE CONSTRUCCIÓN..... 178

CAPÍTULO V CONCLUSIONES Y RECOMENDACIONES..... 180

5.1 CONCLUSIONES..... 181
5.2 RECOMENDACIONES..... 183

BIBLIOGRAFÍA..... 185

ANEXOS..... 189

ÍNDICE DE FIGURAS

1.1	Algunos dispositivos USB	3
1.2	Proceso de conexión	3
1.3	Conectores USB, conector paralelo Centronics y conector Serie RS-232	6
1.4	Topología en estrella extendida	10
1.5	Tramas low y full speed	12
1.6	Conexión de dispositivos de diferentes tasas de transferencia	13
1.7	Estructura de una transferencia USB	16
1.8	Transacciones divididas entre dispositivos de diferentes velocidades	21
1.9	El Administrador de Dispositivos en Windows XP	26
1.10	Conexión entre los distintos dispositivos y su velocidad	40
1.11	Conductores de un cable USB	41
1.12	El icono identifica cable USB	41
1.13	Conector USB de la serie A y conector de la serie B	43
1.14	El conector mini-B	44
1.15	Ejemplo de tipos de Codificación	49
2.1	Diagrama de bloques del Codificador / decodificador	55
2.2	Circuito de transmisión y recepción de la señal de datos	57
2.3	Ciclos de Temporización de la FIFO	58
2.4	Circuito transmisor y receptor de la configuración del Codec	59
2.5	Circuito procesador de la señal de datos codificada	61
2.6	Circuito procesador de la señal a decodificar	63
2.7	Diagrama de flujo de la configuración de parámetros del microcontrolador 16F870	65
2.8	Diagrama de flujo del programa principal	67
2.9	Diagrama de flujo de la Rutina de Interrupción de recepción	70
2.10	Circuito de la etapa de Sincronización	71
2.11	Diagrama de flujo de la Inicialización del dispositivo	72
2.12	Diagrama de flujo de la Configuración del dispositivo	73
2.13	Diagrama de flujo de la Codificación y Decodificación AMI	75
2.14	Diagrama de flujo de la Codificación y Decodificación NRZ Polar	76
2.15	Diagrama de flujo de la Codificación y Decodificación RZ Polar	77
2.16	Diagrama de flujo de la Codificación Manchester Diferencial	78
2.17	Diagrama de flujo de la Codificación Manchester Diferencial	79

3.17	Espectro de Potencia Aproximado del Código de Línea AMI	118
3.18	Pantalla correspondiente a la aplicación para codificar	119
3.19	Pantalla correspondiente a la aplicación para decodificar	120
3.20	Diagrama de flujo de la rutina para la codificación AMI	123
3.21	Diagrama de flujo de la rutina para la decodificación AMI	124
3.22	Diagrama de flujo de la rutina para la codificación B3ZS	125
3.23	Diagrama de flujo de la rutina para la decodificación B3ZS	126
3.24	Diagrama de flujo de la rutina para la codificación Bifase M	127
3.25	Diagrama de flujo de la rutina para la decodificación Bifase M	128
3.26	Diagrama de flujo de la rutina para la codificación Manchester Diferencial	132
3.27	Diagrama de flujo de la rutina para la codificación Miller	133
3.28	Diagrama de flujo de la rutina para la codificación MLT-3	134
3.29	Diagrama de flujo de la rutina para la decodificación MLT-3	137
3.30	Diagrama de flujo de la rutina para la codificación MS43	140
3.31	Diagrama de flujo de las subrutinas para la codificación MS43	142
3.32	Diagrama de flujo de la rutina para la decodificación MS43	144
3.33	Diagrama de flujo de la rutina para la codificación NRZ Polar	145
3.34	Diagrama de flujo de la rutina para la codificación 4B3T	149
3.35	Diagrama de flujo de las subrutinas para la codificación 4B3T	150
4.1	Vista frontal del dispositivo	153
4.2	Vista Trasera del dispositivo	154
4.3	Configuración de los parámetros de comunicación del osciloscopio Tektronix TDS 220	156
4.4	Comunicación exitosa entre el computador y el osciloscopio Tektronix TDS 220	156
4.5	Pantalla principal del programa WaveStar for oscilloscopes versión 2.2	156
4.6	Codificación AMI de la letra "u" a 300 bits/seg	158
4.7	Codificación AMI de la letra "u" a 19200 bits/seg	158
4.8	Codificación NRZ Polar de la letra "U" a 300 bits/seg	159
4.9	Codificación NRZ Polar de la letra "U" a 19200 bits/seg	160
4.10	Codificación HDB3 de la letra "A" a 300 bits/seg	161
4.11	Codificación HDB3 de la letra "A" a 4800 bits/seg	161
4.12	Codificación B3ZS de la letra "A" a 300 bits/seg	162
4.13	Codificación B3ZS de la letra "A" a 4800 bits/seg	163
4.14	Codificación MLT3 de la letra "U" a 300 bits/seg	164
4.15	Codificación MLT3 de la letra "U" a 4800 bits/seg	164
4.16	Codificación RZ de la letra "U" a 300 bits/seg	165

4.17	Codificación RZ de la letra "U" a 4800 bits/seg	166
4.18	Codificación Bifase M de la letra "u" a 300 bits/seg	166
4.19	Codificación Bifase M de la letra "u" a 4800 bits/seg	167
4.20	Codificación Manchester Diferencial de la letra "U" a 300 bits/seg	168
4.21	Codificación Manchester Diferencial de la letra "U" a 4800 bits/seg	168
4.22	Codificación CMI de la letra "U" a 300 bits/seg	169
4.23	Codificación CMI de la letra "U" a 4800 bits/seg	169
4.24	Codificación Miller para letra "A" a 300 bits/seg	170
4.25	Codificación Miller para letra "A" a 4800 bits/seg	170
4.26	Codificación MS43 de la letra "A" a 300 bits/seg	171
4.27	Codificación 4B3T de la letra "A" a 300 bits/seg	172
4.28	Reloj de codificación y decodificación AMI, NRZ Polar, MLT3, B3ZS, HDB3	173
4.29	Reloj de codificación y decodificación RZ Polar, Manchester Diferencial, CMI, Miller, Bifase M	173
4.30	Reloj de codificación y decodificación MS43 y 4B3T	174

ÍNDICE DE TABLAS

1.1	Comparación de los interfaces populares de un computador	5
1.2	Índice máximo para la transferencia de datos	15
1.3	Códigos de las transacciones	17
1.4	Los tipos de transferencias y sus fases	18
1.5	Identificadores de paquete (PID)	19
1.6	Indicadores de estado de cada tipo de transacción	23
1.7	Tipos de descriptores estándares	31
1.8	Voltajes máximo y mínimo en los puertos de un concentrador	35
1.9	Tipos de campo y su propósito	38
1.10	Requisitos de cables y componentes low speed y full/high speed	42
1.11	Asignación de pines y de color para el cable y conectores USB	44
1.12	Algunas características de los códigos de línea	50
2.1	Distribución de pines de las señales del MÓDULO USBMOD21	59
2.2	Niveles de voltaje con sus correspondientes valores binarios	60
2.3	Relación de los Niveles de voltaje con los valores binarios de magnitud y signo	62
2.4	Valores para el reconocimiento del tipo de código	68
2.5	Valores de configuración enviados por la CPU	69

RESUMEN

El proyecto de titulación desarrollado consta de cinco capítulos, cuyo contenido se describe a continuación:

El capítulo I se enfoca al estudio de la interfaz USB (*Universal Serial Bus*) y los fundamentos teóricos de los códigos de línea. El conocimiento del funcionamiento del interfaz USB permitirá entender las comunicaciones que se ejecutan con los periféricos que utilizan este interfaz, así como sus ventajas y desventajas frente a los otros interfaces existentes. Adicionalmente se presenta un breve estudio de los códigos de línea que son utilizados en diferentes tipos de redes en su capa física.

El capítulo II está orientado al diseño y construcción del CODEC Portátil. El diseño contempla la justificación de cada uno de los elementos utilizados en la construcción del dispositivo.

En el capítulo III se presenta una descripción del desarrollo del *software* que se utiliza para la configuración del CODEC (codificador/decodificador), explicando cada uno de los controles de Visual Basic utilizados.

En el capítulo IV se muestra los resultados obtenidos de las pruebas realizadas al dispositivo, además se muestra cada uno de los elementos de los que está constituido externamente el CODEC, así como se indica el costo de cada uno de los elementos utilizados y el monto total del equipo implementado.

En el capítulo V se presenta las conclusiones y recomendaciones del trabajo realizado.

PRESENTACIÓN

Tomando en cuenta el avance tecnológico y el ciclo de vida útil del equipo de laboratorio de Comunicación Digital, se ha visto la necesidad de reemplazarlo por un equipo didáctico acorde con las nuevas tecnologías.

El equipo a diseñarse y construirse es un Codec Portátil, que a diferencia del anterior, utiliza un interfaz USB tanto para su configuración como para su alimentación. El interfaz USB fue seleccionado por su facilidad de uso y por las ventajas que este nos brinda para optimizar el diseño y disminuir los costos de construcción.

La configuración es visualizada en un PC gracias a una interfaz gráfica desarrollada con el lenguaje de programación Visual Basic versión 6.0, facilitando aún más la utilización del Codec Portátil. Adicionalmente el *software* desarrollado presenta tanto información como aplicaciones de cada uno de los códigos de línea implementados.

El dispositivo de codificación digital construído constituye un complemento en la enseñanza de los Códigos de Línea para la materia de Comunicación Digital, permitiendo una mejor comprensión de la teoría.

CAPÍTULO I

LA INTERFAZ USB Y LOS CÓDIGOS DE LÍNEA

LA INTERFAZ USB Y LOS CÓDIGOS DE LÍNEA

1.1 INTRODUCCIÓN

Este capítulo se enfoca al estudio de la interfaz USB (*Universal Serial Bus*) y los códigos de línea. El conocimiento del funcionamiento de la interfaz USB permitirá entender las comunicaciones que se ejecutan con los periféricos que utilizan esta interfaz, así como sus ventajas y desventajas frente a las otras interfaces existentes.

Desde hace mucho tiempo el transporte de la información de un transmisor a un receptor y viceversa, ha sido importante, es por esto que el ser humano ha implementado varias interfaces para realizar las comunicaciones ya sean a cortas o largas distancias. Una de estas interfaces es la interfaz USB, que hoy en día es uno de los más populares.

1.2 INTERFAZ USB

El bus serial universal (USB) es una interfaz rápida y flexible que permite conectar diversos dispositivos a los computadores.

La interfaz USB se utiliza con periféricos estándares como el teclado, ratón, *scanner*, unidades de disco externo, impresoras, así como también con dispositivos especializados, incluyendo los dispositivos prototipo. Hasta la fecha existen más de 250 productos individuales USB en el mercado. En la Figura 1.1 se muestran algunos de los productos.

Todo dispositivo USB tiene la capacidad de ser conectado a un computador personal en pleno funcionamiento, sin tener que reiniciarlo, la configuración del dispositivo es inmediata y transparente al usuario, luego de lo cual el dispositivo está listo para ser utilizado.

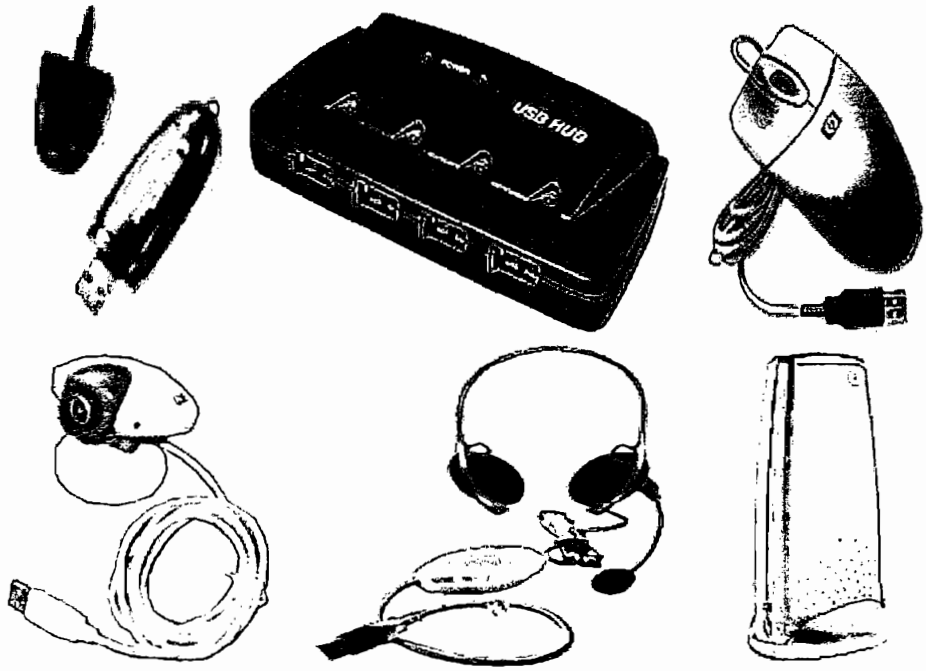


Figura 1.1 Algunos dispositivos USB[1].

El proceso de conexión del interfaz USB es sencillo como se muestra en la Figura 1.2.

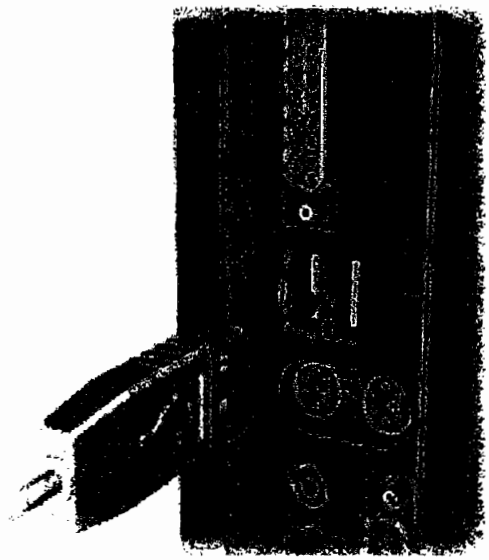


Figura 1.2 Proceso de conexión[2].

[1 y 2] Figuras tomadas de la web www.perifericosUSB.com

1.2.1 VENTAJAS

Las ventajas que presenta la interfaz USB son:

- Interfaz de muchos servicios, pues lo utilizan muchas clases de dispositivos.

- Configuración automática, pues un usuario puede conectar un periférico USB a un PC que está corriendo un sistema operativo *Windows,* el cual detecta automáticamente el periférico y carga el *software* apropiado. Si el periférico se conecta por primera vez, *Windows* puede solicitar al usuario que inserte el disco con el *software* del controlador, pero si ya se los ha utilizado previamente, la instalación es automática.

- Fácil de conectar ya que no hay necesidad de abrir el computador, debido a que un PC típico tiene por lo menos dos puertos USB. Además se puede conectar y desconectar un periférico, con el sistema y/o el periférico activos, lo cual no causa daños al PC o al periférico.

- La interfaz USB maneja la señal de Vcc y tierra, proveniente de la fuente del computador o del concentrador. Un periférico que requiere hasta 500 miliamperios puede obtener su energía del puerto USB.

- Los conectores de la interfaz USB son de tipo A en el extremo que conecta al PC y de tipo B o miniB en el otro extremo del bus que permite conectar el dispositivo, por lo que los usuarios no pueden equivocarse al insertarlos. En la figura 1.3 se muestra los conectores USB, RS-232 y el conector paralelo.

- Las velocidades de bus que soporta la interfaz USB son: *high speed* (alta velocidad) de 480 *Megabits* por segundo, *full speed* (velocidad completa) en 12 *Megabits* por segundo, y *low speed* (baja velocidad) en 1.5 *Megabits* por segundo.

INTERFAZ	FORMATO	NÚMERO DE DISPOSITIVOS (MÁXIMO)	LONGITUD (MÁXIMA, EN PIES)	VELOCIDAD (MÁXIMA, EN *BITS*/SEG)	USO TÍPICO
USB	Serie asíncrono	127	16 (96 pies con 5 concentradores)	1.5 M, 12 M, 480 M.	Ratón, teclado, disqueteras, módems.
RS 232 (EIA/TIA-232)	Serie asíncrono	2	50 – 100	20K (115 K con algún hardware)	Módem, ratón.
RS 485 (TIA/EIA-485)	Serie asíncrono	32 unidades de carga(mas de 256 dispositivos con algo de *hardware*)	4000	10 M	Adquisición de datos y sistemas de Control
IrDA	Infrarrojo serie asíncrono	2	6	115 K	Impresoras, otros periféricos del computador
Microwire	Serie asíncrono	8	10	2 M	Comunicaciones con el microcontrolador
SPI	Serie asíncrono	8	10	2.1 M	Comunicaciones con el microcontrolador
I^2C	Serie asíncrono	40	18	3.4 M	Comunicaciones con el microcontrolador
IEEE-1394 *(Firewire)*	Serie	64	15	400 M (aumento a 3.2 G con IEEE-1394b)	Vídeo, capacidad de almacenaje
IEEE-488 (GPIB)	Paralelo	15	60	8 M	Instrumentación
Ethernet	Serie	1024	1600	10 M/ 100 M/ 1G	Redes de computadoras
MIDI	Lazo de corriente serie	2(mas con flujo por modo)	50	31.5 K	Música, señales de Control
Puerto Paralelo del impresor.	Paralelo	2 (8 con ayuda de múltiples conexiones en serie)	10 – 30	8 M	Impresoras, scanner, controlador de disco duro.

Tabla 1.1 Comparación de las interfaces populares de un computador[1].

- Los periféricos *low speed* son más económicos, pues utilizan cables flexibles de menor blindaje. *Full speed* es comparable o superior a las velocidades alcanzables con los puertos serial y paralelo existentes. En la

[1] Tabla tomada del libro "*USB COMPLETE Second Edition*" de JAN AXELSON

a. Detectar cuando un dispositivo se conecta o se retira del sistema.

b. Proporcionar un mecanismo el cual permite a los controladores del software comunicarse con el *hardware* USB del computador y las aplicaciones a las cuales desean tener acceso los periféricos USB.

c. Si el sistema operativo no incluye el *driver* de un periférico específico, el vendedor del periférico tiene que proporcionar uno.

1.2.2 DESVENTAJAS

Las desventajas que se tienen al utilizar la interfaz USB son:

- La interfaz USB no incluye el soporte para el *hardware* y sistemas operativos anteriores al aparecimiento del USB, los límites de velocidad y de distancia hacen del USB poco práctico para algunas aplicaciones, y los problemas con algunos productos debido a las dificultades experimentadas por los desarrolladores de estos productos USB.

- Si se quiere conectar un periférico no USB a un puerto USB, una solución es un convertidor que traslada entre la interfaz USB a una interfaz paralela o RS-232. Si se desea utilizar un periférico USB con un PC que no soporta a la interfaz USB, la solución es agregar tarjetas de expansión USB al PC, estas tarjetas se colocan en una ranura PCI de la tarjeta madre del PC, tomando en cuenta que el sistema operativo soporte la interfaz USB.

- *High speed* del USB lo hace competitivo con el interfaz IEEE-1394 *(firewire)* de 400 Mbps, pero IEEE-1394b es un interfaz más rápido, puede operar a velocidades de 3.2 Gbps.

- Un segmento de cable puede tener una distancia de 5 metros. Otras interfaces, tales como RS-232, RS-485, y *Ethernet*, permiten cables mucho más largos. Se puede aumentar la longitud de un enlace USB en hasta 30

metros utilizando cables que enlacen 5 concentradores y un dispositivo, usando 6 segmentos de cable de 5 metros por cada uno.

- Para programar un periférico USB hay que conocer acerca de los Protocolos USB. El chip controlador maneja muchas de las comunicaciones automáticamente, pero todavía deben programarse, y esto requiere el conocimiento acerca de la escritura de los programas y las herramientas para realizar la programación. En el lado del PC, los desarrolladores del controlador del dispositivo necesitan familiarizarse con los protocolos USB y las responsabilidades del controlador.

1.2.3 REQUERIMIENTOS MÍNIMOS DEL PC

Para utilizar la interfaz USB, un PC necesita el soporte de *hardware* y *software*. El *hardware* consta de un controlador USB del *host* y de un concentrador raíz con uno o más puertos USB.

1.2.4 LOS COMPONENTES

Los componentes físicos del bus serial universal consisten de los circuitos, conectores y cables entre un *host* y uno o más dispositivos.

El *host* es el PC que contiene dos componentes: un controlador USB del *host*[1] y un concentrador[2] raíz. Éstos trabajan juntos para permitir al sistema operativo comunicarse con los dispositivos. Los formatos de datos del controlador del *host* permiten convertir los datos recibidos, al formato de los componentes del sistema operativo para que puedan ser procesados. El controlador del *host* también realiza funciones relacionadas con el manejo de las comunicaciones en el bus.

El concentrador, conjuntamente con el controlador del *host*, detecta la conexión y el retiro de dispositivos.

[1 y 2] Definición de Términos

- **Tareas del *host*:** El *host* debe estar al tanto de cuáles son los dispositivos en el bus y las capacidades de cada uno, para lo cual debe realizar las siguientes tareas:

  a. Detección de los dispositivos
  b. Manejo del flujo de datos
  c. Chequeo de errores
  d. Entrega de energía a los dispositivos
  e. Intercambio de datos con los periféricos

- **Tareas del periférico:** Las tareas son similares a las del *host*, pues cuando el *host* inicia las comunicaciones, el periférico debe responder. El periférico también tiene tareas que son únicas. Un dispositivo no puede iniciar las comunicaciones sobre el bus USB, ya que debe esperar y responder a una comunicación del *host*.

  El periférico debe realizar las siguientes tareas:

  a. Detección de las comunicaciones dirigidas al *chip*
  b. Respuesta a las peticiones estándares
  c. Chequeo de errores
  d. Manejo de la energía
  e. Intercambio de datos con el *host*

1.2.5 EL PUERTO USB

En general, un puerto de un computador es una localización direccionable que está disponible para conectar circuitos adicionales. La memoria del computador también consiste en localidades direccionables, pero la CPU tiene acceso a memoria con diversas instrucciones de máquina.

Los puertos USB comparten una sola trayectoria al *host* a través del bus. Cada controlador del *host* soporta un solo bus, o trayectoria de datos, por lo que todos

los dispositivos comparten el tiempo disponible. Solo un dispositivo, o *host*, transmite a la vez.

1.2.6 TOPOLOGÍA DEL BUS

La topología física, en el bus es una estrella extendida tal como se muestra en la figura 1.4.

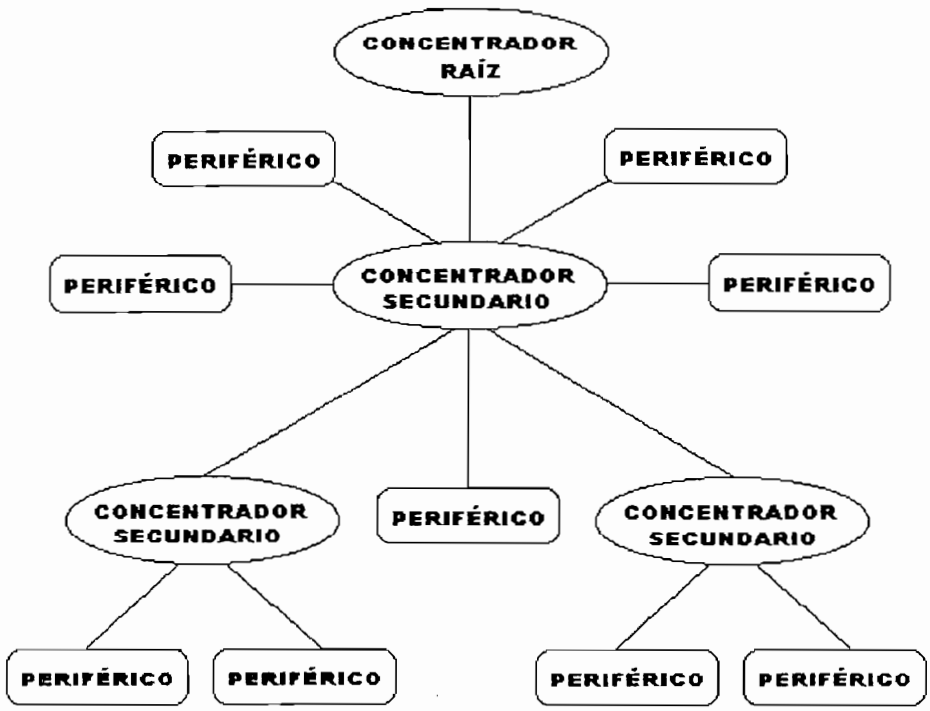


Figura 1.4 Topología en estrella extendida[1].

En la comunicación con un dispositivo USB, ni el *host*, ni el dispositivo saben si una comunicación se realizó a través de un concentrador o a través de cinco, los concentradores lo manejan automáticamente.

1.2.7 VELOCIDAD

Un controlador del dispositivo puede soportar *low* y *full speed*, o *full* y *high speed*. Virtualmente todos los concentradores soportan dispositivos con *low* y *full speed*, un periférico *low* o *full speed* puede conectarse con cualquier concentrador USB.

[1] Figura tomada del libro "*USB COMPLETE Second Edition*" de JAN AXELSON

1.3 FUNDAMENTOS DE LA TRANSFERENCIA

Las comunicaciones en el USB se dividen en dos categorías, dependiendo de si están utilizadas en la configuración o en las aplicaciones que se comunican con el dispositivo.

- **Configuración de las comunicaciones:** Durante la enumeración, el *firmware* del dispositivo responde a una serie de peticiones estándar del *host*, el dispositivo debe identificar cada petición, retornar la información solicitada, y tomar otras acciones especificadas por las peticiones.

  Al terminar la enumeración, se debe tener dos archivos disponibles: un archivo INF que identifica el nombre del fichero y la localización del *driver* del dispositivo; y el *driver* del dispositivo. Si los archivos están disponibles y el *firmware* está en orden, el proceso de enumeración es transparente para el usuario.

- **Aplicaciones de las comunicaciones:** Después de que el *host* ha intercambiado la información de la enumeración con el dispositivo y se ha asignado y cargado el *driver* del dispositivo, el dispositivo requiere envíar los datos al *buffer* de transmisión del controlador USB, leer los datos recibidos del *buffer* receptor, y completar la transferencia, asegurándose de que el dispositivo esté listo para la siguiente transferencia.

1.3.1 MANEJO DE LOS DATOS EN EL BUS

Las dos líneas de señal del USB llevan datos hacia y desde los dispositivos en el bus, el *host* se encarga de todas las transferencias y a cada transferencia se le asigna una porción de trama o micro trama como se muestra en la figura 1.5.

Para los datos *low* y *full speed*, las tramas son de un milisegundo, mientras que para los datos *high speed*, el *host* divide cada trama en ocho microtramas de 125

microsegundos. Cada trama o micro trama comienza con una referencia de sincronización de inicio de la trama.

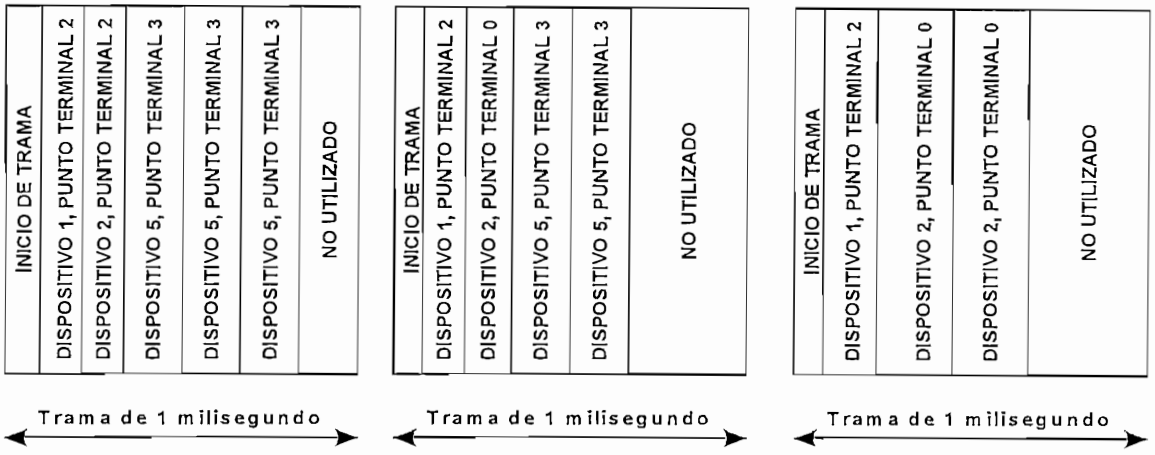


Figura 1.5 Tramas low y full speed[1].

Cada transferencia consta de una o más transacciones, dependiendo de cómo el *host* programa las transacciones y la velocidad de respuesta de un dispositivo, las transacciones de la transferencia pueden estar todas en una sola trama o microtrama, o puede ser múltiples tramas o microtramas.

Debido a que todas las transferencias comparten una trayectoria de datos, cada transacción debe tener una dirección de dispositivo, esta única dirección es asignada por el *host*, y todos los datos viajan hacia o desde el *host*.

1.3.2 VELOCIDAD DEL *HOST* Y VELOCIDAD DEL BUS

Un *host* 1.x soporta *low* y *full speed*, un *host* 2.0 con los puertos accesibles al usuario debe soportar low, *full* y *high speed.*

Cuando un concentrador 1.x recibe tráfico, cambia solamente el índice del borde de las señales para igualar la velocidad de destino, en cambio, un concentrador 2.0 convierte entre *high, low* o *full speed* según sea necesario. El tráfico en un

[1] Figura tomada del libro "*USB COMPLETE Second Edition*" de JAN AXELSON

segmento del bus es *high speed,* solo si el controlador del *host* y todos los concentradores son 2.0. En La figura 1.6 se muestra la conexión entre los diferentes dispositivos USB.

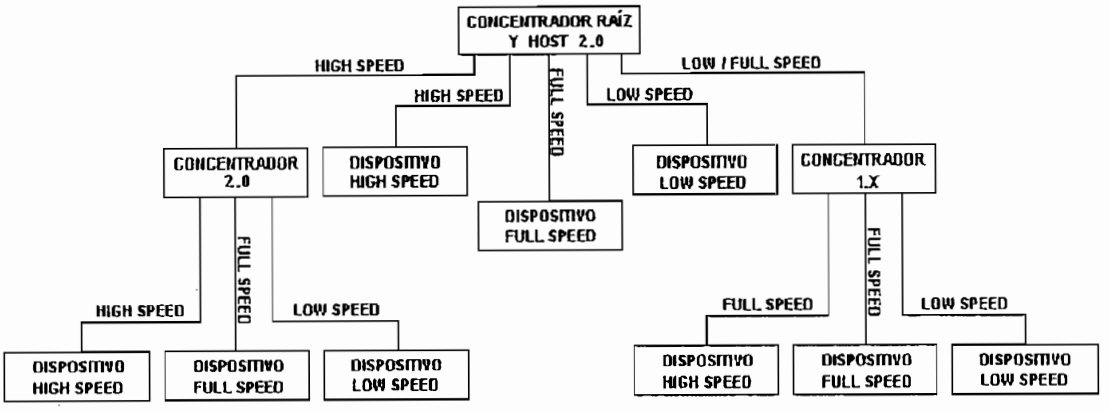


Figura 1.6 Conexión de dispositivos de diferentes tasas de transferencia[1].

1.3.3 ELEMENTOS DE UNA TRANSFERENCIA

Cada transferencia se compone de transacciones, cada transacción se compone de paquetes y cada paquete contiene información. Para entender las transacciones, los paquetes, y su contenido, se debe conocer acerca de los *endpoints* (puntos terminales) y los *pipes* (vías de comunicación).

- ***Endpoint* del dispositivo:** La especificación define a un *endpoint* de un dispositivo como " la comunicación entre el *host* y el dispositivo, en una única dirección". Esto sugiere que un *endpoint* lleva datos en una sola dirección, la dirección de cada *endpoint* consiste de un número que puede ser de 0 a 15.

  Un dispositivo *full speed* puede tener hasta 30 *endpoints* adicionales (1 a 15, cada uno soporta ambas transacciones *IN* y *OUT*) a parte del *endpoint* 0. Un dispositivo *low speed* se limita a dos *endpoints* con cualquier combinación de direcciones. Un *host* no tiene *endpoints*, pero si tiene un *buffer* de transmisión y recepción de datos. El *host* sirve como el punto de partida para comunicarse con los endpoints del dispositivo.

[1] Figura tomada del libro "*USB COMPLETE Second Edition*" de JAN AXELSON

- ***Pipes***: Los *pipes* representan la capacidad de mover datos entre el *buffer* del *host* y un *endpoint* en un dispositivo. Para que ocurra una transferencia, el *host* y el dispositivo deben establecer una vía de comunicación, la cual es establecida por el *host* poco después de cargar el sistema o la conexión del dispositivo. Cada dispositivo tiene un *pipe* de control por defecto que utiliza el *endpoint* 0.

1.3.4 TIPOS DE TRANSFERENCIAS

Las transferencias de datos que se tienen en el USB difieren en el manejo y el tipo de transferencia que el dispositivo puede soportar. Es por eso que la especificación USB tiene los siguientes tipos de trasferencias:

- Transferencias de Control
- Transferencias *Bulk*
- Transferencias de Interrupción
- Transferencias Isócronas

A continuación se describen brevemente las funciones de cada tipo de transferencia:

1.3.4.1 Transferencias de Control

Las transferencias de control son utilizadas por el *software* del USB para configurar los dispositivos cuando estos son conectados; y para llevar las peticiones que son definidas por la especificación del USB y utilizadas por el *host* para recibir información.

1.3.4.2 Transferencias Bulk

Las transferencias *bulk* son útiles para transferir datos cuando el tiempo no es crítico. Una transferencia *bulk* puede enviar cantidades grandes de datos sin saturar el bus, de manera secuencial.

1.3.4.3 Transferencias de Interrupción

Las transferencias de interrupción son útiles cuando los datos tienen que enviarse en un determinado tiempo. Las aplicaciones que utilizan este tipo de transferencia son los teclados, ratones y otros dispositivos como *joysticks* (palanca de juegos), y reportes del estado del concentrador.

Estas transferencias ocurren solo cuando el *host* sondea un dispositivo. Las transferencias de interrupciones, garantizan que el *host* solicite o envíe datos con un mínimo retardo. Una transferencia de interrupción termina por dos razones: cuando la cantidad de datos solicitada ha sido transferida, o cuando el paquete de datos contiene una cantidad menor que la máxima, incluyendo un paquete de longitud cero.

En la tabla 1.2 se muestra la cantidad de datos que cada tipo de transferencia puede llevar en cada una de las tres velocidades.

Tipo de transferencia	Máxima tasa de transferencia de datos por *endpoint* (*kilobytes*/seg. de carga útil transferida)		
	Low speed	*Full speed*	*High speed*
Control	24	832	15.872
Interrupción	0.8	64	24.576
Bulk	No permitido	1216	53.248
Isócrona		1023	24.576

Tabla 1.2 Índice máximo para la transferencia de datos[1].

1.3.4.4 Transferencias Isócronas

Las transferencias isócronas son transferencias en tiempo real que son útiles cuando los datos deben llegar a una tasa constante, o a un determinado tiempo, y puede tolerar errores ocasionales. Las aplicaciones más comunes para las transferencias isócronas incluyen voz y música codificadas que pueden ser

[1] Tabla tomada del libro "*USB COMPLETE Second Edition*" de JAN AXELSON

reproducidas en tiempo real. Aunque las transferencias isócronas puedan enviar un número fijo de *bytes* por trama, los datos no se transfieren en un número constante de *bits* por segundo.

1.3.5 INICIO DE UNA TRANSFERENCIA

Cuando un *driver* del dispositivo en el *host* desea comunicarse con un dispositivo, inicia una transferencia. Una transferencia puede ser tan corta como un *byte* de datos, o tan larga como el contenido de un archivo.

Las aplicaciones pueden solicitar datos de un dispositivo o proporcionar los datos para enviar al dispositivo. Cuando una aplicación solicita una transferencia, el sistema operativo pasa la petición al *driver* del dispositivo apropiado. El controlador del *host* entonces inicia la transferencia en el bus. La Figura 1.7 muestra la estructura de una transferencia típica.

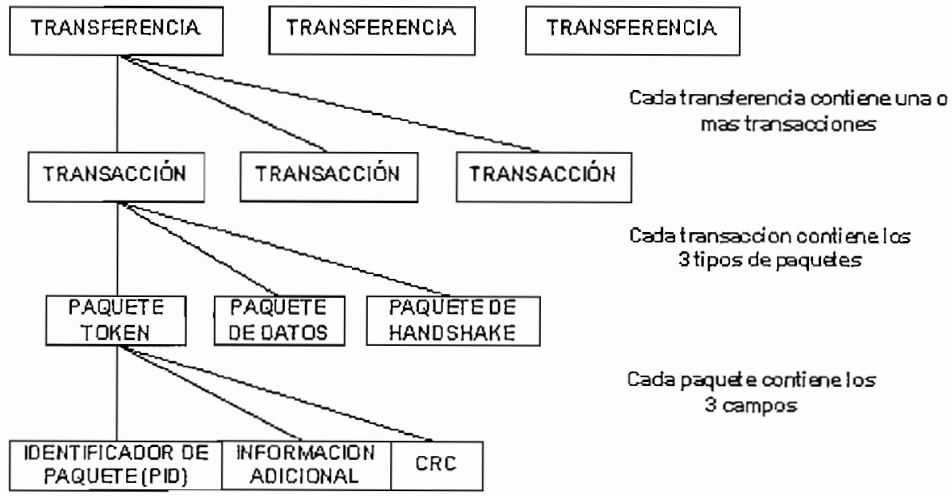


Figura 1.7 Estructura de una transferencia USB[1].

1.3.6 TRANSACCIONES

La especificación define una transacción, como la entrega de un servicio a un *endpoint*; el servicio implica el envío de información del *host* al dispositivo, o del *host* solicitando y recibiendo una parte de información del dispositivo.

[1] Figura tomada del libro "*USB COMPLETE Second Edition*" de JAN AXELSON

Cada transacción en el bus incluye un número de *endpoints* y un código que indica la dirección del flujo de datos y si la transacción está iniciando una transferencia de control. En la tabla 1.3 se muestran los códigos de las transacciones utilizadas en las transferencias.

Tipo de transacción	Fuente de datos	Tipos de Transferencia que usa este tipo de transacción	Contenido
IN	Dispositivo	Todas	Datos genéricos
OUT	*Host*	Todas	Datos genéricos
SETUP	*Host*	Control	Una petición

Tabla 1.3 Códigos de las transacciones[1].

Los tres tipos de transacción son definidos por su propósito y la dirección del flujo de datos: en la transacción *IN*, los datos recorren del periférico al *host*. En una transacción *OUT*, el recorrido de los datos va del *host* al periférico. En una transacción *SETUP*, los datos también viajan del *host* al periférico, pero inicia una Transferencia de Control, además son las únicas que los dispositivos deben identificarlas y aceptarlas siempre.

Cada transacción incluye identificación, chequeo de errores, estado, e información de control, así como cualquier dato que se intercambie. Una transferencia completa puede ocurrir sobre múltiples tramas, pero una transacción debe terminar ininterrumpidamente.

En la tabla 1.4 se enumeran los elementos que forman parte de cada una de las cuatro tipos de transferencia.

1.3.6.1 Fases de una Transacción

Cada transacción tiene tres fases que ocurren en secuencia: *Token*, Datos, y *Handshake*. Cada fase consiste de uno o dos paquetes, todos los paquetes comienzan con un identificador del paquete como se muestra en la tabla 1.5.

[1] Tabla tomada del libro "*USB COMPLETE Second Edition*" de JAN AXELSON

Tipo de transferencia	Etapa (cero o más transacciones)	Fases
Control	*Setup*	*Token*
		Datos
		Handshake
	Datos (*IN u OUT*) (opcional)	*Token*
		Datos
		Handshake
	Estado (*IN u OUT*)	*Token*
		Datos
		Handshake
Bulk	Datos (*IN u OUT*)	*Token*
		Datos
		Handshake
Interrupción	Datos (*IN u OUT*)	*Token*
		Datos
		Handshake
Isócrona	Datos (*IN u OUT*)	*Token*
		Datos

Tabla 1.4 Los tipos de transferencias y sus fases[1].

- **Fase de *Token*:** el *host* envía peticiones de comunicación en un paquete *Token*. El PID indica el tipo de transacción. El paquete *Token* puede llevar un marcador de inicio de trama (SOF) que es una referencia de sincronización que el *host* envía en intervalos de 1 milisegundo a *full speed* y de 125 microsegundos a *high speed*.

- **Fase de Datos:** el *host* o el dispositivo puede transferir cualquier clase de información en un paquete de datos, el PID es el valor que indica la posición de los datos cuando hay múltiples paquetes de datos.

- **Fase de *Handshake*:** el *host* o el dispositivo envía el estado de la comunicación en un paquete de *handshake*, el PID mantiene el código de estado (*ACK, NAK, STALL, NYET*).

Los dispositivos *low speed* no consideran el paquete de SOF, pues utilizan una señal más simple de fin de Paquete (EOP) llamada señal *low speed* que se mantiene activa, enviada una vez por cada trama, como el SOF actúa para los

[1] Tabla tomada del libro "*USB COMPLETE Second Edition*" de JAN AXELSON

dispositivos *full speed*, los dispositivos *low speed* se mantienen activos, fuera del estado de suspensión.

Tipo de Paquete	Nombre del PID	Valor	Tipos de transferencia utilizadas en	Fuente	Velocidad del bus
Token	*OUT*	0001	Todas	*Host*	Todas
	IN	1001	Todas	*Host*	Todas
	SOF	0101	Inicio de trama	*Host*	Todas
	SETUP	1101	Control	*Host*	Todas
Datos	*DATA 0*	0011	Todas	*Host*, dispositivo	Todas
	DATA 1	1011	Todas	*Host*, dispositivo	Todas
	DATA 2	0111	Isócrona	*Host*, dispositivo	*High*
	MDATA	1111	Isócrona, Interrupción	*Host*, dispositivo	*High*
Handshake	*ACK*	0010	Todas	*Host*, dispositivo	Todas
	NAK	1010	Control, Bulk, Interrupción.	Dispositivo	Todas
	STALL	1110	Control, Bulk, Interrupción	Dispositivo	Todas
	NYET	0110	Control, escritura, Bulk OUT, transacciones split.	Dispositivo	*High*
Especial	*PRE*	1100	Control, Interrupción	*Host*	*Full*
	ERR	1100	Todas	Dispositivo concentrador	*High*
	SPLIT	1000	Todas	*Host*	*High*
	PING	0100	Todas	*Host*	*High*
	Reservado	0000	_	_	_

Tabla 1.5 Identificadores de paquete (PID)[1].

El PID utilizado por los dispositivos *low speed* es *PRE*, el cual contiene un código de preámbulo que indica a los concentradores que el siguiente paquete será *low speed*. El PID utilizado por los dispositivos *high speed* es el *PING*, el *host* envía un *PING* para descubrir si un *endpoint* de un dispositivo *high speed* está ocupado antes de enviar el siguiente paquete de datos.

El PID *SPLIT* identifica un paquete *Token* como parte de una transacción dividida. El PID *ERR* se utiliza solamente en transacciones dividas, cuando un

[1] Tabla tomada del libro "*USB COMPLETE Second Edition*" de JAN AXELSON

concentrador 2.0 utiliza este PID para reportar un error al *host* en una transacción *low* o *full speed*. Los PID *ERR* y *PRE* tienen el mismo valor, pero no habrá confusión porque un concentrador nunca envía un *PRE* al *host* o un *ERR* a un dispositivo.

1.3.7 SINCRONIZACIÓN

Los retardos entre el paquete *Token*, los paquetes de datos, y los paquetes de *handshake* de una transacción deben ser pequeños, permitiendo retardos en el cable, tiempos de conmutación y un retardo que el *hardware* genera en la preparación de una respuesta como contestación al paquete recibido.

Aunque los dispositivos deben terminar cada transacción rápidamente, el bus puede acomodar las transferencias con los dispositivos que necesitan tiempo adicional para responder.

1.3.8 TRANSACCIONES DIVIDIDAS

Un concentrador 2.0 se comunica con un *host* 2.0 a *high speed*, a menos que un concentrador 1.x esté entre ellos. Un *host* 2.0 utiliza transacciones divididas cuando se comunica con un dispositivo *low speed* o *full speed* en un bus *high speed.* La figura 1.8 muestra las transacciones que ocurren entre dispositivos de diferentes velocidades.

Cuando un dispositivo *low* o *full speed* se conecta a un concentrador 2.0, el concentrador se encarga de convertir las velocidades según lo que necesite, lo que permite que el bus no espere mientras un concentrador intercambia datos con un dispositivo *low* o *full speed*.

Las transacciones divididas permiten el mejor uso del tiempo del bus, pues reducen al mínimo el tiempo de espera del bus para un dispositivo *low o full speed.*

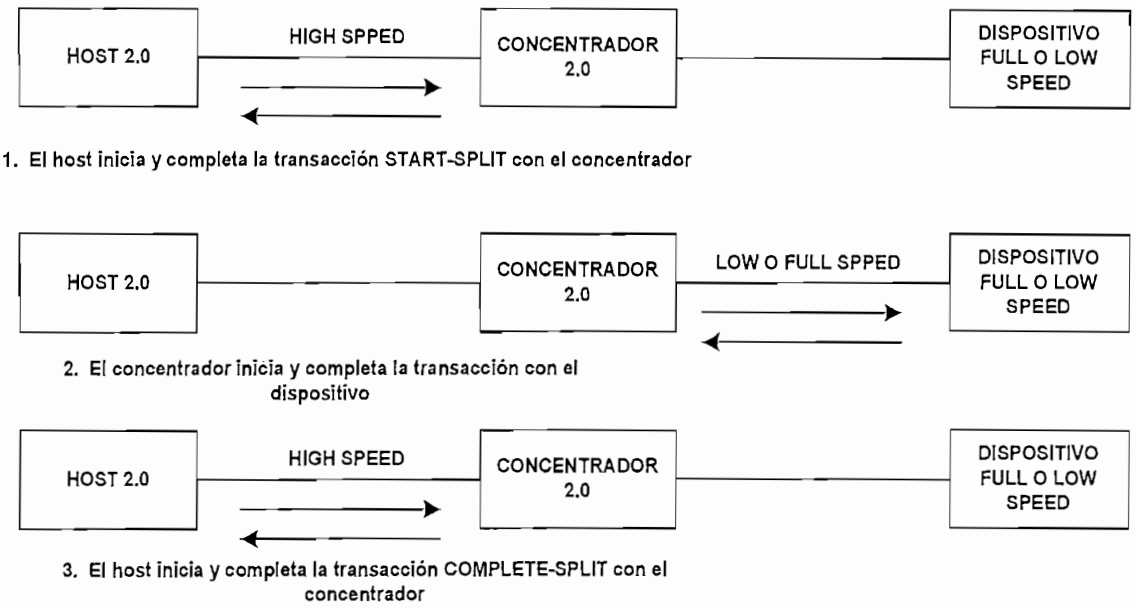


Figura 1.8 transacciones divididas entre dispositivos de diferentes velocidades[1].

En el inicio de una transacción dividida, el *host* 2.0 envía un paquete *Token start-split* (*SSPLIT*), seguido por el usual paquete *low speed o full speed*, y cualquier paquete de datos destinado para el dispositivo. El concentrador 2.0 retorna un *ACK o NAK*. Entonces el *host* está libre para utilizar el bus para otras transacciones.

Para terminar la transacción, el concentrador convierte el paquete o los paquetes recibidos del *host* a la velocidad apropiada, los envía al dispositivo, y almacena la respuesta del dispositivo, si la hay dependiendo de la transacción, el dispositivo puede retornar datos, señal de *handshake*, o nada. El *host* todavía no ha recibido respuesta del dispositivo.

Mientras que el concentrador está terminando la transacción con el dispositivo, el *host* puede iniciar otro tráfico a través del bus, que el concentrador debe manejar. En una transacción *complete-split*, el host envía un paquete *Token complete-split* (CSPLIT), seguido por el usual paquete *Token low* o *full speed* para solicitar el estado del dispositivo recibido por el concentrador. El concentrador envía al host, los datos solicitados o un código de estado. Esto termina la transacción. El

[1] Figura tomada del libro "*USB COMPLETE Second Edition*" de JAN AXELSON

dispositivo no tiene ningún conocimiento de la transacción complete-split pues terminó la transacción con el concentrador.

1.3.9 CONTROL DE FLUJO Y CHEQUEO DE ERRORES

Para asegurarse de que cada transferencia tenga éxito, el USB utiliza *handshaking* y chequeo de errores.

1.3.9.1 *Handshaking*

El USB utiliza el *handshaking* como información para manejar el flujo de datos, un código indica el éxito o la falla de todas las transacciones excepto en las transferencias Isócronas.

Los códigos de estado definidos son *ACK, NAK, STALL, NYET, y ERR.* Un sexto indicador de estado es la ausencia del código esperado de *handshake*, indicando un error drástico en el bus. La tabla 1.6 muestra los indicadores de estado y a donde se transmiten en cada tipo de transacción.

- *ACK*: Indica que un *host* o un dispositivo ha recibido datos sin error.

- *NAK*: Indica que el dispositivo está ocupado o que no tiene ningún dato por retornar.

- *STALL*: El *handshake STALL* puede significar: petición de control sin soporte, petición del control fallida, o *endpoint* fallido. La especificación USB define los siguientes tipos de *STALL*.

  Protocolo STALL: Cuando un dispositivo recibe una petición de transferencia de control que el *endpoint* no soporta, el dispositivo devuelve un *STALL* al *host* o envía un *STALL* si soporta la petición pero por alguna razón no puede tomar la acción solicitada.

***STALL* funcional:** consiste en responder a las peticiones de transferencia cuando el *endpoint* detiene la característica establecida, indicando que el *endpoint* no puede enviar o recibir todos los datos.

Tipo de transacción o *PING*	Orígen de los paquetes de datos	Contenido de los paquetes de datos	Orígen de los paquetes de *handshake*	Contenido de los paquetes de *handshake*
Setup	*Host*	Datos	dispositivo	*ACK*
OUT	*Host*	Datos	dispositivo	*ACK, NAK, STALL, NYET* (para *high speed*), *ERR* (para el concentrador en *complete-split*)
IN	Dispositivo	Datos, *NAK, STALL, ERR* (para el concentrador en *complete-split*)	*host*	*ACK*
PING (solo *high speed*)	Nada	Nada	dispositivo	*ACK, NAK, STALL.*

Tabla 1.6 Indicadores de estado de cada tipo de transacción[1].

- ***NYET*:** Solo los dispositivos *high speed* utilizan *NYET*. Cuando un dispositivo recibe un paquete de datos, el dispositivo puede devolver al *host* un paquete *NYET*, que indica que los datos fueron aceptados pero el *endpoint* no está listo para recibir otro paquete de datos.

- ***ERR*:** Es utilizado solo por los concentradores *high speed* e indica que el dispositivo no devolvió un *handshake* previsto en la transacción que el concentrador está terminando con el *host*.

- **Sin respuesta:** Ocurre cuando el *host* o un dispositivo espera recibir un *handshake* previsto, pero no recibe nada. Esto indica generalmente que el cálculo del chequeo de errores en el receptor detectó un error e informa al transmisor que debe intentar otra vez.

[1] Tabla tomada del libro "*USB COMPLETE Second Edition*" de JAN AXELSON

1.3.9.2 Chequeo de errores

Debido a que la interfaz utiliza cableado externo, hay ocasiones en que el ruido inesperado en el cable puede corromper la transmisión. Por esta razón, los paquetes incluyen *bits* de chequeo de errores que permiten a un receptor identificar si los datos recibidos son iguales a los que fueron enviados.

- ***Bits* de chequeo de errores**

  Todo paquete *Token*, Datos, y de inicio de trama incluyen los *bits* de chequeo de errores, que son calculados usando el algoritmo matemático llamado Control de Redundancia Cíclica (CRC).

  El transmisor realiza el cálculo y envía el resultado junto con los datos, el dispositivo receptor también realiza el cálculo sobre los datos recibidos. Si los resultados son iguales, los datos no tienen error y el dispositivo de recepción envía el ACK, si los resultados son diferentes, el dispositivo de recepción no envía ningún *handshake*, lo que permite indicar al transmisor que retransmita la información. Típicamente, el *host* intenta un total de tres veces, aunque la especificación da al *host* flexibilidad en la determinación del número de retransmisiones.

- ***Bits* data toggle**

  En las transferencias que requieren múltiples transacciones, el *bit data toggle* asegura que no falten transacciones, manteniendo a los dispositivos transmisor y receptor sincronizados. Cuando el receptor detecta una transacción entrante de datos, compara el *bit data toggle* recibido con su propio *data toggle*. Si los bits son iguales, el receptor acciona su *bit* y envía un *ACK* al transmisor.

Si un receptor retorna un *ACK*, pero por alguna razón el transmisor no la ve, el transmisor no modifica su *bit data toggle* e intenta otra vez con los mismos datos, en este caso, el receptor descarta los datos repetidos y no acciona su *bit data toggle*, pero envía un *ACK*.

1.4 ENUMERACIÓN

La enumeración es el intercambio inicial de información que permite al *host* recibir información del dispositivo y así asignarle un controlador.

1.4.1 PROCESO

Un concentrador detecta la conexión y desconexión de los dispositivos, cada concentrador tiene una interrupción *IN* para informar estos eventos al *host*. En el arranque del sistema el *host* sondea al concentrador raíz para recibir información de los dispositivos que están conectados, después del arranque, el *host* continúa sondeando periódicamente para recibir información de cualquier dispositivo conectado o desconectado recientemente, enviando una serie de peticiones al concentrador, permitiendo al concentrador establecer una ruta de comunicaciones entre el *host* y el dispositivo.

Para el usuario, la enumeración es transparente y automática, a excepción de la aparición de una ventana que anuncia la detección de un nuevo dispositivo y si el intento de configurar tuvo éxito. Algunas veces cuando el dispositivo se conecta por primera vez, el usuario necesita proporcionar un disco que contiene los archivos INF y los *drivers* del dispositivo.

Cuando se completa la enumeración, Windows agrega el nuevo dispositivo a la pantalla de administración de dispositivos en el panel de control como se muestra en la Figura 1.9.

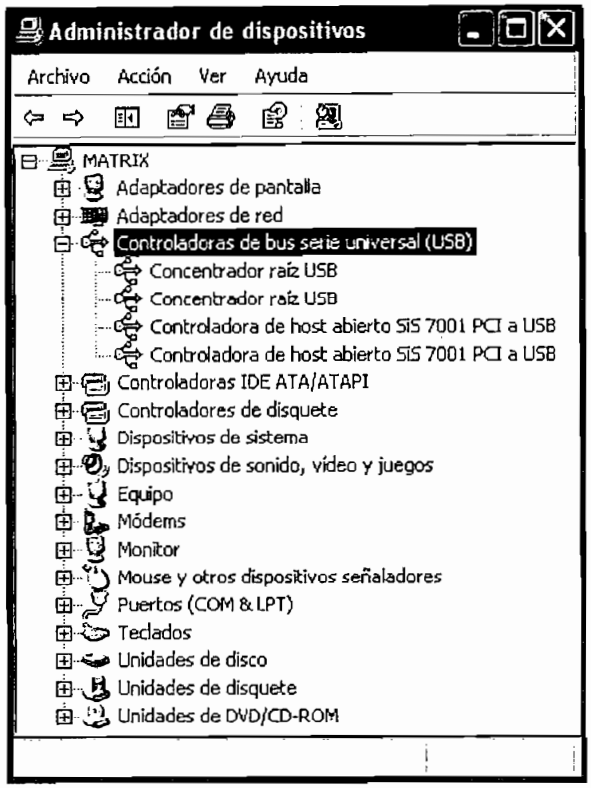


Figura 1.9 El Administrador de Dispositivos en Windows XP[1].

1.4.2 PASOS DE LA ENUMERACIÓN

Durante el proceso de la enumeración, el dispositivo y los concentradores se mueven a través de cuatro de los seis estados del dispositivo, definidos por la especificación: Encendido, Defecto, Dirección y Configuración. (Los otros estados son Conexión y Suspensión). Los pasos siguientes que ocurren en la enumeración, son una secuencia típica que ocurre en *Windows*:

- La conexión de un dispositivo en el puerto USB puede realizarse antes o después de arrancar el sistema.

- La detección del dispositivo por parte del concentrador se la hace revisando el nivel en alto en las líneas de señal del dispositivo, el nivel en alto en D+ es para dispositivos *full speed* o en D- para dispositivos *low speed.*

[1] Figura tomada del sistema operativo *Windows XP*

- El *host* recibe información del nuevo dispositivo a través del concentrador, para recibir dicha información, el *host* envía una petición para averiguar el estado de los puertos del concentrador, una vez recibida esta petición, el concentrador envía la información del nuevo dispositivo conectado recientemente.

- La detección del dispositivo *low* o *full speed* por parte del concentrador se la realiza detectando la velocidad del dispositivo, para lo cual el concentrador determina cual línea tiene el voltaje más alto cuando está inactivo.

- El concentrador reinicia el dispositivo, siempre que el *host* recibe información del nuevo dispositivo, el controlador del *host* envía al concentrador una petición, en la cual pide que se reinicie el puerto. El concentrador pone las líneas de datos del USB del dispositivo en la condición de Reinicio por lo menos durante 10 milisegundos. La condición de Reinicio es una condición especial en la cual tanto D+ como D-, están en bajo. (Normalmente las líneas tienen estados lógicos contrarios). El concentrador envía al reinicio solo al nuevo dispositivo; lo cual es transparente para los concentradores y dispositivos que comparten el bus.

- Para detectar si un dispositivo soporta *high speed*, se basa en el estado de dos señales especiales, en el estado *chirp J*, en donde solo la línea D+ es controlada o en el estado *chirp K*, en donde solo la línea D- es controlada. Los chirp J y K están definidos como voltajes DC diferenciales , es decir en un chirp J la línea D+ es mas positiva que la línea D- y en un chirp K la línea D- es mas positiva que la línea D+.

  Durante el reinicio, un dispositivo que soporta *high speed* envía un *chirp K*, Un concentrador *high speed* detecta el *chirp* y responde con una serie de *chirp* Ks y Js alternados. Cuando el dispositivo detecta el patrón KJKJKJ, este quita su *pull-up full speed* y realiza las comunicaciones a *high speed*. Si el concentrador no responde al *chirp* K del dispositivo, el dispositivo

continúa comunicándose a *full speed*. Todos los dispositivos *high speed* deben ser capaces de responder a las peticiones de enumeración a *full speed*.

- Para establecer una vía de comunicación entre el dispositivo y el bus, el host hace peticiones al concentrador para verificar si el dispositivo ha salido del estado de reinicio, esto lo hace hasta que el dispositivo salga del estado de reinicio.

  Cuando el concentrador quita el estado de reinicio, el dispositivo está en el estado por Defecto. Los registros USB del dispositivo están en su estado de reinicio y el dispositivo está listo para responder a las transferencias de control sobre la vía de comunicación por defecto en el *endpoint* 0. El dispositivo puede ahora comunicarse con el *host* utilizando la dirección por defecto 00H.

- El *host* envía una petición al dispositivo para obtener información del tamaño máximo del paquete que se puede transmitir, puesto que el *host* enumera un dispositivo a la vez, solo un dispositivo responderá a las comunicaciones, incluso si varios dispositivos se conectan al mismo tiempo. Una vez que se ha completado este estado, el *host Windows*, solicita al concentrador que reinicie el dispositivo, según la especificación no requiere un reinicio, porque los dispositivos deben de ser capaces de manejar una transferencia de control y responder al siguiente paquete de configuración.

- El *host* asigna una dirección única al dispositivo enviando una petición, el dispositivo lee la petición, retorna un acuse de recibo y almacena la dirección. En ese instante el dispositivo se encuentra en el estado de Dirección.

  Todas las comunicaciones utilizan esta dirección, la cual es válida hasta que el dispositivo sea desconectado, se lo reinicie o el sistema se apague.

En la siguiente enumeración, el mismo dispositivo puede obtener una dirección diferente.

- El *host* recibe información de las capacidades del dispositivo mediante una petición que le permite leer el descriptor del dispositivo.

- Después de que el *host* recibe información de sus descriptores, busca el mejor controlador para manejar las comunicaciones con el dispositivo, de no encontrarlo, *Windows* busca uno con cualquier clase, subclase y valor de protocolo recuperado del dispositivo.

  Una excepción a esta secuencia la tienen los dispositivos compuestos, los cuales tienen múltiples interfaces, y requieren un controlador para cada interfaz. El *host* puede asignar estos controladores, solo después de habilitar las interfaces, las cuales requieren que el dispositivo esté configurado.

- Después de recibir la información del dispositivo a partir del descriptor, el controlador del dispositivo requiere ser configurado, por lo que realiza una petición para fijar una configuración. Muchos dispositivos soportan solo una configuración, pero si un dispositivo soporta múltiples configuraciones, el controlador puede decidir cuál usar.

  El dispositivo lee la petición y establece la mejor configuración, el dispositivo está ahora en el estado de configuración y las interfaces están habilitadas, el *host* usa la información recuperada del dispositivo para encontrar el controlador apropiado. El dispositivo esta listo para ser utilizado. Los otros dos estados del dispositivo Conexión y Suspensión, pueden existir en cualquier momento.

- Si el concentrador no está proporcionando energía (VBUS) al puerto, el dispositivo está en el estado de Conexión. Esto puede ocurrir si el concentrador ha detectado una condición de sobre intensidad de corriente,

o si el *host* requiere que el concentrador quite la energía del puerto. Sin energía en VBUS, el *host* y el dispositivo no pueden comunicarse.

- El estado de suspensión significa que el dispositivo no se encuentra activo. En el estado de suspensión, debe consumir la mínima energía del bus.

1.4.3 DESCONEXIÓN DEL DISPOSITIVO

Cuando se retira un dispositivo del bus, el concentrador deshabilita el puerto del dispositivo, el *host* recibe información de la desconexión después de sondear al concentrador. *Windows* entonces quita el dispositivo de la Pantalla del Administrador de Dispositivos y la dirección del dispositivo llega a estar disponible para otro dispositivo.

1.4.4 TIPOS DE DESCRIPTORES

Los descriptores son estructuras de datos que habilitan al *host* para recibir información de un dispositivo. Cada descriptor contiene información sobre el dispositivo en su totalidad o de un elemento en el dispositivo. En la Tabla 1.7 se enumeran los tipos de descriptores.

La especificación 2.0 agrega un descriptor de *Interface_Power*[1] y los dispositivos que soportan *full y high speed*, soportan dos tipos adicionales de descriptores: *Device_Qualifier y Other_Speed_Configuration.*

1.4.4.1 Descriptor del Dispositivo

Cada dispositivo tiene solo un descriptor del dispositivo que contiene información sobre el dispositivo, el número de configuraciones que el dispositivo soporta y la clase.

[1] El documento que describe la estructura y el uso de este descriptor es: Especificación de la característica del USB: Administración de la energía del interfaz.

Tipos de Descriptores	¿Requerido?
Dispositivo	Si
Device_Qualifier	Si, para dispositivos que soportan *full y high speed.*
Configuración	Si
Other_Speed_Configuration	Si, para dispositivos que soportan *full y high speed.*
Interfaz	Si
Endpoint	No, si el dispositivo utiliza solo *endpoint* 0.
Secuencia	No. Texto descriptivo opcional.
Interface_Power	No. Soporta administración del nivel de energía de la interfaz.

Tabla 1.7 Tipos de descriptores estándares[1].

1.4.4.2 Descriptor *Device_Qualifier*

Los dispositivos *full* y *high speed* tienen un descriptor *Device_Qualifier.* Si el dispositivo conmuta velocidades, algunos campos en el descriptor del dispositivo pueden cambiar.

1.4.4.3 Descriptor de Configuración

Cada dispositivo tiene uno o más descriptores de configuración que contienen información sobre el uso de la energía del dispositivo, el número de interfaces soportadas por la configuración, las características y capacidades del dispositivo.

Una sola configuración es suficiente, pero un dispositivo puede soportar múltiples configuraciones. Solo una configuración está activa a la vez. Cada configuración requiere un descriptor.

1.4.4.4 Descriptor *Other_Speed_Configuration*

Este descriptor lo soportan los dispositivos *full* y *high speed* y contiene información de configuración cuando el dispositivo está funcionando con la otra velocidad posible.

[1] Tablas tomadas del libro "*USB COMPLETE Second Edition*" de JAN AXELSON

1.4.4.5 Descriptor de la Interfaz

El descriptor de la interfaz contiene información sobre los *endpoints* que la interfaz soporta, cada configuración debe soportar una interfaz, y para muchos dispositivos, un descriptor es suficiente. Pero se pueden tener múltiples interfaces las cuales estén activas al mismo tiempo. Un dispositivo compuesto es capaz de tener una configuración con múltiples interfaces activas al mismo tiempo. El *host* carga un controlador para cada interfaz. Cambiar interfaces es más simple que cambiar las configuraciones de un dispositivo.

1.4.4.6 Descriptor del *Endpoint*

Cada descriptor de la interfaz tiene cero o más descriptores del *endpoint* que contienen la información necesaria para comunicarse con un *endpoint*. El *endpoint* 0 nunca tiene un descriptor porque cada dispositivo debe soportar el *endpoint* 0, el descriptor del dispositivo contiene el tamaño máximo de paquete, y la especificación define todo sobre el *endpoint*.

1.4.4.7 Descriptor de Secuencia

Un descriptor de secuencia contiene el texto descriptivo, la especificación define los descriptores de secuencia para el fabricante, el producto, el número de serie, la configuración, y la interfaz, los descriptores de secuencia son opcionales.

1.4.5 LAS PETICIONES

Todos los dispositivos deben responder a estas peticiones[1]. Los valores se extienden a partir de 00h a 0Ch, con algunos valores no utilizados. La mayoría de las peticiones están en pares, por cada petición *SET,* se tiene su correspondiente petición *GET* o *CLEAR.* Las excepciones son: *Set_Address*, *Synch_Frame* y *Get_Status.*

[1] En el anexo A se presenta una tabla que resume las peticiones estándares del USB 1.1, seguidas por una descripción de cada petición.

- ***Set_Address***: En esta petición el *host* especifica la nueva dirección para comunicarse en un futuro con el dispositivo. Los valores permitidos son de 1 a 127, cada dispositivo en el bus, incluyendo el concentrador raíz, tiene una dirección única.

- ***Get_Descriptor***: El *host* solicita un descriptor específico. Éste contiene dos *bytes*: el *byte* superior que contiene el tipo del descriptor y el *byte* inferior que contiene el índice del descriptor.

- ***Set_Descriptor***: El *host* agrega un descriptor o actualiza el descriptor existente. Esta petición contiene dos *bytes*: el *byte* superior indica el tipo de descriptor y el *byte* inferior indica el índice del descriptor.

- ***Set_Configuration***: Enseña al dispositivo para utilizar la configuración seleccionada. El *byte* inferior específica la configuración, si los valores coinciden con una configuración soportada por el dispositivo, el dispositivo selecciona la configuración solicitada. Un valor de 0 indica no configurado. Si el valor es 0, el dispositivo entra al estado de dirección y requiere una nueva Petición *Set_Configuration*.

- ***Get_Configuration***: El *host* solicita el valor de la configuración actual del dispositivo.

- ***Set_Interface***: Para los dispositivos que soportan múltiples configuraciones, se realiza los ajustes exclusivos para una interfaz, el *host* solicita al dispositivo utilizar un ajuste específico.

- ***Get_Interface***: Para los dispositivos que soportan múltiples configuraciones, se realiza los ajustes exclusivos para una interfaz, el *host* solicita el ajuste actual.

- ***Set_Feature***: Peticiones del *host* para habilitar una función en un dispositivo, una interfaz, o un *endpoint*.

- ***Clear_Feature***: Peticiones del *host* para inhabilitar una función en un dispositivo, una interfaz, o un *endpoint*.

- ***Get_Status***: El *host* solicita el estado de las funciones de un dispositivo, interfaz, o *endpoint*.

- ***Synch_Frame***: Le permite al dispositivo generar una trama de sincronización de un *endpoint* con la cual se indique al PC el número de tramas que preceden al comienzo de una nueva secuencia.

1.5 ADMINISTRACIÓN DE LA ENERGÍA

Una característica muy conveniente del USB es la capacidad para que los periféricos tomen energía del bus, la misma que tiene un límite disponible.

1.5.1 OPCIONES DE ENERGÍA

La capacidad de tomar energía del mismo cable que lleva datos hacia y desde el PC tiene varias ventajas. Del punto de vista del usuario, elimina la necesidad de un enchufe eléctrico cerca del periférico y hace al periférico más pequeño y ligero, del punto de vista del fabricante, hace al periférico más barato de fabricar.

1.5.2 VOLTAJES

El voltaje nominal entre las líneas de VBUS y de tierra en un cable del USB es 5V, pero el valor real puede ser poco mayor o un poco menor. Un dispositivo que usa energía del bus debe manejar las variaciones y conformarse con la especificación. La Tabla 1.8 muestra los voltajes del mínimo y del máximo permitidos en los puertos de un concentrador:

Tipo de concentrador	Voltaje mínimo	Voltaje máximo
Alta Energía	4.75	5.25
Baja Energía	4.4	5.25

Tabla 1.8 Voltajes máximo y mínimo en los puertos de un concentrador[1].

1.5.3 PERIFÉRICOS QUE PUEDEN UTILIZAR ENERGÍA DEL BUS

Un periférico que requiere hasta 500 miliamperios puede utilizar energía del bus con algunas limitaciones: computadores con batería no soportan periféricos que necesiten más de 100 miliamperios del bus.

1.5.4 NECESIDADES DE ENERGÍA

Un dispositivo de baja energía es un dispositivo que necesita 100 miliamperios del bus, y un dispositivo de alta energía es un dispositivo que necesita 500 miliamperios del bus. Un dispositivo independiente tiene su propia fuente de alimentación y utiliza la energía que su fuente le proporcione.

En el arranque, cualquier dispositivo puede necesitar hasta 100 miliamperios del bus hasta que se configure el dispositivo. Un dispositivo de alta energía no puede tomar más de 100 miliamperios hasta que el *host* le permita utilizar la energía que requiere. Un dispositivo independiente puede también tomar 100 miliamperios del bus en cualquier momento.

1.6 CODIFICACIÓN DE LOS DATOS

Todos los datos sobre el bus se codifican utilizando el formato de codificación *Non-Return to Zero inverted* (NRZI) con *bit* de relleno, esta codificación permite la sincronización de los dispositivos con el transmisor, sin necesidad de enviar por separado una señal del reloj o bits de inicio y parada con cada *byte*.

[1] Tabla tomada del libro "*USB COMPLETE Second Edition*" de JAN AXELSON

En lugar de definir la lógica 0s y 1s como voltajes, el USB define la lógica 0 como un cambio del nivel de voltaje, y la lógica 1 como el estado anterior de voltaje. Afortunadamente, el hardware del USB hace toda la codificación y decodificación automáticamente.

1.6.1 SINCRONIZACIÓN PERMANENTE

El USB utiliza dos técnicas para mantenerse sincronizado: bits de relleno y los campos *SYNC*. Cada uno agrega cierto *overhead* a cada transacción, pero la cantidad es insignificante con respecto a los paquetes grandes.

1.6.1.1 *Bit* de relleno

Se requiere el *bit* de relleno porque el receptor sincroniza en transiciones. Si los datos son todos 0s, hay muchas transiciones. Pero si los datos contienen una cadena larga de 1s, la carencia de transiciones podría hacer que el receptor pierda sincronización. Si los datos tienen seis 1s consecutivos, el transmisor rellena o inserta un 0 después del sexto 1. Esto asegura por lo menos una transición para anchos de siete bits. El receptor detecta y desecha cualquier *bit* que siga a continuación de los seis 1s consecutivos. El *bit* de relleno puede aumentar el número de bits transmitidos hasta un 17 por ciento. El promedio en la práctica es mucho menos. El *overhead* de los bits de relleno para los datos al azar son apenas 0.8 por ciento, o un *bit* de relleno por 125 bits de datos.

1.6.1.2 Campo *SYNC*

Para mantener la sincronización, cada paquete comienza con un campo *SYNC* permitiendo al dispositivo de recepción sincronizar su reloj a los datos transmitidos. Para velocidades *low* y *full speed*, el patrón SYNC es 8 bits: KJKJKJKK. La transición del estado libre a la primera K sirve como una clase de *bit* de inicio que indica la llegada de un nuevo paquete.

Para *high speed* el patrón *SYNC* es 32 bits: quince repeticiones del KJ, seguidas por KK. Para el final del patrón *SYNC*, el dispositivo de recepción puede

determinar exactamente cuándo llegará cada uno de los bits restantes en el paquete.

El aumento de 8 o 32 periodos de *bits*, aumenta el *overhead* por cada paquete enviado, por lo que es más eficiente con paquetes de gran tamaño.

1.6.1.3 Señal de fin de paquete

La señal de fin de paquete a *low* o *full speed* es un solo cero de finalización que dura dos periodos de *bit*. Los receptores *high speed* tratan cualquier error del *bit* de relleno como el fin del paquete, así que el fin del paquete *high speed* debe causar un error del *bit* de relleno. Para todos los paquetes *high speed* excepto los paquetes de inicio de trama, el fin de paquete es un *byte* codificado 01111111, sin el *bit* de relleno. En paquetes de inicio de trama *high speed*, el fin del paquete son 40 bits. El *byte* codificado comienza con un cero, seguido por 39 unos, que da lugar a un final del paquete que consiste en 40 Js o 40 Ks.

1.7 FORMATO DEL PAQUETE

Los paquetes alternadamente contienen los siguientes tipos de campos: *SYNC, PID*, Dirección, *Endpoint*, Número de trama, Datos, y CRC. En la Tabla 1.9 se enumeran los campos y sus propósitos.

1.7.1 CAMPO *SYNC*

Cada paquete comienza con un campo *SYNC* de 8 bits, este campo sirve como el delimitador de inicio del paquete y se puede transmitir solamente en un bus libre.

1.7.2 CAMPO DE IDENTIFICADOR DEL PAQUETE

El campo del identificador del paquete (PID) es de 8 bits. Los bits 0 a 3 identifican el tipo de paquete y los bits 4 a 7 son para chequeo de errores.

Nombre	Tamaño (bits)	Tipo de paquete	Propósito
SYNC	8	Todos	Inicio del paquete y sincronización
PID	8	Todos	Identifica el tipo de paquete
Dirección	7	*IN, OUT, Setup*	Identifica la dirección de función
Endpoint	4	*IN, OUT, Setup*	Identifica el *endpoint*
Número de Trama	11	SOF	Identifica la trama
Datos	0 a 8192 (1024 bytes para hardware 2.0), 0 a 8184 (1023 bytes para hardware 1.x)	Data 0, Data 1	Datos
CRC	5 o 16	*IN, OUT, Setup, Data 0, Data 1*	Detección de errores

Tabla 1.9 Tipos de campo y su propósito[1].

1.7.3 CAMPO DE DIRECCIÓN

El campo de dirección es de 7 bits que identifican la función con la cual el *host* se está comunicando. La función es un dispositivo (que puede ser un concentrador), o una función específica en un dispositivo compuesto.

1.7.4 CAMPO DEL ENDPOINT

El campo del *endpoint* es de 4 bits que identifica un número de *endpoint* dentro de una función. Una función *low speed* puede tener no más de 3 números de *endpoint*. Una función *full o high speed* puede tener hasta 16 números.

1.7.5 CAMPO DEL NÚMERO DE TRAMA

Este campo es de 11 bits que identifica una trama específica. El *host* envía este campo en el paquete de inicio de trama con el que empieza cada trama o micro trama. Los valores varían entre 0 a 7FFh.

[1] Tabla tomada del libro "*USB COMPLETE Second Edition*" de JAN AXELSON

1.7.6 CAMPO DE DATOS

Este campo puede extenderse a partir de 0 a 1024 *bytes*, dependiendo del tipo de transferencia, la cantidad de datos en la transacción, y la versión del USB. Dispositivos 1.x soportan el campo de datos de hasta 1023 *bytes*.

1.7.7 CAMPO CRC

El campo del CRC es de 5 bits para los campos de dirección y de *endpoint* y 16 *bits* para el campo de datos, los *bits* son usados para chequeo de errores. El *hardware* que transmite inserta los bits CRC y el *hardware* de recepción hace los cálculos requeridos.

1.7.8 RETARDO ENTRE PAQUETES

El USB lleva datos desde múltiples fuentes, en ambas direcciones, en un par de líneas, los datos pueden viajar en una dirección a la vez. Para asegurarse de que el dispositivo que anteriormente transmitía haya tenido tiempo para apagar su controlador, el bus requiere un breve retardo entre el final de un paquete y el inicio del paquete siguiente en una transacción. Este tiempo de retardo es limitado, sin embargo, los dispositivos deben poder cambiar direcciones rápidamente, la especificación define un retardo diferente *para low/full* y *high speed.*

1.8 TRANSMISORES-RECEPTORES Y SEÑALES

Las características eléctricas de las señales llevadas por un cable USB varían dependiendo de la velocidad del segmento del cable.

1.8.1 SEGMENTOS DE CABLE

Un segmento de cable es un solo cable físico que conecta un dispositivo con un concentrador (que puede ser el concentrador raíz en el *host*) como se muestra en

la Figura 1.10. Los segmentos *low speed* existen solo entre los dispositivos *low speed* y sus concentradores ya que solo transporta datos *low speed*.

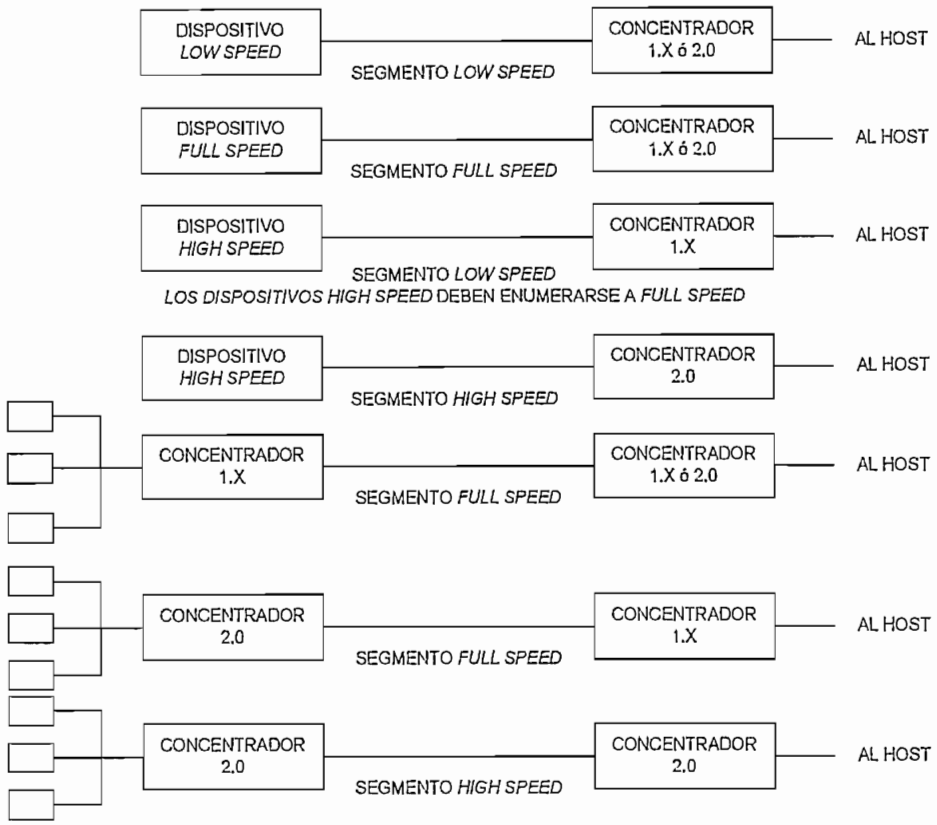


Figura 1.10 Conexión entre los distintos dispositivos y su velocidad[1]

Un segmento *full speed* existe cuando el dispositivo es *full speed*, este dispositivo puede ser un 1.x o un concentrador 2.0. Cuando el dispositivo es un concentrador, el segmento puede llevar datos hacia y desde los dispositivos *low speed* que se encuentren conectados a ese concentrador. Los segmentos *full speed* nunca llevan datos *high speed*.

Los segmentos *high speed* existen solo si se tiene un concentrador 2.0 y los dispositivos conectados a éste son de *high speed*. Cuando el dispositivo conectado al concentrador 2.0 es otro concentrador, el segmento puede también llevar datos a y desde los dispositivos *low* y *full speed* que están conectados a éste concentrador. Los datos en segmentos *high speed* viajan a *high speed*.

[1] Tabla tomada del libro "*USB COMPLETE Second Edition*" de JAN AXELSON

1.8.2 CABLES

La especificación del USB incluye los siguientes requisitos para los cables.

1.8.2.1 Conductores

Los cables USB tienen cuatro conductores: VBUS, tierra, D+ y D-, como se muestra en la Figura 1.11. VBUS es la fuente de +5V. La tierra es la referencia de tierra para VBUS, D+ y D-. D+ y D- son el par diferenciado de la señal.

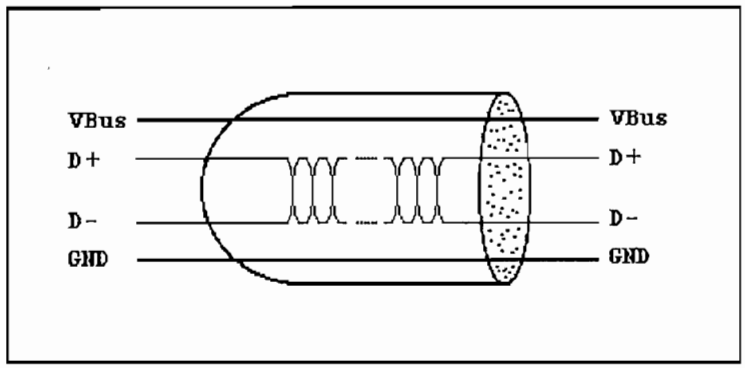


Figura 1.11 Conductores de un cable USB[1].

El icono del USB realzado en el conector del enchufe identifica un cable del USB (Figura 1.12). Los PC y los concentradores pueden utilizar el icono para identificar sus conectores USB. Un "+" agregado al icono indica el rendimiento del USB 2.0.

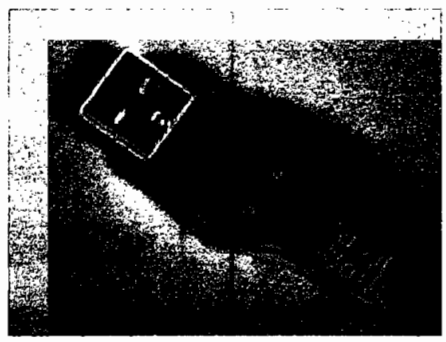


Figura 1.12 El icono identifica cable USB[2].

[1] Figura tomada de la web http://sipan.inictel.gob.pe

[2] Figura tomada del libro "*USB COMPLETE Second Edition*" de JAN AXELSON

Los cables al ser usados en segmentos *full* y *high speed* tienen requisitos diferentes a los de segmentos *low speed.* La Tabla 1.10 compara los dos tipos del cable. La especificación 2.0 intensificó los requisitos para los cables *low speed.* Un cable *low speed* 1.1 no requería ningún blindaje, mientras que un cable *low speed* 2.0 debe tener el mismo protector interno y el alambre requerido para *full speed.* La especificación tambien recomienda, un protector externo y par trenzado para los datos, para los dos tipos de segmentos.

Especificación	Low Speed	Full/High Speed
Longitud máxima (metros)	3	5
¿Protector interno y línea de drenaje requerido?	Si (Nuevo en USB 2.0)	Si
¿Protector trenzado externo?	No, pero recomendado	Si
¿Par trenzado requerido?	No, pero recomendado	Si
Impedancia de modo común (ohmios)	No especificado	30 ± 30%
Impedancia característica diferencial (ohmios)	No especificado	90
Cable *skew* (picosegundos)	< 100	
Tipo de cable (AWG)	28 o inferior	
Resistencia DC (ohmios)	0.6	
Retardo del cable	18 nanosegundos	5.2 nanosegundos por metro
Localización de la resistencia *pull-up* en el dispositivo	D-	D+
¿Cable desmontable?	No	Si
¿Cable prisionero?	Si	

Tabla 1.10 Requisitos de cables y componentes low speed y full/high speed[1].

En un cable *full/high speed*, las líneas de señal deben tener una impedancia característica diferenciada de 90 ohmios. Este valor es una medida de la impedancia de entrada de una línea infinita, línea abierta y determina la corriente inicial en las líneas cuando el interruptor de salida conmuta.

La especificación enumera los requisitos para los conductores, blindaje y el aislamiento del cable. Éstos son los principales requisitos para los cables de *full/high-speed*:

[1] Tabla tomada del libro "*USB COMPLETE Second Edition*" de JAN AXELSON

- Línea de datos: par trenzado, AWG #28.
- Energía y tierra: no-trenzado, AWG #20 a #28.
- Línea de drenaje: alambre de cobre trenzado, estañado, AWG #28.
- Protector interno: poliéster metalizado aluminio.
- Protector externo: cobre trenzado, estañado.

La especificación también enumera los requisitos para la durabilidad y el funcionamiento del cable. Un dispositivo *low speed* puede utilizar un cable *full speed* si el cable satisface todos los requerimientos del cable *low speed*.

1.8.2.2 Conectores

La especificación 2.0 describe dos tipos de conectores: serie A para el extremo del cable que se conecta al *host* o al concentrador y serie B para el extremo del cable que se conecta al dispositivo. Figura 1.13.

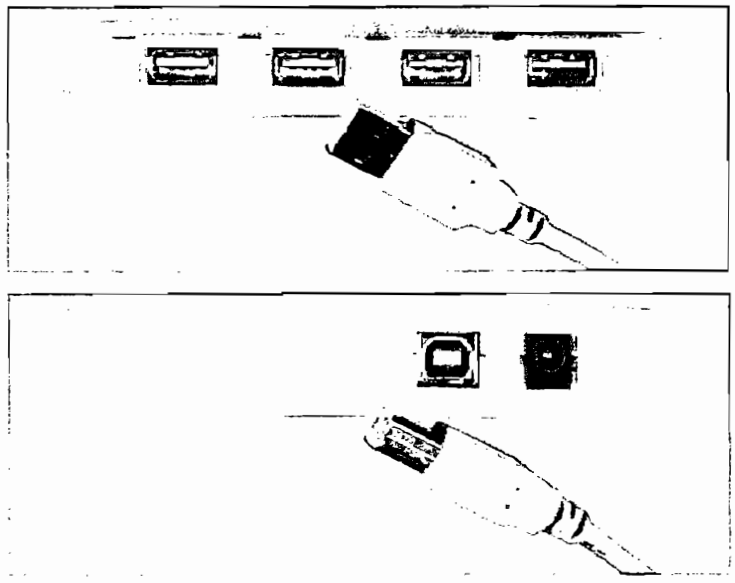


Figura 1.13 Conector de la serie A (superior) y conector de la serie B (inferior)[1].

Los conectores de la serie B eran voluminosos para algunos dispositivos, así que un ECN (*Engineering Change Notice*) a la especificación 2.0 agregó una opción para un nuevo conector mini-B. Figura 1.14.

[1] Tabla tomada del libro "*USB COMPLETE Second Edition*" de JAN AXELSON

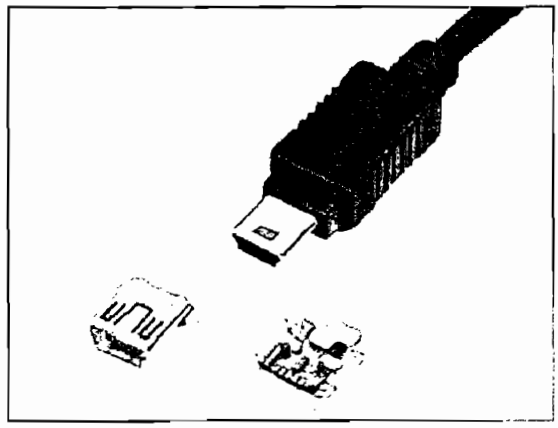


Figura 1.14 El conector mini-B

Un receptáculo mini-B es menos de la mitad de la altura de un receptáculo de la serie B. Usted puede utilizar los conectores mini-B donde quiera que pueda utilizar los conectores de la serie B. Cada cable debe tener un conector de la serie A, pero no todos los cables requieren conectores de la serie B o conector mini-B. Un concentrador típico tendrá un receptáculo de la serie B en su puerto de *uplink*. El receptáculo acepta los conectores de la serie B en un cable que se conecte con el concentrador raíz u otro concentrador.

Las conexiones de la señal se ahuecan levemente para asegurarse de que las líneas de energía se conecten primero cuando se conecta un cable. Todos los conectores tienen conexiones para las dos líneas de señal del bus, la fuente de VBUS y tierra. El conector mini-B tiene un pin adicional de identificación. La especificación da las asignaciones de pines y el color para el cable y los conectores como lo indica la Tabla 1.11.

Pin Series A/B	Pin Mini-B	Conductor	Color del Cable
1	1	VBUS (+5V)	Rojo
2	2	D-	Blanco
3	3	D+	Verde
4	5	GND	Negro
-	4	ID	No conectado
Shell		Protector	Línea de drenaje

Tabla 1.11 Asignación de pines y de color para el cable y conectores USB[1].

[1] Tabla tomada del libro "*USB COMPLETE Second Edition*" de JAN AXELSON

1.8.2.3 Cables desmontables y prisioneros

La especificación define los cables como desmontables o prisioneros. Un cable desmontable debe ser *full/high speed*, con un conector de la serie A para la conexión al concentrador raíz o el *host* 2.0 y un conector de la serie B o mini-B para la conexión con el dispositivo. Los cables genéricos USB ofrecidos por los varios vendedores son de este tipo.

Un cable prisionero puede ser *low* o *full/high speed*. El extremo que se conecta al *host* o al concentrador tiene un conector de la serie A. Para la conexión del otro extremo, el cable puede ser conectado permanentemente, o puede ser desprendible con un conector del tipo no estandarizado.

1.8.2.4 Longitud del cable

La versión 1.0 de la especificación del USB especificó las longitudes máximas para los segmentos del cable. Un segmento *full speed* podría tener una longitud máxima de 5 metros y un segmento *low speed* podría tener una longitud máxima de 3 metros.

En los cables *full/high speed*, las limitaciones se dan debido a la atenuación de la señal, retardo de propagación en el cable, y la caída de voltaje en las líneas VBUS y de tierra.

En los cables *low speed*, la longitud se limita por los tiempos de subida y de caída de las señales, la carga capacitiva presentada por el segmento, y las caídas de voltaje en las líneas de VBUS y de tierra. Los límites originales de 3 y 5 metros siguen siendo buenas pautas generales.

La especificación del USB prohíbe la extensión de los cables, que elimina la tentación de estirar un segmento más allá de los límites físicos de la interfaz. Una opción para cubrir largas distancias, es utilizar un cable estándar del USB que se conecte con un dispositivo que traduzca entre el USB y RS-485 u otra interfaz diseñada para ser utilizada en largas distancias.

interfaz de control de audio permite acceso a controles tales como volumen, modo *silencio*, bajos y altos.

1.9.5 *MÓDEMS*

El controlador del módem incluido con *Windows 98 SE* y posteriores es compatible con los módems que utilizan otro tipo de interfaz.

1.10 CÓDIGOS DE LÍNEA

Una vez analizada la interfaz USB y sus especificaciones, se realiza una breve revisión de los códigos de línea. La codificación de línea es una técnica que permite adecuar las señales digitales al medio de transmisión, manteniendo la naturaleza de la señal. Tomando en consideración lo expuesto se hace necesario que los códigos de línea cumplan con los siguientes parámetros:

1.10.1 PARÁMETROS

1.10.1.1 Eficiencia espectral

Al realizar el análisis espectral de un código se debe tomar en cuenta la presencia de la componente continua, que es la causa de la generación de dificultades en la recepción. Un código debe presentar el menor ancho de banda posible en banda base, además se debe tomar en cuenta que las componentes espectrales fuera de la banda principal sean insignificantes, de modo que si se usan para modular una portadora, no generen muchas señales espurias.

1.10.1.2 Sincronismo y Transparencia

Un código debe incorporar información de reloj en los datos, que permita al receptor sincronizarse para detectar claramente los límites de tiempo de cada símbolo recibido, esto se logra incorporando transiciones suficientes en los datos, estas transiciones no deben aumentar el ancho de banda.

1.10.1.3 Capacidad de detección de errores

Los códigos de línea deben permitir al receptor detectar la aparición de errores en la recepción, incorporando la información necesaria para cumplir este propósito.

1.10.1.4 Baja probabilidad de errores

La inmunidad al ruido, es otro de los parámetros a cumplirse, de modo que el receptor no incurra en muchos errores en la detección de los símbolos recibidos, a este parámetro se conoce como BER (*Bit Error Rate*) o tasa de errores.

1.10.2 CLASES DE CÓDIGOS DE LÍNEA

Tomando en cuenta los parámetros antes mencionados, existen diferentes Códigos de Línea que se clasifican según la polaridad y por su forma general, de los cuales para la implementación se han tomado en consideración los siguientes:

- Código NRZ Polar
- Código AMI
- Código MLT-3
- Código RZ Polar
- Código Manchester Diferencial
- Código Miller
- Código CMI
- Código Bifase M
- Código 4B3T
- Código MS43
- Código B3ZS
- Código HDB3

A continuación en la figura 1.15 se muestra un ejemplo de codificación para una secuencia de datos dada y para cada uno de los códigos mencionados.

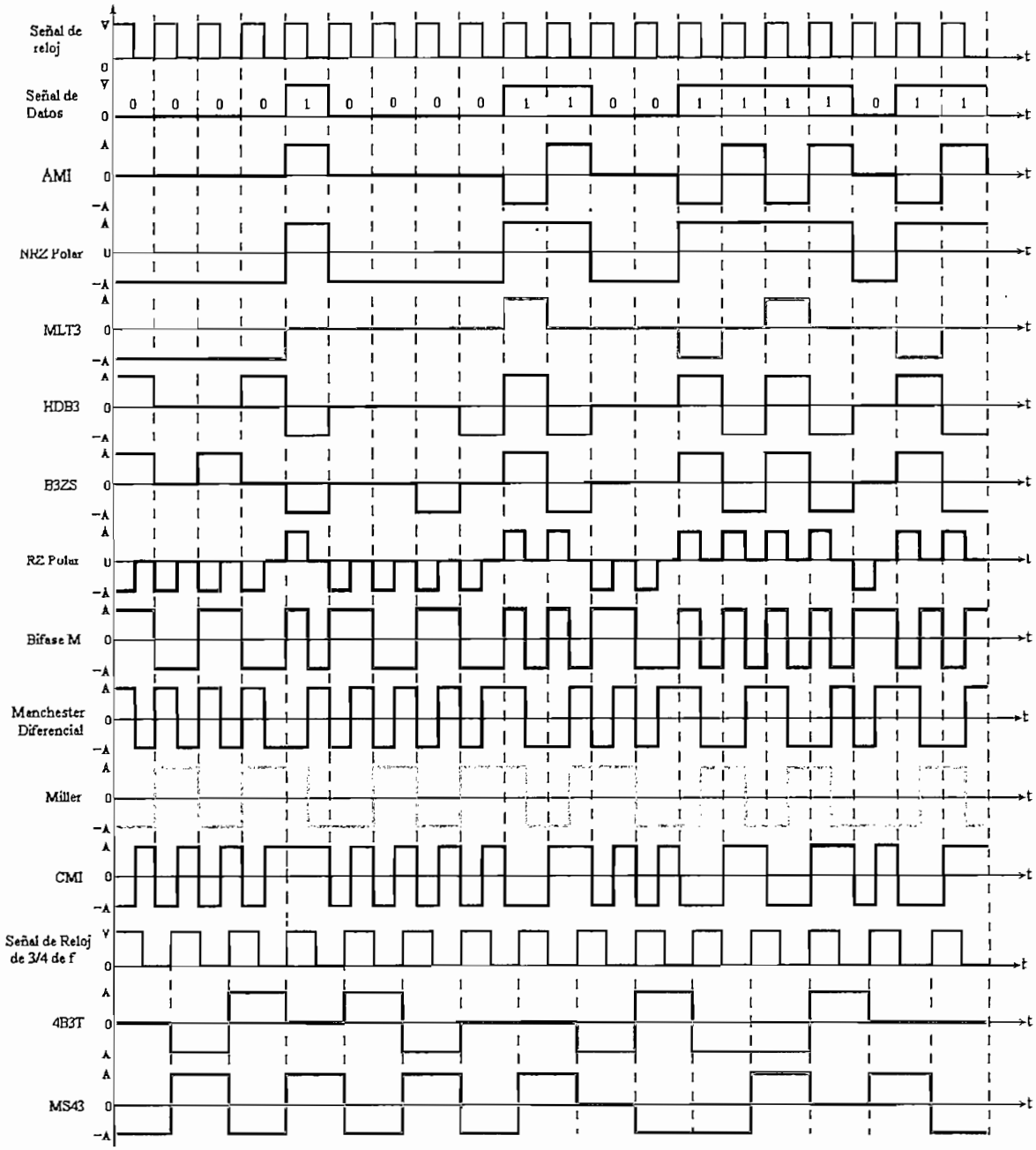


Figura 1.15 Ejemplo de tipos de Codificación

En el ejemplo de la figura 1.15 las señales de codificación tienen un uno lógico como estado precedente, y con un nivel de codificación precedente de –A, además en los códigos B3ZS y HDB3 se ha considerado que el número de unos lógicos precedentes ha sido impar. Para los códigos MS43 y 4B3T se ha considerado una polaridad acumulada de +2.

Para mayor información de la forma de codificar con los diferentes códigos que se muestran en la figura 1.15, se debe referir al anexo A.

En La tabla 1.12 se presenta un resumen de las características que presentan algunos de los códigos a implementarse.

CÓDIGO	CARACTERÍSTICA
AMI	Pérdida de sincronismo en cadenas largas de ceros.
NRZ Polar	Perdida de sincronismo con secuencias largas de 1 o 0 lógicos
B3ZS	Garantizan continua información de sincronismo independiente de la fuente de datos.
HDB3	Características de sincronización mejoradas, puesto que evita cadenas largas de ceros.
MLT3	El efecto del esquema de codificación MLT-3 es concentrar la mayor parte de la energía en la señal transmitida bajo los 30 MHz, lo cual reduce las emisiones radiadas.
RZ Polar	El muestreo de la señal codificada es mas dificil a medida que disminuye el ancho del pulso.
Bifase M	Posibilita la detección de errores en caso de ausencia de transición esperada.
Miller	Moderada recuperación de sincronismo.
Manchester Diferencial	Posibilita la detección de errores en caso de ausencia de transición esperada.
CMI	En este código se repite la equiprobabilidad de polaridad y la consiguiente minimización de componente DC.
4B3T	Presenta gran cantidad de potencia a bajas frecuencias.
MS43	Presenta gran cantidad de potencia a bajas frecuencias.

Tabla 1.12 Algunas características de los Códigos de Línea

1.11 DEFINICIÓN DE TÉRMINOS

HOST: Es el computador que controla la interfaz.

FUNCION: Como un dispositivo que proporciona una capacidad al *host*. Los ejemplos de funciones son un ratón, un sistema de altavoces.

CONCENTRADOR. Es un dispositivo que contiene uno o más conectores o conexiones internas a los dispositivos USB, junto con el hardware para permitir la comunicación con cada dispositivo.

DISPOSITIVO O PERIFERICO. Es algo que se conecta a un puerto USB en un PC o concentrador. La definición oficial de un dispositivo es una función o un concentrador a excepción de un dispositivo compuesto, que contiene un concentrador y una o más funciones.

PIPE: Consiste en una vía de comunicación entre el endpoint de un dispositivo y el software del controlador del *host*.

TRASFERENCIA: La especificación USB define una transferencia como el proceso de la creación y realización de una petición de comunicación.

TRANSACCIÓN: La especificación define una transacción como la entrega del servicio a un endpoint.

CONTROLADOR USB DEL HOST: es aquel permite la administración del tiempo disponible para la comunicación de los diferentes dispositivos USB con el host.

CONCENTRADOR RAÍZ: es aquel que está formado por uno o más conectores que permiten la conección de los dispositvos al host.

CAPÍTULO II

DISEÑO Y CONSTRUCCIÓN DEL DISPOSITIVO

DISEÑO Y CONSTRUCCIÓN DEL DISPOSITIVO

Este capítulo está orientado al diseño y construcción del CODEC didáctico. El diseño contempla la justificación de cada uno de los elementos utilizados en la construcción del dispositivo.

2.1 REQUERIMIENTOS

Para un diseño lo más importante es tomar en cuenta los requerimientos que debe cumplir determinado proyecto, para el presente dispositivo uno de los requerimientos que se presenta es que el CODEC tenga capacidad para realizar comunicaciones en modo *full dúplex*. Además, el CODEC debe recibir señales binarias tanto en niveles TTL o niveles RS – 232, las mismas que deben ser procesadas de acuerdo al tipo de código seleccionado.

En la codificación, la señal de entrada podrá ser una señal TTL o una señal RS – 232, estas señales serán generadas por cualquier fuente de datos que maneje este tipo de niveles como lo es el computador o el generador de secuencias digitales.

En la decodificación, la señal de entrada será la señal codificada que se quiere decodificar para obtener los correspondientes niveles binarios RS – 232 y/o niveles TTL.

Para la operación del CODEC es necesario seleccionar los siguientes parámetros:

- Código de línea
- Velocidad de transmisión

2.1.1 CÓDIGO DE LÍNEA

Los códigos de línea que se pueden seleccionar en el CODEC didáctico son los siguientes: NRZ Polar, RZ Polar, AMI, Bifase M, Manchester Diferencial, CMI, Miller, HDB3, B3ZS, 4B3T, MS43, MLT3.

2.1.2 VELOCIDAD DE TRANSMISIÓN

Las velocidades de transmisión que se pueden seleccionar son: 300, 600, 1200, 2400, 4800, 9600 y 19200 bits por segundo.

Para seleccionar el tipo de código y la velocidad de transmisión, el dispositivo debe presentar alguna opción que permita variar estos parámetros, dicho mecanismo está basado en la comunicación con el computador utilizando el interfaz USB.

Con respecto a los niveles de voltaje que se van a utilizar en el codificador / decodificador se manejan + 5 voltios, 0 voltios y – 5 voltios, estos niveles no son niveles de voltaje normalizados, por el mismo hecho que el dispositivo es para uso didáctico, demostrativo y no de uso comercial, en donde para cumplir este objetivo se debería cumplir las normas establecidas por los organismos reguladores.

2.2 DISEÑO

El circuito codificador / decodificador consta de los siguientes bloques:

- Transmisión y Recepción de la señal de datos
- Transmisión y Recepción de la configuración del dispositivo
- Procesamiento de la señal de datos codificados
- Procesamiento de la señal de datos a decodificar
- Sincronización
- Unidad Central de Procesos (C.P.U.)

La figura 2.1 muestra el diagrama en bloques de cada una de las etapas de las que está formado el CODEC.

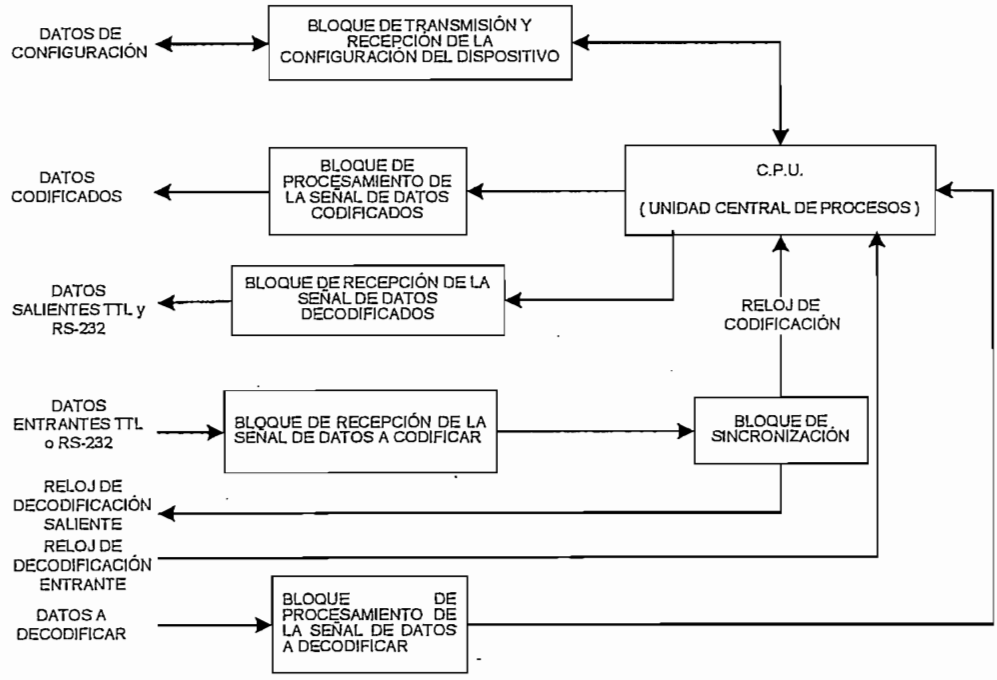


Figura 2.1 Diagrama de bloques del Codificador / decodificador.

La descripción de cada uno de los bloques que conforman el circuito, se realiza tomando en cuenta su grado de dificultad, por lo que empezaremos con el siguiente bloque:

2.2.1 BLOQUE DE TRANSMISIÓN Y RECEPCIÓN DE LA SEÑAL DE DATOS

Este bloque permite adecuar las señales de datos para la transmisión y recepción, ya sean niveles RS-232 o niveles TTL, para lo cual se utiliza:

El circuito integrado MAX-232, que es un circuito conversor de niveles que permite cambiar los niveles lógicos RS-232 a niveles lógicos TTL y viceversa. Las características más relevantes de este circuito integrado son las siguientes:

- Requiere una fuente de + 5 voltios para su operación.
- Conversor DC – DC de +5 V a ± 12 voltios.
- Tecnología CMOS

- Baja potencia – I_{cc} máxima de 3 mA.
- Posee dos entradas RS-232 y sus dos entradas TTL.
- Cumple con el Standard TIA/EIA-232-E y CCITT V.28.

En este bloque a más de recibir y transmitir señales RS-232, muy importantes para la comunicación con el PC, se tiene un acondicionador de señales TTL, que está formado por un amplificador operacional LM318N de la *National Semiconductor*, cuyas características más importantes son:

- Ancho de banda de 15 MHz.
- Opera con fuentes de ± 5 voltios a ± 20 voltios.
- Posee una compensación interna de frecuencia.
- Las entradas y salidas están protegidas contra sobrecargas.
- Permite la configuración como un rápido emisor seguidor de voltaje.

El amplificador operacional LM318N, se lo utiliza en el diseño como seguidor de voltaje[1] para acoplar la señal entrante al circuito. Además la entrada positiva se conecta a una resistencia de *pull-up* para mantener la entrada en alto. Si se toma en cuenta que la corriente de entrada debe ser menor o aproximadamente igual a 10 mA, y el voltaje de alimentación de + 5V se tiene:

$$R_{12} = \frac{V}{I} \approx \frac{5 \quad V}{10 \quad mA} \approx 500 \quad \Omega$$

Puesto que no existe el valor normalizado de 500 Ω se utiliza un valor normalizado de 1000 Ω (se asume este valor debido a que a mayor resistencia menor será la corriente que circule por la misma).

Para seleccionar el nivel de entrada se ha colocado un multiplexor, el cual dependiendo de la selección se tendrá a la salida el nivel de entrada, además nos permitirá visualizar dicha selección mediante leds.

[1] El circuito seguidor de voltaje fue tomado de la hoja de datos LM318N de la *National Semiconductor*. Anexo B.

A continuación se muestran las conexiones de cada uno de los circuitos en la figura 2.2

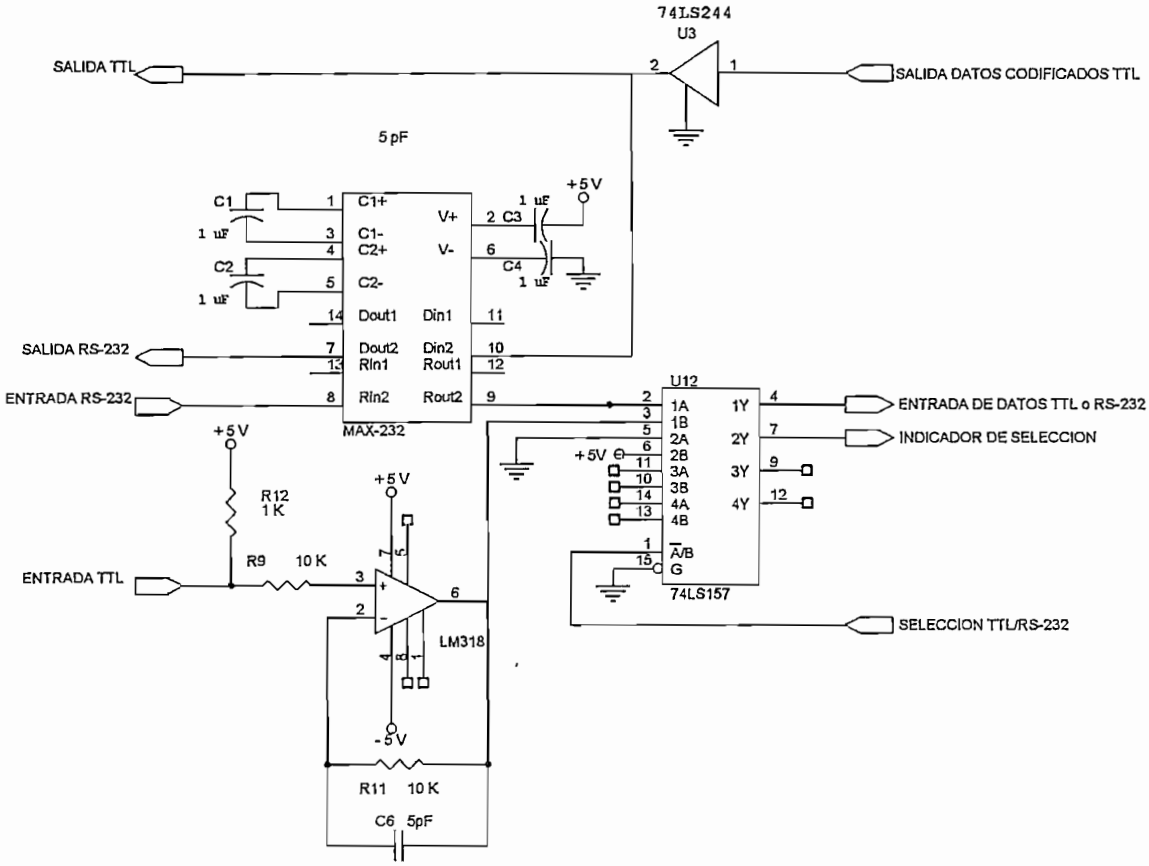


Figura 2.2 Circuito de transmisión y recepción de la señal de datos.

Las salidas que se tienen en este bloque deben ser sincronizadas con el reloj generado en el bloque de sincronización para poder realizar lecturas a medio período de bit en la etapa de codificación.

2.2.2 BLOQUE DE TRANSMISIÓN Y RECEPCIÓN DE LA CONFIGURACIÓN DEL DISPOSITIVO

La transmisión y recepción de los datos necesarios para la configuración del dispositivo se realiza utilizando la interfaz USB, que permite la comunicación con el computador. Para poder comunicar el bloque de la CPU vía la interfaz USB se utiliza el módulo USBMOD2 de RAVAR, el cual permite realizar la comunicación con el microcontrolador en forma paralela y comunicarse con el computador personal utilizando la interfaz USB.

El módulo USBMOD2 permite la transferencia de datos desde y hacia el computador, se basa en el circuito integrado FTDI FT8U245 USB FIFO. Entre las características[1] mas importantes del módulo están:

- Módulo de 32 pines.
- Módulo USB de alta velocidad.
- Posee un cristal de 6 MHz.
- Posee un conector tipo B.
- No requiere de elementos pasivos externos.
- Transferencia de datos de USB a paralelo.

Para leer la memoria FIFO del módulo USBMOD2 se utilizan las señales RXF y RD, en tanto que las señales que permiten la escritura son TXE y WR; además de las señales que permiten la lectura y escritura en la memoria FIFO del módulo USB MOD2, se utilizan las líneas D0 a D7 para capturar el dato enviado desde o hacia el computador.

En la figura 2.3 se muestran los diagramas de temporización de lectura y escritura de la memoria FIFO del módulo USBMOD2.

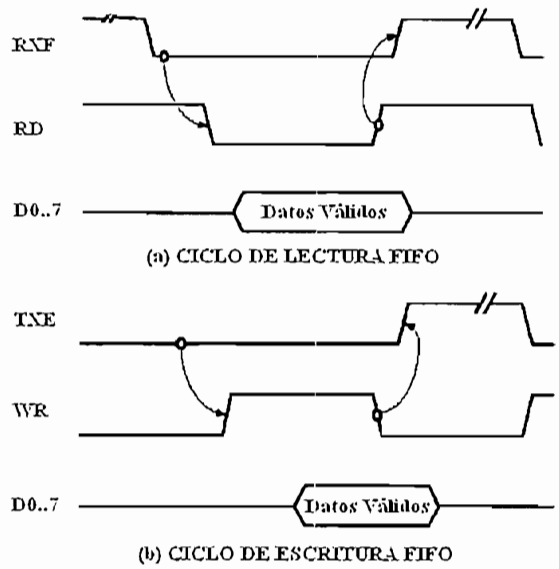


Figura 2.3 Ciclos de Temporización de la FIFO[2].

[1] Características tomadas del Manual de usuario del módulo USBMOD2. Anexo B.
[2] Ciclos de temporización tomados de la hoja de datos del circuito integrado FT245AM. Anexo B.

En la tabla 2.1 se muestran cada una de las señales utilizadas para la lectura y escritura.

N° PIN	SEÑAL	TIPO	FUNCION
7	RXF	Salida	En 1L no lee los datos de la FIFO. En 0L permite leer el dato disponible en la FIFO siempre que la señal RD pase de 1L a 0L.
23	RD	Entrada	En 0L toma los datos del bus de datos (D0 a D7). Al pasar de 0L a 1L permite capturar el siguiente dato disponible en la FIFO.
24	WR	Entrada	Permite la escritura en el bus de datos cuando pasa de 1L a 0L.
25	TXE	Salida	Cuando se encuentra en 1L no permite la escritura de datos en la FIFO. Cuando se encuentra en 0L permite la escritura de datos en la FIFO, siempre y cuando la señal WR pase de 1L a 0L.
8	D7	Entrada / Salida	Bit # 7 del bus de datos Bi direccional
9	D6	Entrada / Salida	Bit # 6 del bus de datos Bi direccional
10	D5	Entrada / Salida	Bit # 5 del bus de datos Bi direccional
11	D4	Entrada / Salida	Bit # 4 del bus de datos Bi direccional
12	D3	Entrada / Salida	Bit # 3 del bus de datos Bi direccional
13	D2	Entrada / Salida	Bit # 2 del bus de datos Bi direccional
14	D1	Entrada / Salida	Bit # 1 del bus de datos Bi direccional
15	D0	Entrada / Salida	Bit # 0 del bus de datos Bi direccional

Tabla 2.1 Distribución de pines de las señales del MÓDULO USBMOD2[1].

Explicada la lectura y escritura de datos en la FIFO del módulo USBMOD2, se presenta en la figura 2.4 el diagrama de conexiones.

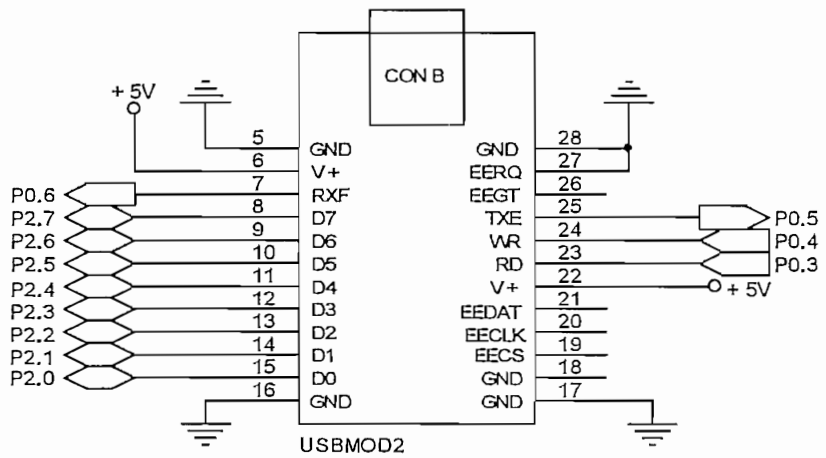


Figura 2.4 Circuito transmisor y receptor de la configuración del CODEC.

[1] Características de las señales tomadas del manual de usuario del módulo USBMOD2. Anexo B.

2.2.3 BLOQUE DE PROCESAMIENTO DE LA SEÑAL DE DATOS CODIFICADOS

En este bloque se procede a generar la señal de datos codificada, para lograr este objetivo es necesario utilizar 3 distintos niveles de voltaje + 5V, 0V y – 5V. Los niveles cambian dependiendo de los valores que entregue la CPU, para esto debemos tomar en cuenta las siguientes combinaciones para cada uno de los pines P1.1 Y P1.3 del microcontrolador Atmel At89C52 de la tabla 2.2.

COMBINACIÓN		VOLTAJES	PINES
PIN P1.1 (MUX B)	PIN P1.3 (MUX A)	X	
0 L	0 L	+ 5V	X_0
0 L	1 L	0V	X_1
1 L	0 L	- 5V	X_2
1 L	1 L	0 V	X_3

Tabla 2.2 Niveles de voltaje con sus correspondientes valores binarios.

Estos dos valores ingresan a un multiplexor el cual permite obtener una señal dependiendo de los valores que ingresen a las entradas de selección.

Luego la señal a la salida del multiplexor debe ser acoplada para que pueda ser tomada por cualquier dispositivo o equipo, para esto se coloca el Amplificador operacional LM310N en configuración seguidor de voltaje[1].

Cabe recalcar que el circuito pertenece única y exclusivamente a la etapa de codificación de datos.

El circuito de este bloque se muestra en la figura 2.5 con los circuitos integrados que lo forman.

[1] La hoja de datos del LM310N de la *National Semiconductor* se encuentra en el Anexo B.

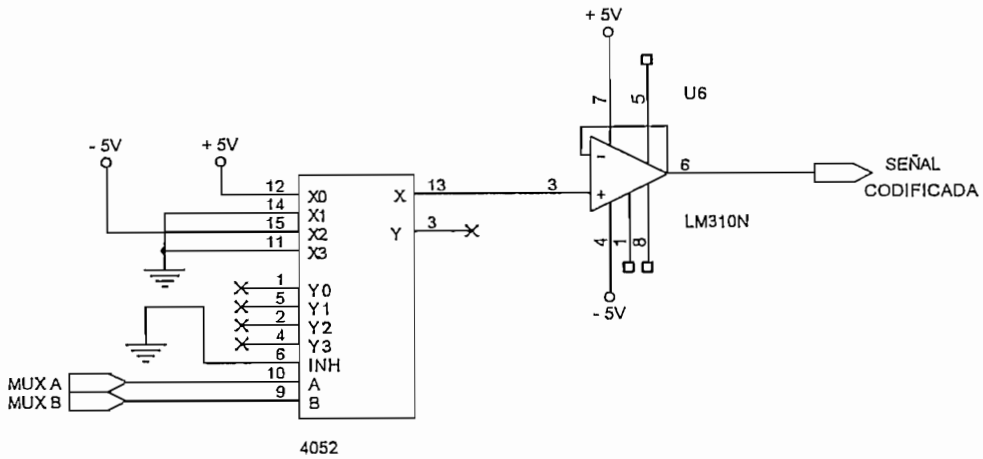


Figura 2.5 Circuito procesador de la señal de datos codificada.

2.2.4 BLOQUE DE PROCESAMIENTO DE LA SEÑAL DE DATOS A DECODIFICAR

Para decodificar la señal, es necesario determinar en base a circuitos la magnitud y el signo de la señal, para ello se toma la señal a decodificar, se la ingresa a la entrada del seguidor de voltaje LM310N[1] para poder acoplar la señal.

A continuación la señal se la divide en dos rutas para que ingrese a dos comparadores de alta velocidad. El comparador de alta velocidad utilizado en el diseño es el circuito integrado LM319N, que entre sus características principales se tienen:

- Dos comparadores independientes.
- Fuente de alimentación de +5 voltios a +15 voltios.
- El tiempo de respuesta típico a +15 voltios es de 80 ns.
- Máxima corriente de entrada de 10 μA operando sobre un rango de temperatura.

Los comparadores, van a permitir determinar la magnitud y el signo de la señal entrante. A continuación se explica como se determina la señal de magnitud y signo al utilizar los comparadores de voltaje LM319N[1].

[1] La hoja de datos del LM310N de la *National Semiconductor* se encuentra en el Anexo B.

En el primer comparador, la entrada negativa (-) se la fija a un voltaje de referencia aproximado de +1.5 V utilizando un potenciómetro de precisión de 50 KΩ el cual está asignado como un nivel de umbral para detectar un valor de voltaje positivo, mientras que en la entrada positiva (+) ingresa la señal entrante para este comparador. A la salida de este comparador se obtiene el valor en binario del signo de la señal entrante.

En el segundo comparador, la entrada positiva (+) se establece un voltaje de referencia de aproximadamente –1.5 V utilizando un potenciómetro de precisión de 50 KΩ el cual está asignado como un nivel de umbral para detectar un valor de voltaje negativo, mientras que en la entrada negativa (-) se conecta la señal entrante.

Para obtener el valor en binario de la magnitud de la señal entrante, es decir para reconocer si se trata de un nivel de voltaje + 5 V, 0 V ó – 5 V, se debe colocar a las entradas de una compuerta lógica OR, las salidas de los dos comparadores.

En la tabla 2.3 se muestra los niveles de voltaje que representan tanto la señal del signo como la señal de la magnitud de la señal entrante.

NIVELES LOGICOS		SEÑAL ENTRANTE
SIGNO	MAGNITUD	VOLTAJE (V)
1 L	1 L	+ 5
0 L	0 L	0
0 L	1 L	- 5

Tabla 2.3 Relación de los Niveles de voltaje con los valores binarios de magnitud y signo.

La figura 2.6 muestra el circuito procesador de la señal a decodificar con sus respectivos componentes.

[1] Las características del LM319N tomadas de la hoja de datos de la *Nacional Semiconductor*. Anexo B.

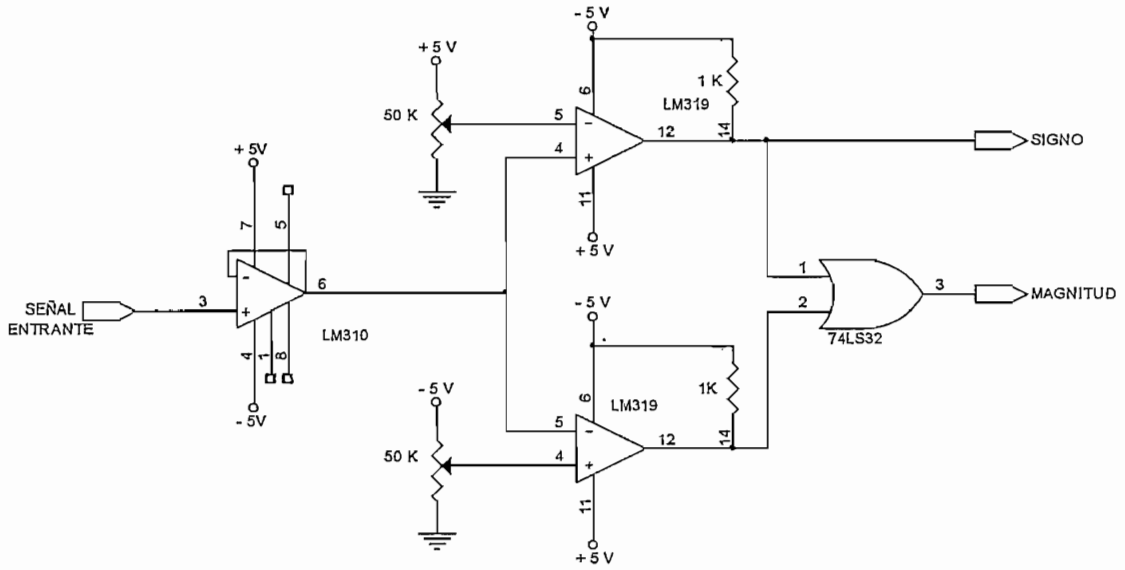


Figura 2.6 Circuito procesador de la señal a decodificar

El circuito de este bloque es utilizado únicamente por la etapa de decodificación de la señal de datos.

2.2.5 BLOQUE GENERADOR DEL RELOJ DE SINCRONISMO Y SINCRONIZACIÓN DE LOS DATOS

La generación de la señal de reloj de codificación y decodificación está a cargo del microcontrolador 16F870[1] de la Microchip, el cual entre sus *características* presenta:

- Velocidad de operación: 20 MHz reloj de entrada como máximo.
- Memoria flash de 2048 x 14 palabras.
- Memoria RAM de 128 x 8 bytes.
- Memoria EEPROM de 64 x 8 bytes.
- 16 registros de funciones especiales.
- Pines de entrada y salida con control de direccionamiento individual.
- 3 temporizadores / contadores.
- Transmisión y recepción serial asíncrona.
- Rango de voltaje de operación es de 3.0V a 5.5V.

[1] Mayor información acerca del PIC 16F870. Anexo B.

Debido a las características del microcontrolador PIC 16F870 y especialmente por el retardo por cada ciclo de máquina se lo ha escogido para que sea el dispositivo que genere la señal de reloj de codificación y la señal de reloj de decodificación, así como, el sincronismo de las señales de reloj con sus respectiva señal de datos entrantes (datos a codificar).

El programa del microcontrolador debe realizar las siguientes tareas:

- Generación del reloj de codificación y decodificación por defecto, con una frecuencia de 300 Hz para el código AMI.
- Generación del sincronismo entre los datos entrantes sin codificar, con el reloj de codificación.
- Generación del sincronismo entre la señal decodificada y el reloj de decodificación.
- Rutina de interrupción para la comunicación serial, la misma que permita configurar la frecuencia de las señales de reloj, dependiendo del tipo de código que se va a utilizar en el *Codec*, pues la señal de reloj de decodificación que utilizan los códigos AMI, NRZ Polar, MLT3, HDB3 y B3ZS no es aplicable a los códigos Bifase M, Manchester diferencial, Miller, CMI, NZ Polar, 4B3T y MS43, pues necesitan frecuencias de reloj de decodificación del doble o de 4/3 a la frecuencia de codificaciòn.

La figura 2.7 muestra el diagrama de flujo de los parámetros por defecto que se configuran en el microcontrolador 16F870.

En la figura 2.8 se muestra el diagrama de flujo del programa principal que permite la generación de las señales de reloj y la sincronización de los datos.

Dentro del diagrama de flujo de la figura 2.8 se tiene la etapa que permite la sincronización de los datos a codificar y la forma de generar las señales de reloj de codificación y decodificación. La sincronización de estas señales se realiza tomando muestras de la señal del dato a codificar, esto se debe a que hay que

muestrearlas cada medio período de bit al momento de ser procesadas por la Unidad Central de Procesos.

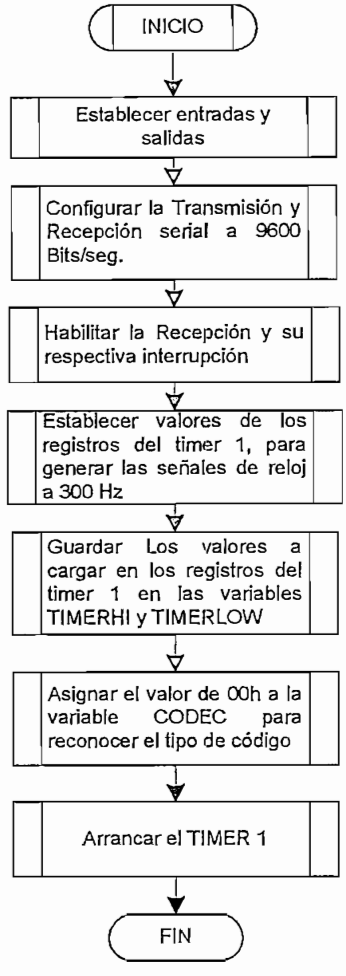
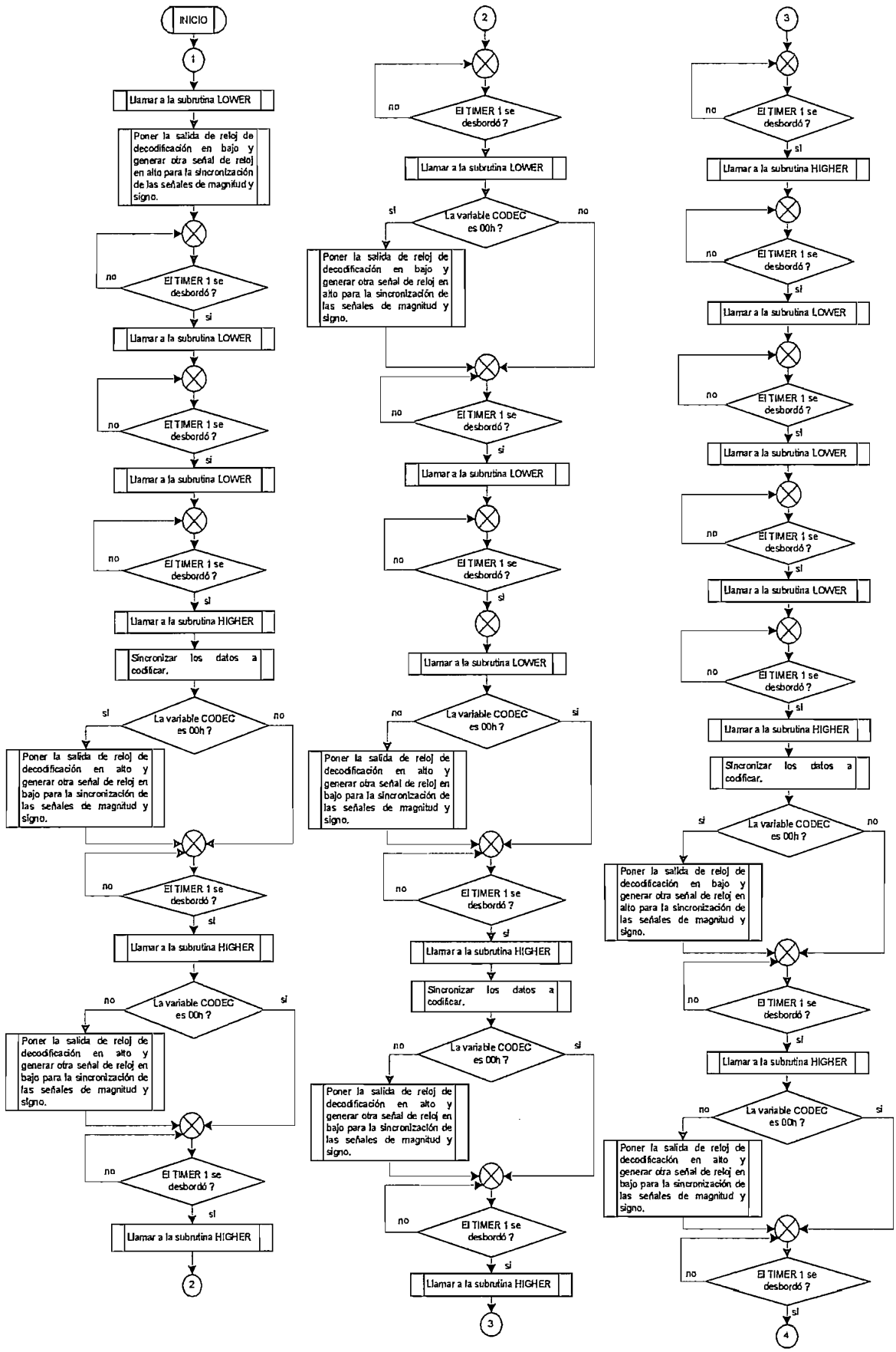


Figura 2.7 Diagrama de flujo de la configuración de parámetros del microcontrolador 16F870.

Para generar las diferentes señales de reloj que requieren los distintos códigos para su posterior procesamiento, se les ha dividido a las señales de reloj en tres categorías según el código de línea seleccionado.

La primera categoría contempla los códigos de línea AMI, NRZ Polar, MLT3, HDB3, B3ZS, estos códigos necesitan una señal de reloj de codificación igual a la señal de reloj de decodificación.


INICIO
1
Llamar a la subrutina LOWER
Poner la salida de reloj de decodificación en bajo y generar otra señal de reloj en alto para la sincronización de las señales de magnitud y signo.
no
El TIMER 1 se desbordó ?
si
Llamar a la subrutina LOWER
no
El TIMER 1 se desbordó ?
si
Llamar a la subrutina LOWER
no
El TIMER 1 se desbordó ?
si
Llamar a la subrutina HIGHER
Sincronizar los datos a codificar.
si
La variable CODEC es 00h ?
no
Poner la salida de reloj de decodificación en alto y generar otra señal de reloj en bajo para la sincronización de las señales de magnitud y signo.
no
El TIMER 1 se desbordó ?
si
Llamar a la subrutina HIGHER
no
La variable CODEC es 00h ?
si
Poner la salida de reloj de decodificación en alto y generar otra señal de reloj en bajo para la sincronización de las señales de magnitud y signo.
no
El TIMER 1 se desbordó ?
si
Llamar a la subrutina HIGHER
2
2
no
El TIMER 1 se desbordó ?
Llamar a la subrutina LOWER
si
La variable CODEC es 00h ?
no
Poner la salida de reloj de decodificación en bajo y generar otra señal de reloj en alto para la sincronización de las señales de magnitud y signo.
no
El TIMER 1 se desbordó ?
si
Llamar a la subrutina LOWER
no
El TIMER 1 se desbordó ?
si
Llamar a la subrutina LOWER
no
La variable CODEC es 00h ?
si
Poner la salida de reloj de decodificación en alto y generar otra señal de reloj en bajo para la sincronización de las señales de magnitud y signo.
no
El TIMER 1 se desbordó ?
si
Llamar a la subrutina HIGHER
Sincronizar los datos a codificar.
no
La variable CODEC es 00h ?
si
Poner la salida de reloj de decodificación en alto y generar otra señal de reloj en bajo para la sincronización de las señales de magnitud y signo.
no
El TIMER 1 se desbordó ?
si
Llamar a la subrutina HIGHER
3
3
no
El TIMER 1 se desbordó ?
si
Llamar a la subrutina HIGHER
no
El TIMER 1 se desbordó ?
si
Llamar a la subrutina LOWER
no
El TIMER 1 se desbordó ?
si
Llamar a la subrutina LOWER
no
El TIMER 1 se desbordó ?
si
Llamar a la subrutina LOWER
no
El TIMER 1 se desbordó ?
si
Llamar a la subrutina HIGHER
Sincronizar los datos a codificar.
si
La variable CODEC es 00h ?
no
Poner la salida de reloj de decodificación en bajo y generar otra señal de reloj en alto para la sincronización de las señales de magnitud y signo.
no
El TIMER 1 se desbordó ?
si
Llamar a la subrutina HIGHER
no
La variable CODEC es 00h ?
si
Poner la salida de reloj de decodificación en alto y generar otra señal de reloj en bajo para la sincronización de las señales de magnitud y signo.
no
El TIMER 1 se desbordó ?
si
4

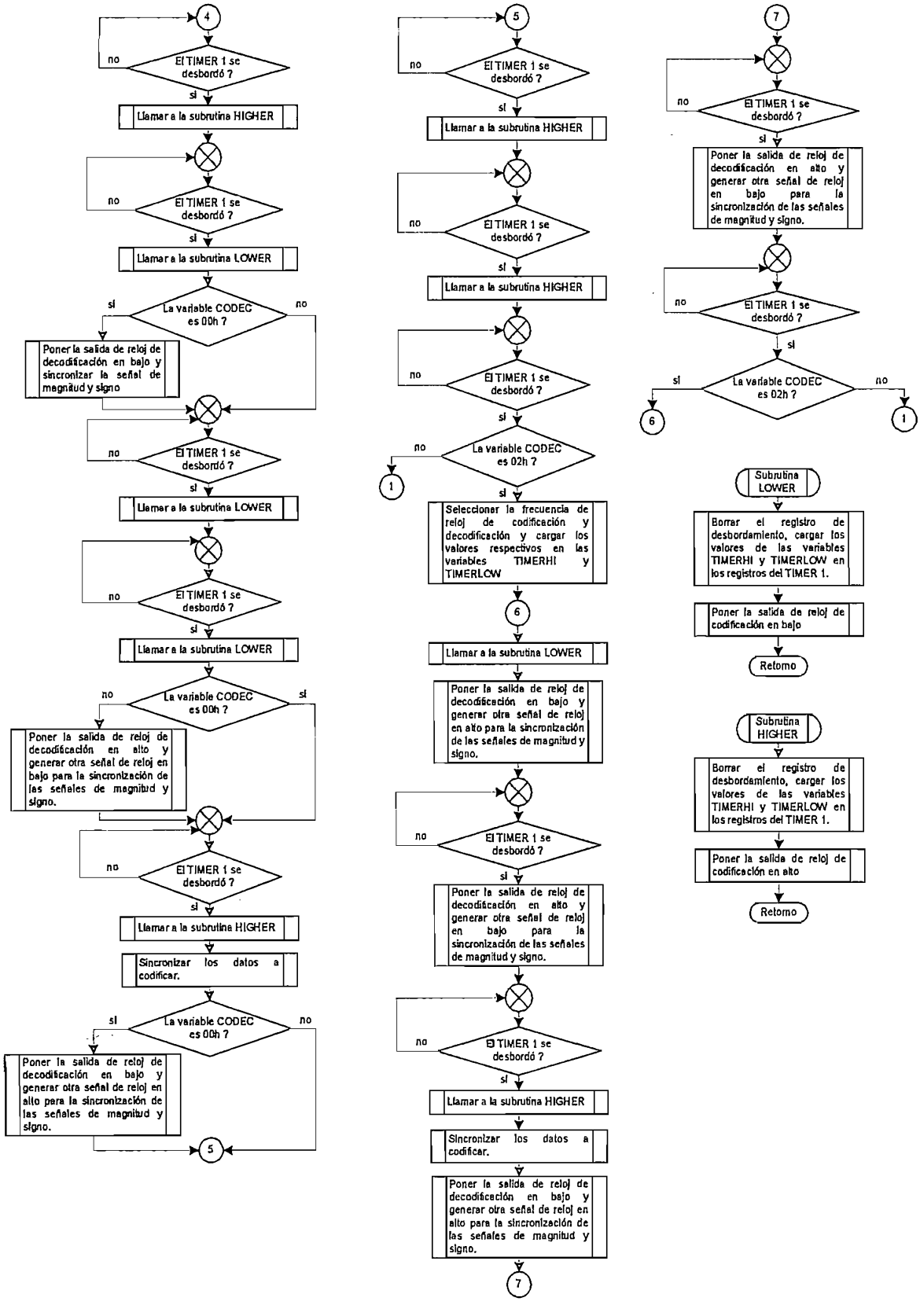


Figura 2.8 Diagrama de flujo del programa principal

La segunda categoría abarca los códigos de línea RZ Polar, Bifase M, CMI, Miller y Manchester Diferencial, estos códigos utilizan para su procesamiento una señal de reloj de decodificación del doble de la frecuencia del reloj de codificación, esto se debe a que la señal codificada posee transiciones a medio período de *bit*.

En la tercera categoría se encuentran los códigos de línea 4B3T y MS43, estos códigos necesitan para su decodificación una señal de reloj con una frecuencia igual a los tres cuartos de la frecuencia del reloj de codificación.

En el diagrama de flujo de la figura 2.8 se puede observar que la señal de reloj de codificación y la señal de reloj de decodificación tienen divididos sus niveles alto y bajo en tres partes iguales, esto se debe a que mediante esta segmentación se puede generar dos tipos de señales de reloj, una señal de reloj para los códigos AMI, NRZ Polar, MLT3, HDB3, B3ZS y la otra para los códigos 4B3T y MS43. Para generar la señal de reloj de los códigos RZ Polar, Manchester Diferencial, Bifase M, CMI, Miller se ha creado un algoritmo que divide la señal de reloj en dos partes iguales, este algoritmo tiene el mismo lineamiento que el anterior algoritmo es por eso que no se lo presenta.

Para que el microcontrolador P16F870 procese el tipo de señal de reloj que deben generar, se utiliza dentro del programa principal la variable CODEC como se observa en el diagrama de flujo, la cual puede almacenar cualquiera de los siguientes valores 00H, 01H o 02H, estos valores están asociados a los diversos códigos que posee el codificador / decodificador, como se muestra en la tabla 2.4.

Valor de la variable CODEC	Tipos de Códigos
00H	AMI, NRZ Polar, LMT3, HDB3, B3ZS
01H	RZ Polar, Miller, CMI, Bifase M, Manchester Diferencial
02H	4B3T o MS43

Tabla 2.4 Valores para el reconocimiento del tipo de código

Para realizar el cambio de frecuencia de las señales de reloj, así como el código al que se le ha de aplicar estas señales de reloj, se ha creado una subrutina la

misma que es utilizada cuando se produce la interrupción por recepción de los datos de configuración. Los datos de configuración que recibe el microcontrolador son mediante el puerto serial como se muestran en la tabla 2.5.

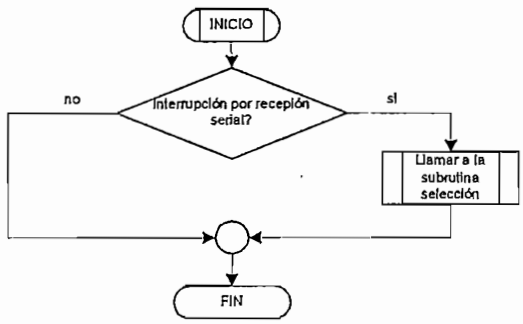
ASIGNACION DE LETRAS PARA CADA VELOCIDAD DE TRANSMISIÓN	300	A
	600	B
	1200	C
	2400	D
	4800	E
	9600	F
	19200	G
LETRAS DE SELECCIÓN DE CODIGOS	4B3T o MS43	H
	AMI, NRZ Polar, LMT3, HDB3, B3ZS	I
	RZ Polar, Miller, CMI, Bifase M, Manchester Diferencial	J

Tabla 2.5 Valores de configuración enviados por la CPU.

La recepción de los datos en forma serial se realizan a una velocidad de 9600 *bits/segundo*, un *bit* de inicio, ocho *bits* de datos, sin paridad y un *bit* de parada.

En la figura 2.9 se muestra el diagrama de flujo de la interrupción serial así como de la subrutina para cambiar la frecuencia de las señales de reloj o el tipo de código mediante el envío de alguno de los valores de configuración que se encuentran en la tabla 2.4.

Luego de presentar la información acerca del programa generador y sincronizador de los datos, se presenta en la figura 2.10 el diagrama de conexiones que se tienen en el PIC 16F870 para cada una de las señales que intervienen en la etapa de sincronización.



(a) Interrupción Serial

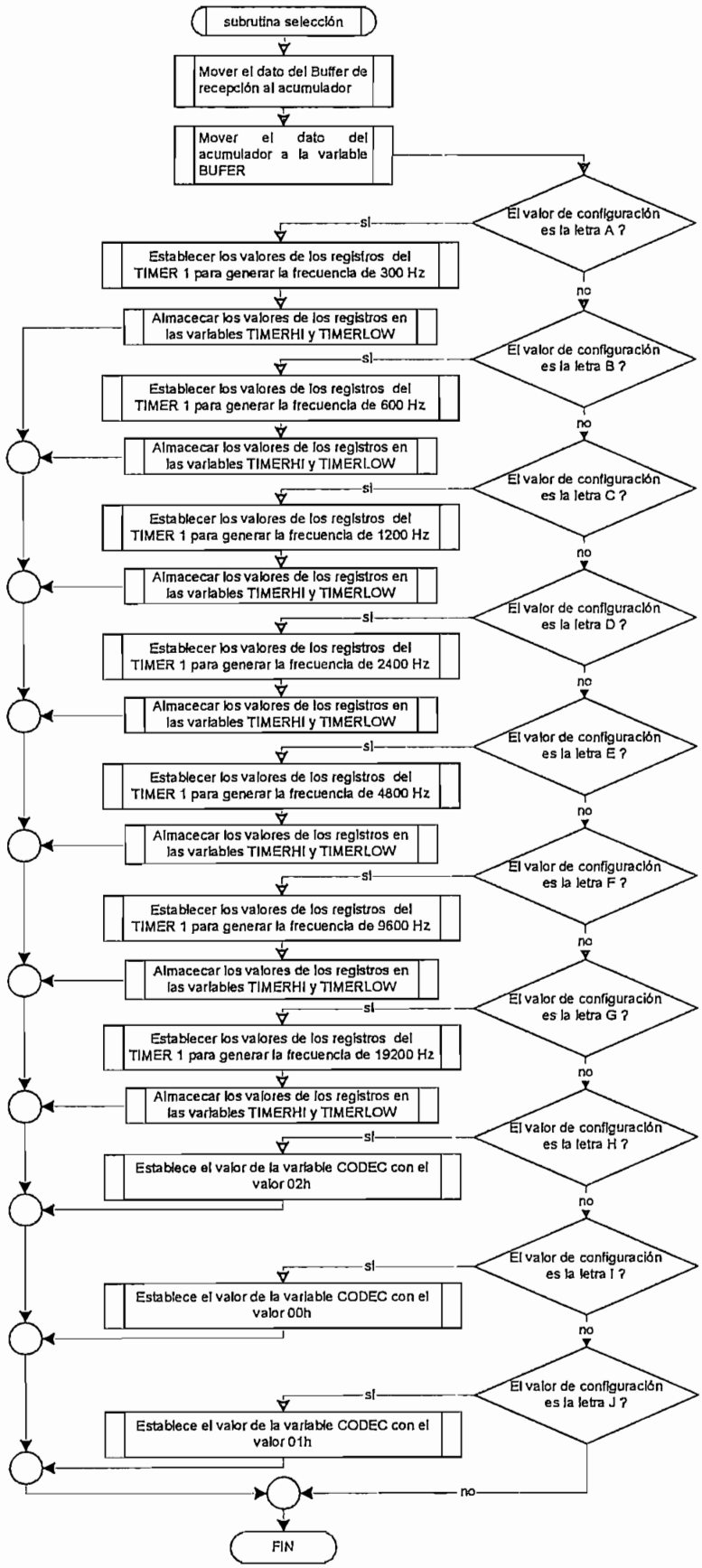


Figura 2.9 Diagrama de flujo de la Rutina de Interrupción de recepción.

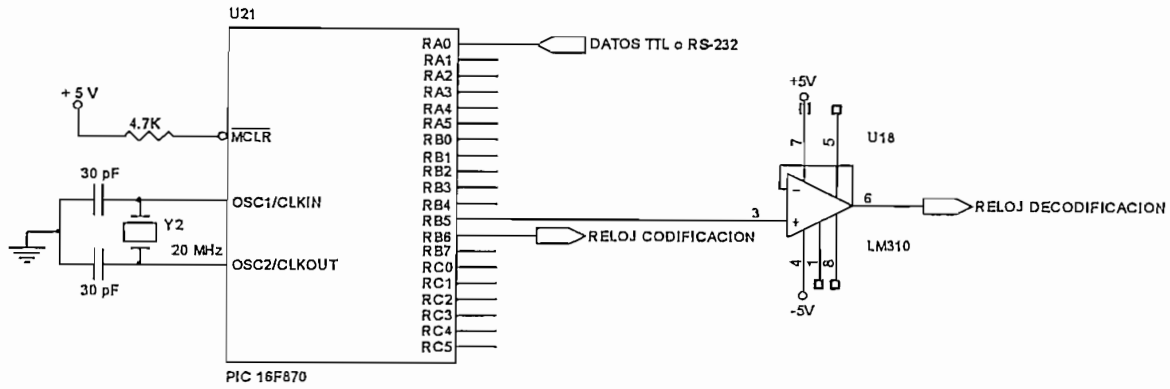


Figura 2.10 Circuito de la etapa de Sincronización.

2.2.6 UNIDAD CENTRAL DE PROCESOS

El cerebro del dispositivo es el bloque de la Unidad Central de Procesos pues este bloque es el encargado de recibir los datos de configuración provenientes del computador para luego procesarlos.

Además es el encargado de recibir la señal de datos sincronizada con la señal de reloj de codificación, para procesarlos dependiendo del código seleccionado y a continuación enviar las respectivas señales que permitan gobernar la salida del multiplexor analógico. También este bloque está encargado de procesar las señales digitales (1L y/o 0L) que representan la magnitud y el signo de la señal a decodificar.

Es por esto que el programa de la Unidad Central de Procesos está estructurado por los siguientes eventos:

- Inicialización del dispositivo.
- Configuración del dispositivo.
- Codificación.
- Decodificación.

La inicialización del dispositivo ocurre un instante después de ser alimentado el circuito a través del interfaz USB. La inicialización del dispositivo se expone en el diagrama de flujo de la figura 2.11.

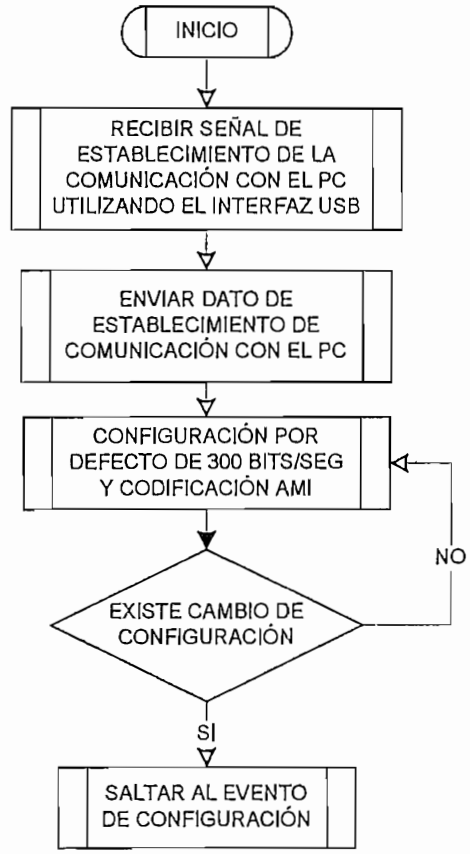


Figura 2.11 Diagrama de flujo de la inicialización del dispositivo

Una vez inicializado el dispositivo es posible cambiar su configuración, es decir seleccionar el código de línea y la velocidad de transmisión según lo requerido. La configuración del dispositivo se realiza según se explica en el diagrama de flujo de la figura 2.12.

Antes de realizar el salto a la codificación seleccionada, se envía la configuración de velocidad al PIC 16F870 para que genere las señales de reloj de codificación y decodificación respectivas. Los valores que reciben los PIC fueron expuestas en la tabla 2.4.

Una vez configurado el dispositivo, éste empieza a codificar y decodificar, al final de cada algoritmo de codificación siempre se procede a verificar si existe un cambio en la configuración.

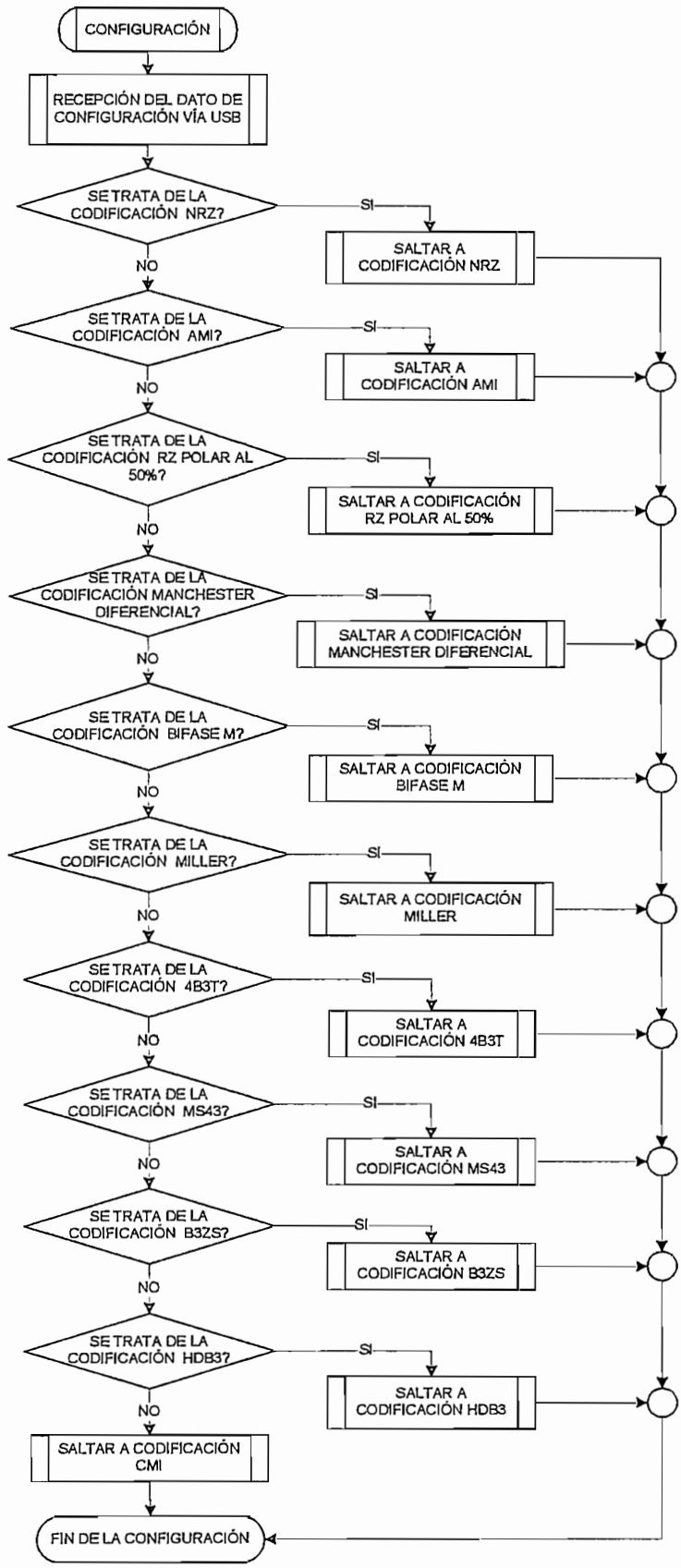


Figura 2.12 Diagrama de flujo de la Configuración del dispositivo.

Los códigos de línea que genera la unidad central de procesos son:

- Código AMI
- Código NRZ Polar
- Código RZ Polar
- Código Manchester Diferencial
- Código Bifase M
- Código Miller
- Código CMI
- Código B3ZS
- Código HDB3
- Código 4B3T
- Código MS43
- Código MLT3

A continuación se explican los algoritmos para generar la codificación en base a sus diagramas de flujo.

2.2.6.1 Código AMI

- **Codificación AMI**

La codificación AMI presenta su algoritmo de codificación en el diagrama de flujo de la figura 2.13a.

En la codificación AMI se espera una transición descendente para poder codificar por lo que se tiene un retardo de medio periodo de bit en el procesamiento de la señal codificada.

- **Decodificación AMI**

La decodificación AMI se produce mediante la interrupción externa 1, la cual se activa con la transición descendente de la señal de reloj, esto permite tomar la

lectura de la señal codificada a medio periodo de bit, como se puede ver según el diagrama de flujo la decodificación es fácil pues lo único que hay que tomar en cuenta es la señal de magnitud que es la que indica si se trata de un 1L o un 0L, esto se puede observar en el diagrama de flujo de la figura 2.13b.

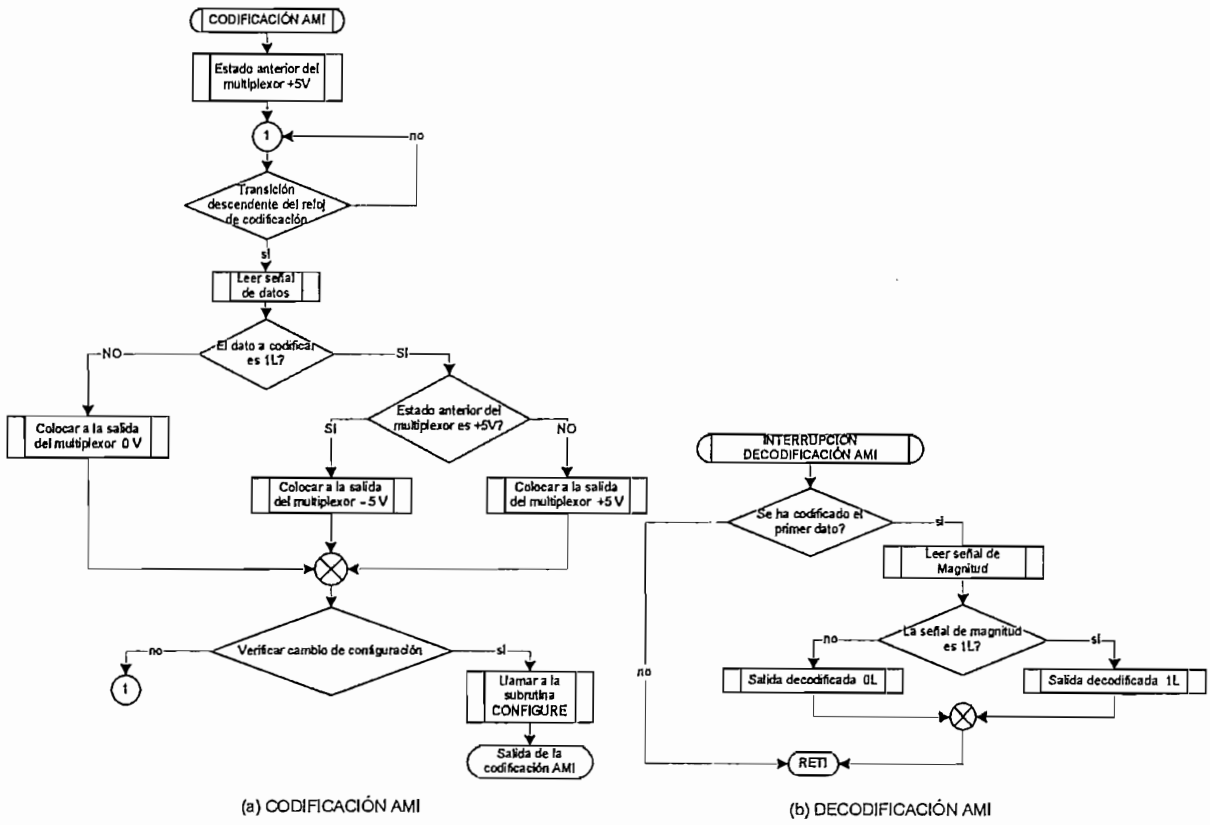


Figura 2.13 Diagrama de flujo de la Codificación y Decodificación AMI.

2.2.6.2 Código NRZ Polar

- **Codificación NRZ Polar**

La codificación NRZ Polar al igual que el código AMI presenta un retardo de medio periodo de bit en el procesamiento, pues se debe esperar que se produzca una transición descendente para poder hacer la lectura del dato a medio periodo de bit, esto se presenta en el diagrama de flujo de la figura 2.14a.

➢ **Decodificación NRZ Polar**

La decodificación NRZ Polar se ejecuta una vez producido el evento de interrupción, mediante la llamada a subrutina, esto se puede observar en el diagrama de flujo de la figura 2.14b.

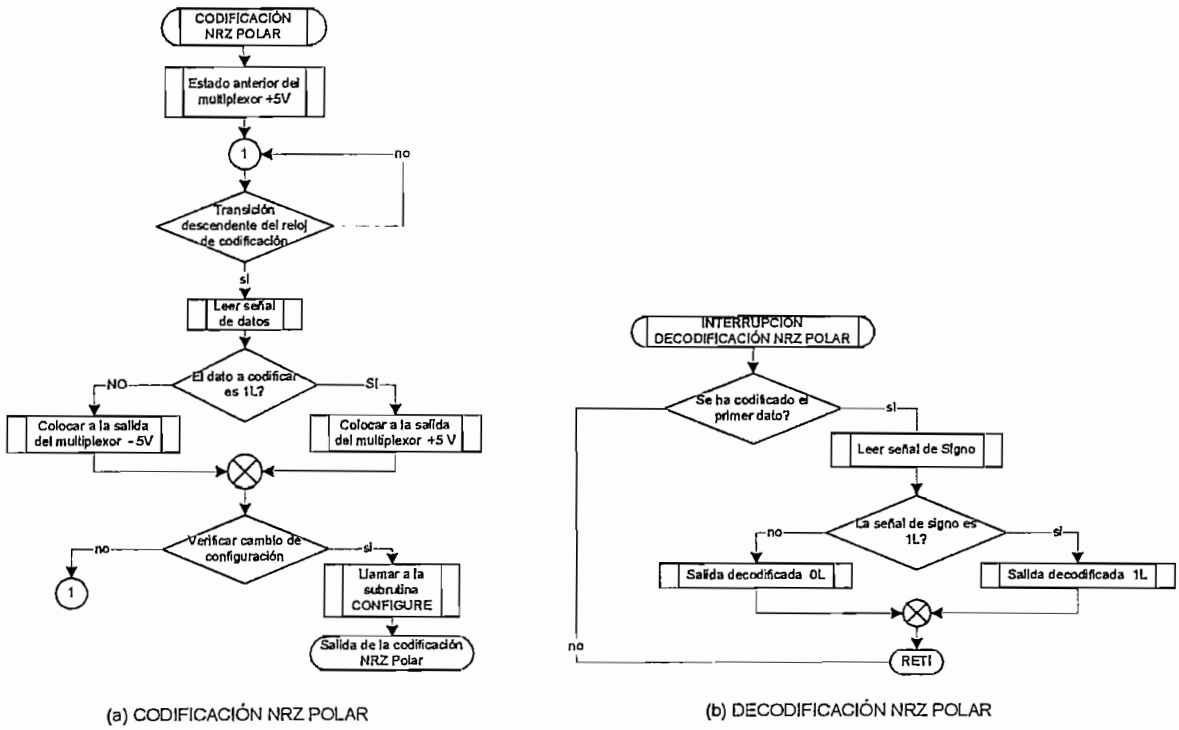


Figura 2.14 Diagrama de flujo de la Codificación y Decodificación NRZ Polar.

2.2.6.3 Código RZ Polar

➢ **Codificación RZ Polar**

Para codificar los datos se debe hacer la lectura en la transición descendente y enviar inmediatamente el nivel que se deba tener a la salida del multiplexor según la regla de codificación, luego en la transición ascendente se hace que la señal retorne a cero, el algoritmo se presenta en el diagrama de flujo de la figura 2.15a.

➢ **Decodificación RZ Polar**

Como en las anteriores decodificaciones se espera el evento de interrupción, pero lo diferente es que este código utiliza para su decodificación un reloj del doble de frecuencia de la señal de reloj de codificación, todo el procedimiento de la decodificación RZ Polar se presenta en el diagrama de flujo de la figura 2.15b.

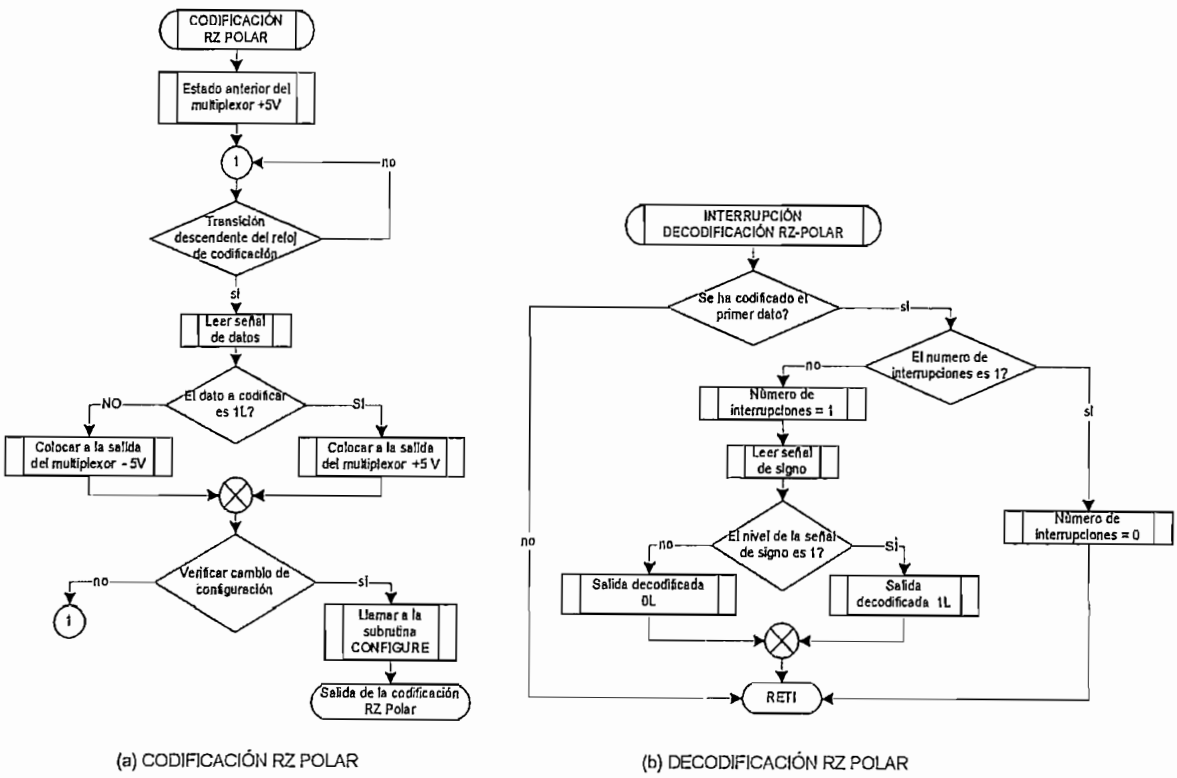


Figura 2.15 Diagrama de flujo de la Codificación y Decodificación RZ Polar.

2.2.6.4 Código Manchester Diferencial

➢ **Codificación Manchester Diferencial**

La codificación Manchester Diferencial al igual que la codificación RZ Polar, para generar la señal codificada necesita de las dos transiciones que genera el reloj. En el diagrama de flujo de la figura 2.16 se muestra el algoritmo de codificación.

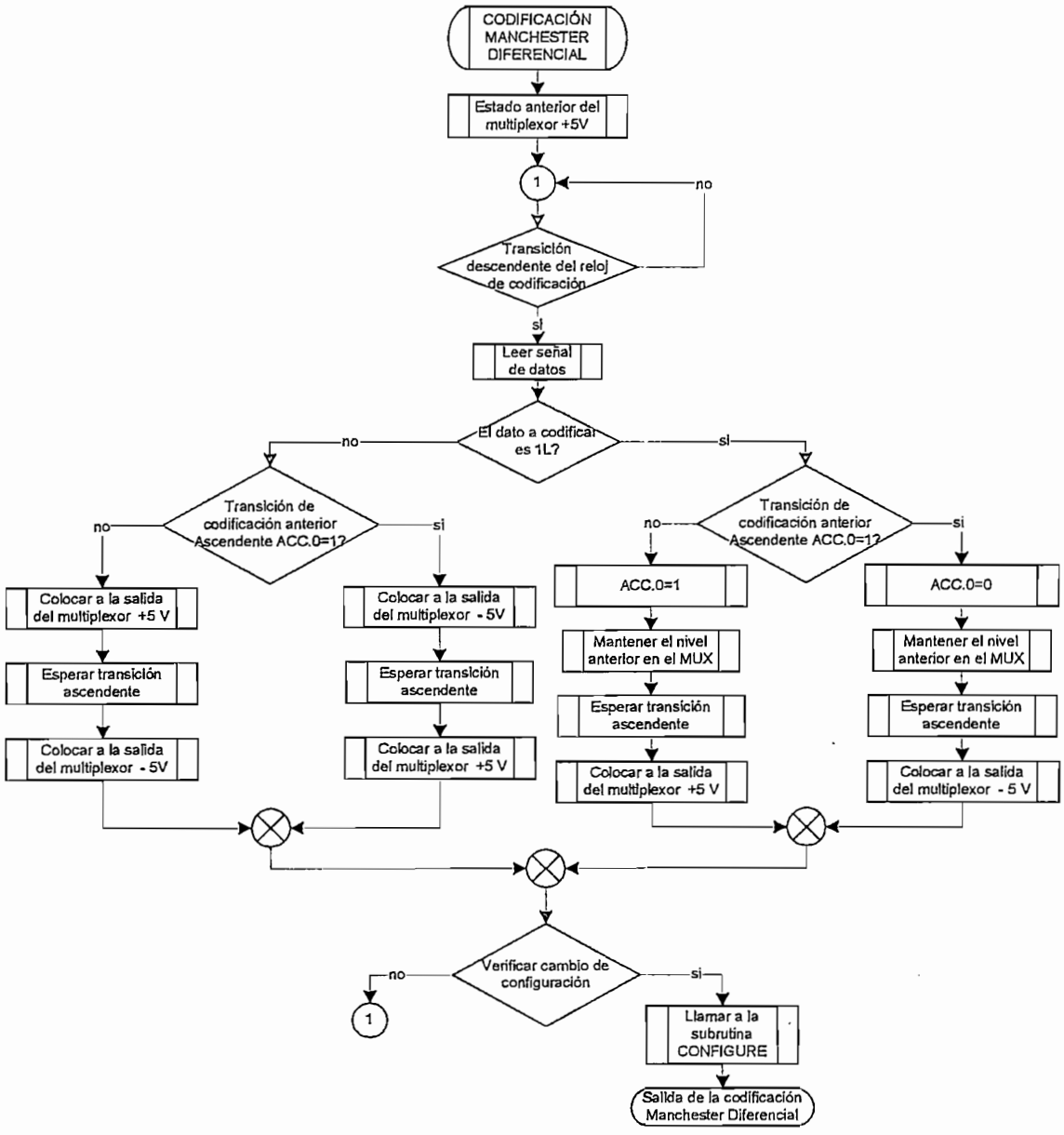


Figura 2.16 Diagrama de flujo de la Codificación Manchester Diferencial.

➢ Decodificación Manchester Diferencial

La decodificación Manchester Diferencial se produce al ejecutarse la interrupción externa 1, el reloj que se utiliza para la decodificación es del doble de frecuencia del reloj de decodificación. En el diagrama de flujo de la figura 2.17 se presenta el esquema de decodificación.

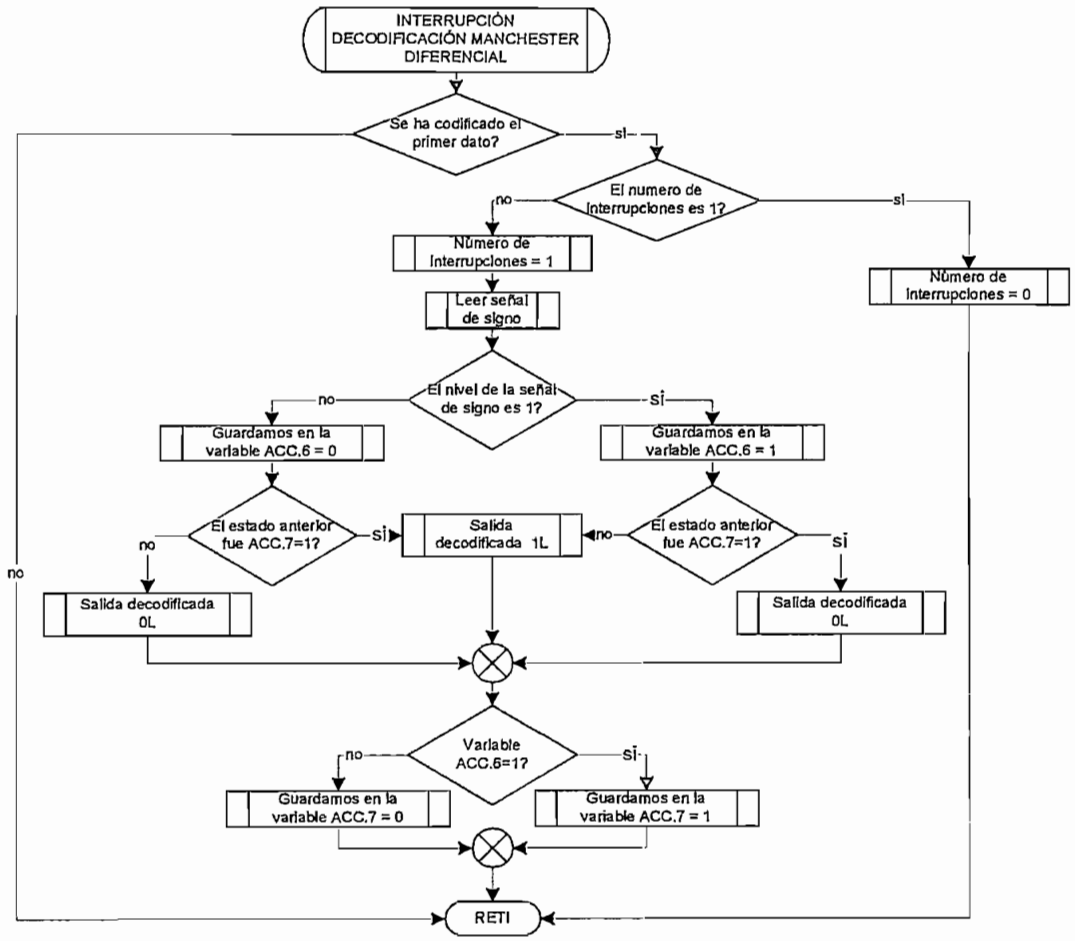


Figura 2.17 Diagrama de flujo de la Decodificación Manchester Diferencial.

2.2.6.5 Código Bifase M

➢ **Codificación Bifase M**

En esta codificación así, como en todas las codificaciones Bifase se debe codificar utilizando las transiciones producidas por el reloj de codificación. En el diagrama de flujo de la figura 2.18 se presenta el esquema de codificación.

➢ **Decodificación Bifase M**

Para la decodificación es necesario tomar en cuenta que la señal utilizada para este fin es la señal de signo, que es la señal que lleva la información de decodificación, debido a que esta señal indica la existencia o no de transiciones a medio o al inicio del bit en la señal codificada. En el diagrama de flujo de la figura

2.19 se presenta el esquema de decodificación, en el cual se toma muy en cuenta el número de interrupciones que se han producido.

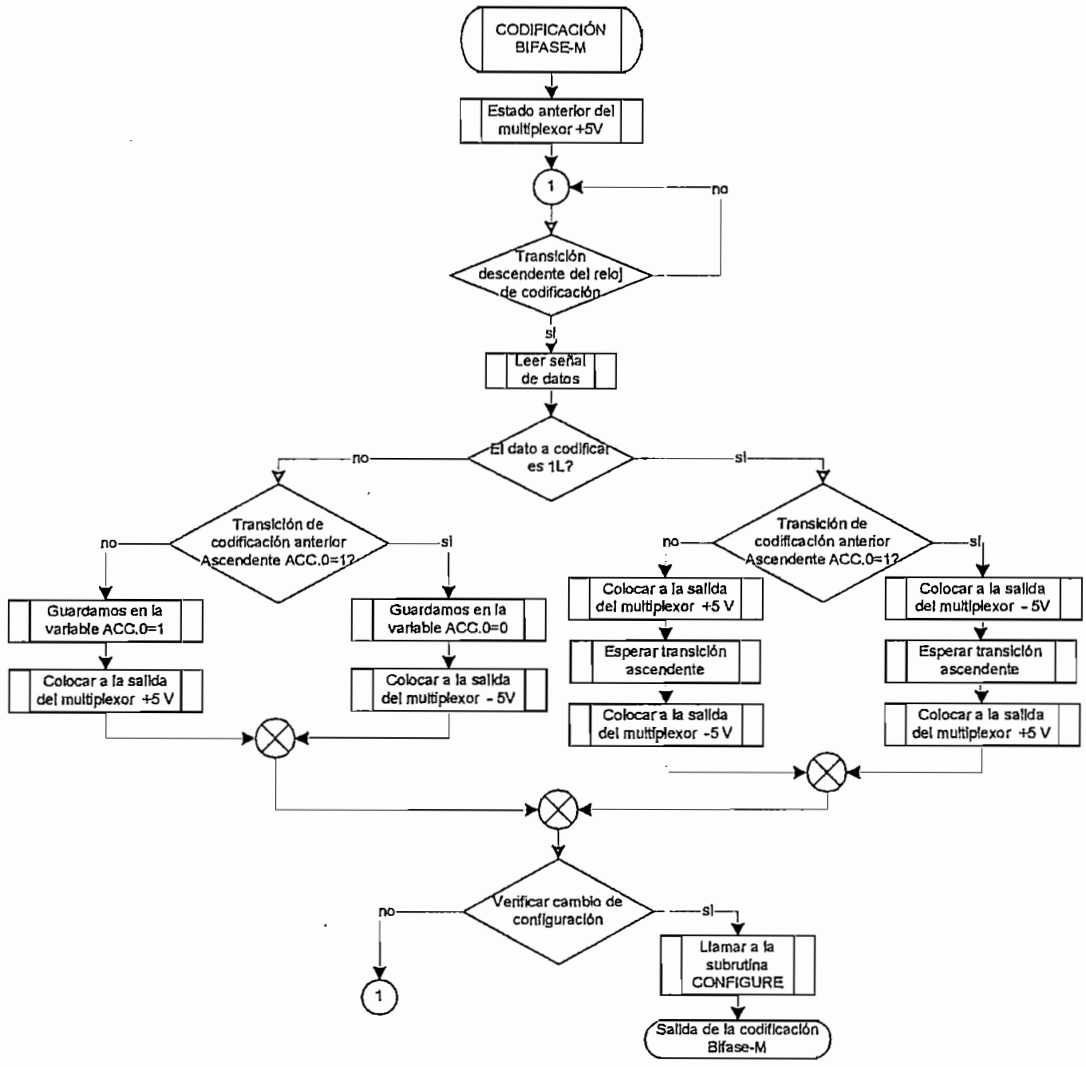


Figura 2.18 Diagrama de flujo de la Codificación Bifase M.

2.2.6.6 Código CMI

- **Codificación CMI**

El diagrama de flujo de la figura 2.20 para esta codificación muestra la forma en la cual se ha realizado el algoritmo de codificación. Como se puede observar en el diagrama se tiene un estado anterior de + 5V y para empezar a codificar es necesario esperar una transición descendente, y a continuación de acuerdo al algoritmo empezar a generar la señal de salida.

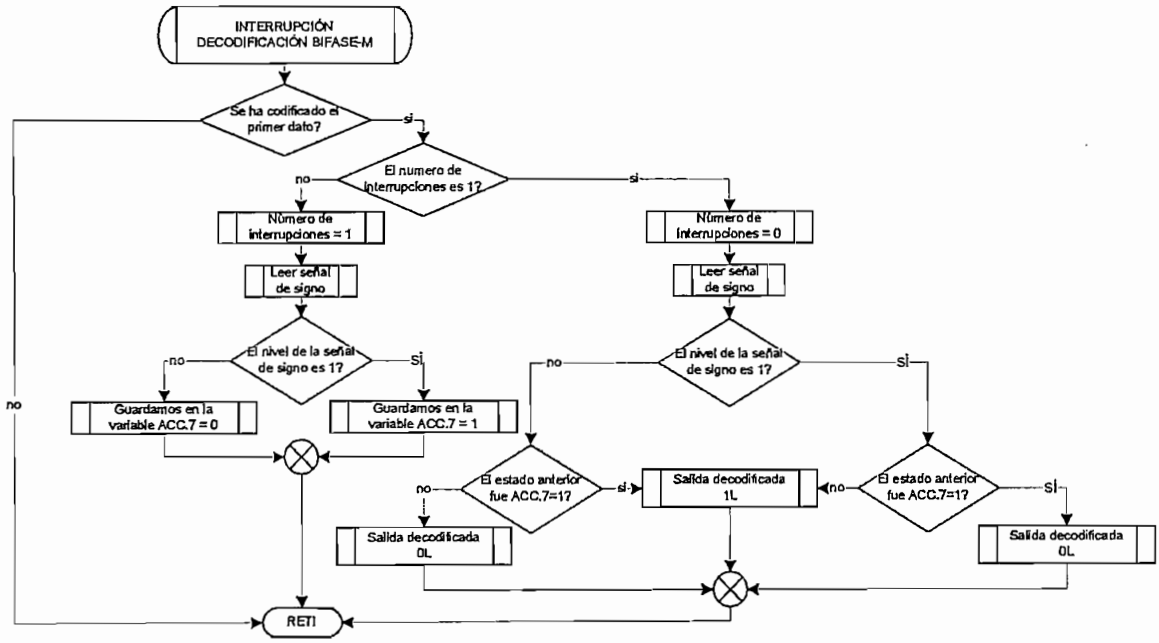


Figura 2.19 Diagrama de flujo de la Decodificación Bifase M.

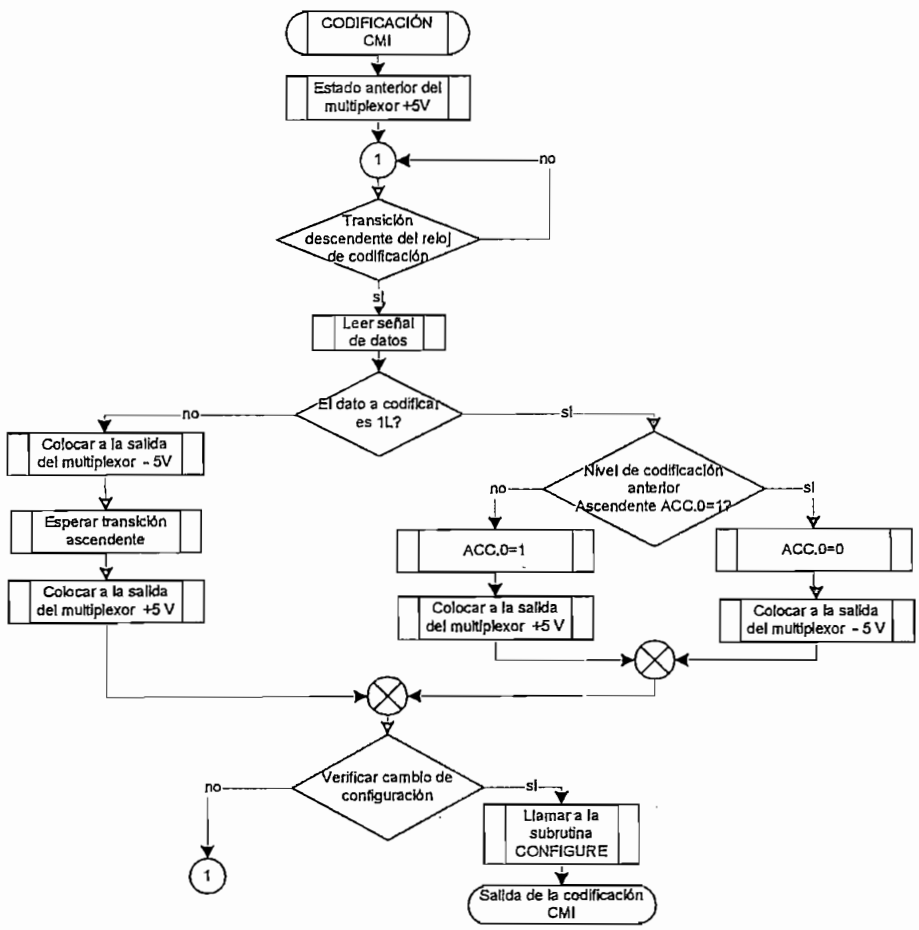


Figura 2.20 Diagrama de flujo de la Codificación CMI.

➢ **Decodificación CMI**

Para la decodificación es necesario tomar en cuenta el estado anterior de la señal codificada, es por esto que dentro del algoritmo de decodificación se toma en cuenta el número de interrupciones que se han generado, para según esto ir aplicando el algoritmo expuesto en el diagrama de flujo de la figura 2.21.

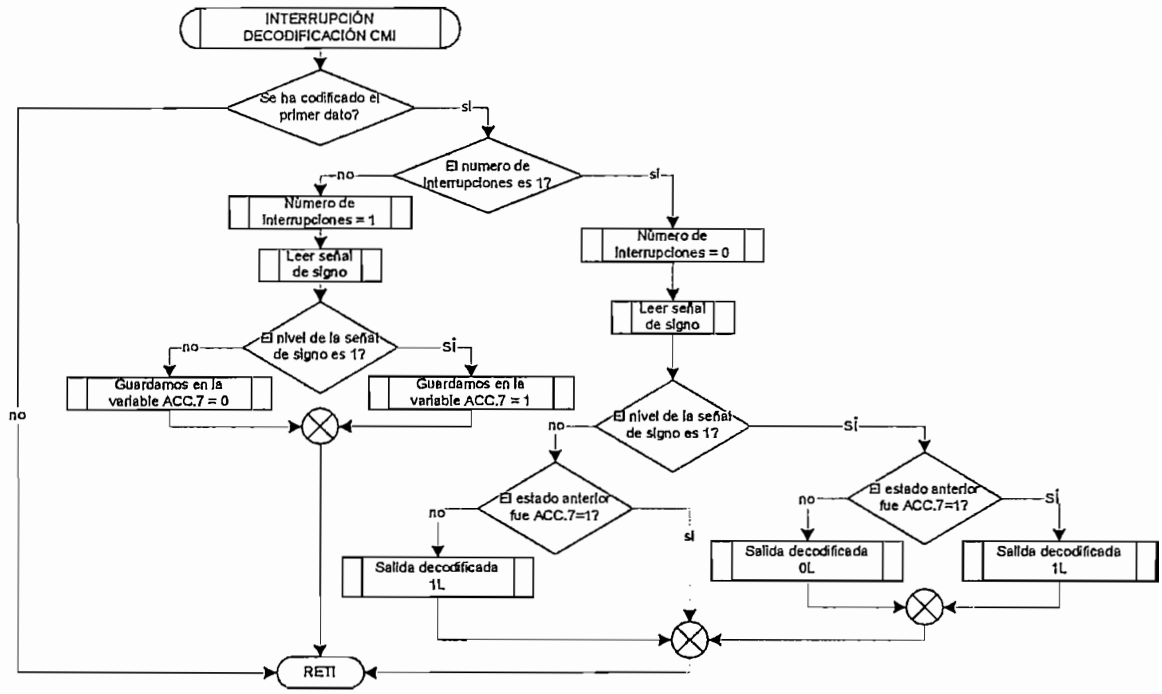


Figura 2.21 Diagrama de flujo de la Decodificación CMI.

2.2.6.7 Código Miller

➢ **Codificación Miller**

En la codificación es necesario tomar en cuenta el número de ceros consecutivos que se tiene en la señal de datos para cumplir con la regla que presenta este tipo de código, es por esto que en el algoritmo se anticipa el conteo del número de ceros, esto se hace guardando el estado anterior. En el diagrama de flujo de la figura 2.22 se muestra el algoritmo de codificación utilizado para la generación de la señal codificada.

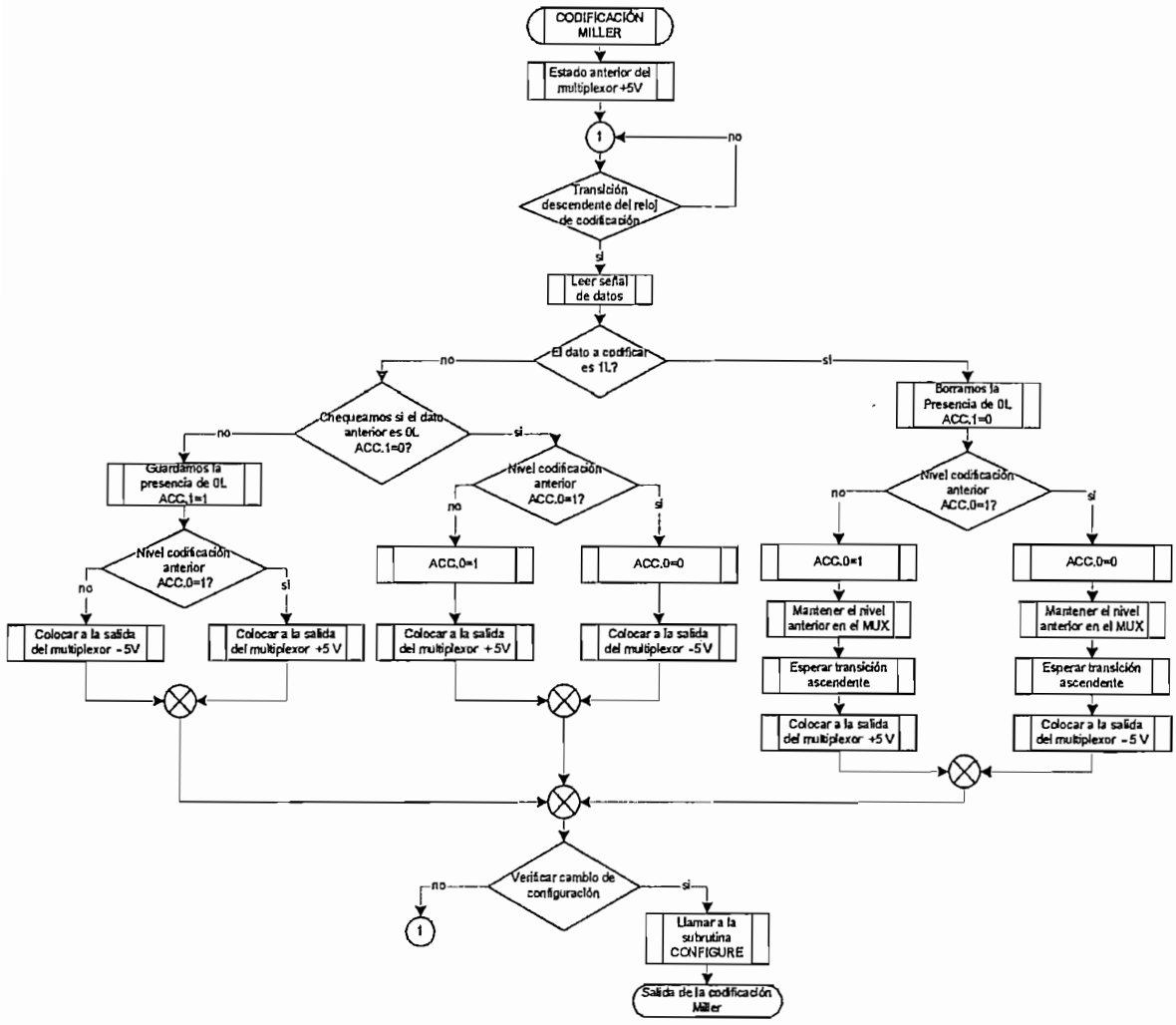


Figura 2.22 Diagrama de flujo de la Codificación Miller.

➢ Decodificación Miller

La decodificación Miller implica tomar en cuenta el estado anterior, pues para obtener a la salida un 0L basta con verificar si existe o no una transición a medio periodo de bit, de lo contrario se trata de un cero lógico. Este proceso de muestreo de la señal se muestra en el diagrama de la figura 2.23.

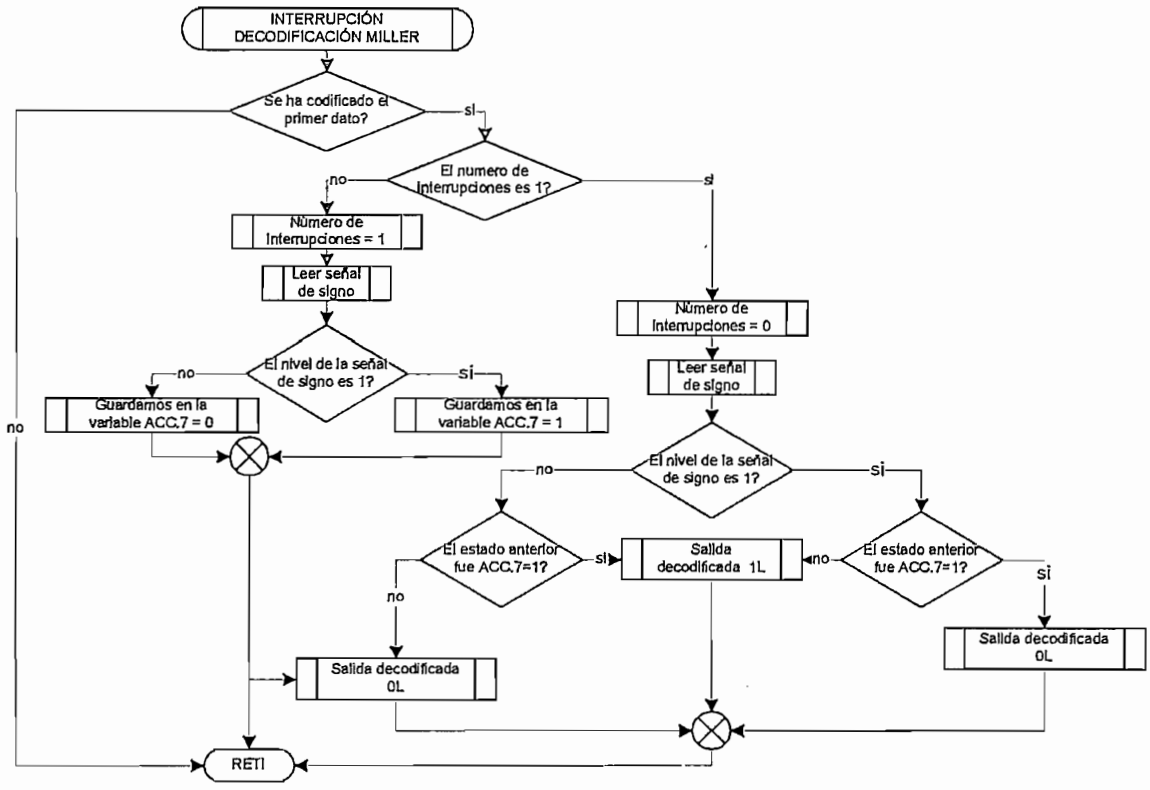


Figura 2.23 Diagrama de flujo de la Decodificación Miller.

2.2.6.8 Código MLT3

- **Codificación MLT3**

Para la codificación MLT3 lo que se toma en cuenta en la entrada es el nivel cero lógico, pues de existir este nivel lo que se hace es mantener el estado anterior, para lo cual no implica ningún procesamiento; lo contrario sucede si se tiene un nivel uno lógico, para este caso lo que se debe tener a la salida del codificador es una señal que varíe en los niveles de +5V, 0V, -5V, 0 V, +5V y así sucesivamente. En el diagrama de flujo de la figura 2.24 se muestra el algoritmo de codificación.

- **Decodificación MLT3**

Para decodificar es necesario tomar en cuenta el estado anterior, pues si se mantiene el nivel anterior se trata de un 0L, esto se lo puede reconocer utilizando

la señal de magnitud de la cual es necesario guardar un estado inicial. En la figura 2.25 se muestra el diagrama de flujo de la decodificación.

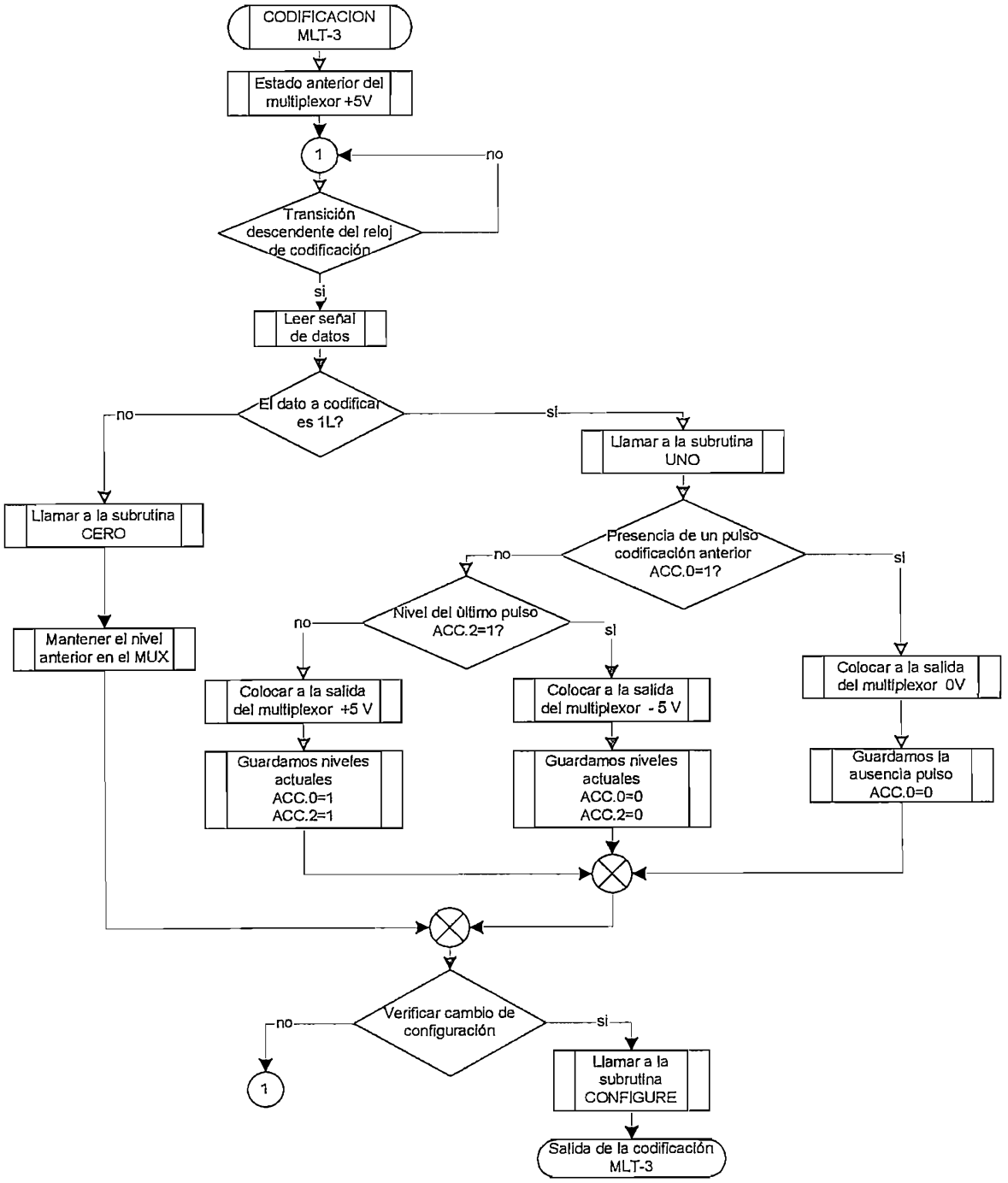


Figura 2.24 Diagrama de flujo de la Codificación MLT-3.

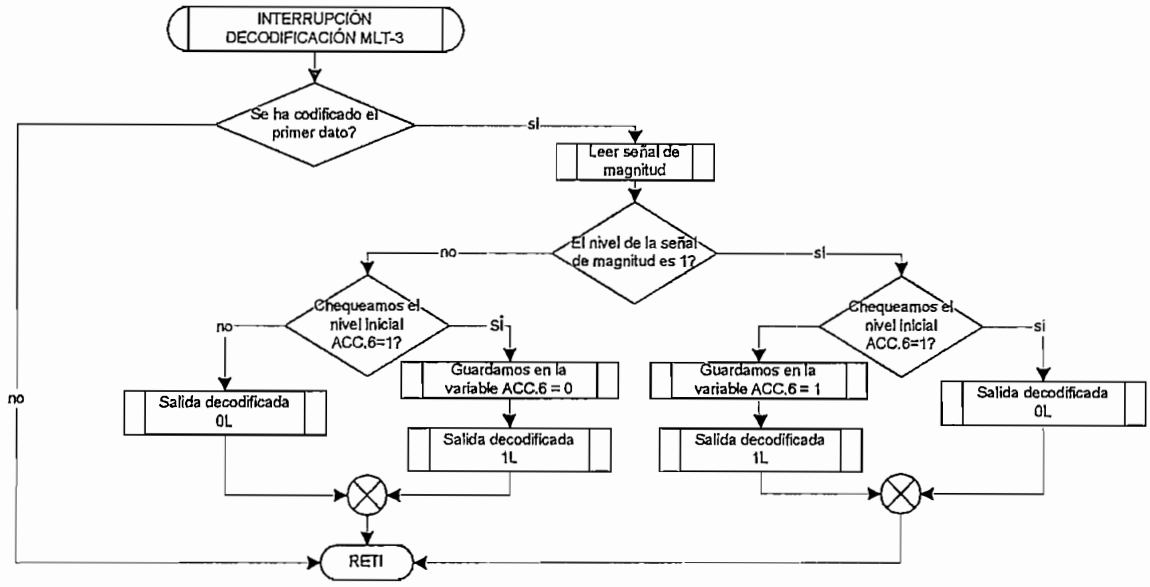


Figura 2.25 Diagrama de flujo de la Decodificación MLT-3.

2.2.6.9 Código B3ZS

➢ **Codificación B3ZS**

El diagrama de flujo de la figura 2.26 muestra el algoritmo que permitirá realizar la codificación B3ZS, para lo cual es necesario procesar los tres primeros bits que lleguen al codificador con la finalidad de verificar si se tienen tres ceros consecutivos, de cumplirse la existencia de tres ceros consecutivos se debe producir la violación a la regla de AMI y modificar sus niveles por la secuencia B0V o 00V según sea el caso. Es por esto que para esta codificación se tiene un retardo de 3 bits en el procesamiento de la señal.

El algoritmo para generar la codificación HDB3 es similar a la codificación B3ZS por lo que no se lo va ha realizar, pues lo único que cambia es el número de bits que al inicio hay que procesar, para este caso se debe procesar 4 bits con la finalidad de chequear la existencia de 4 ceros consecutivos, si se tiene este número de ceros consecutivos a la entrada se debe reemplazar por la secuencia B00V o 000V.

➢ Decodificación B3ZS

Al igual que en la codificación, para decodificar la señal codificada es necesario procesar los 3 primeros niveles entrantes, con la finalidad de encontrar la existencia de la violación a la regla de AMI, de ser así la secuencia encontrada es reemplazada por ceros, es por esto que en la decodificación también se tiene un retardo de 3 bits. En la figura 2.27 se muestra el diagrama de flujo del proceso de decodificación.

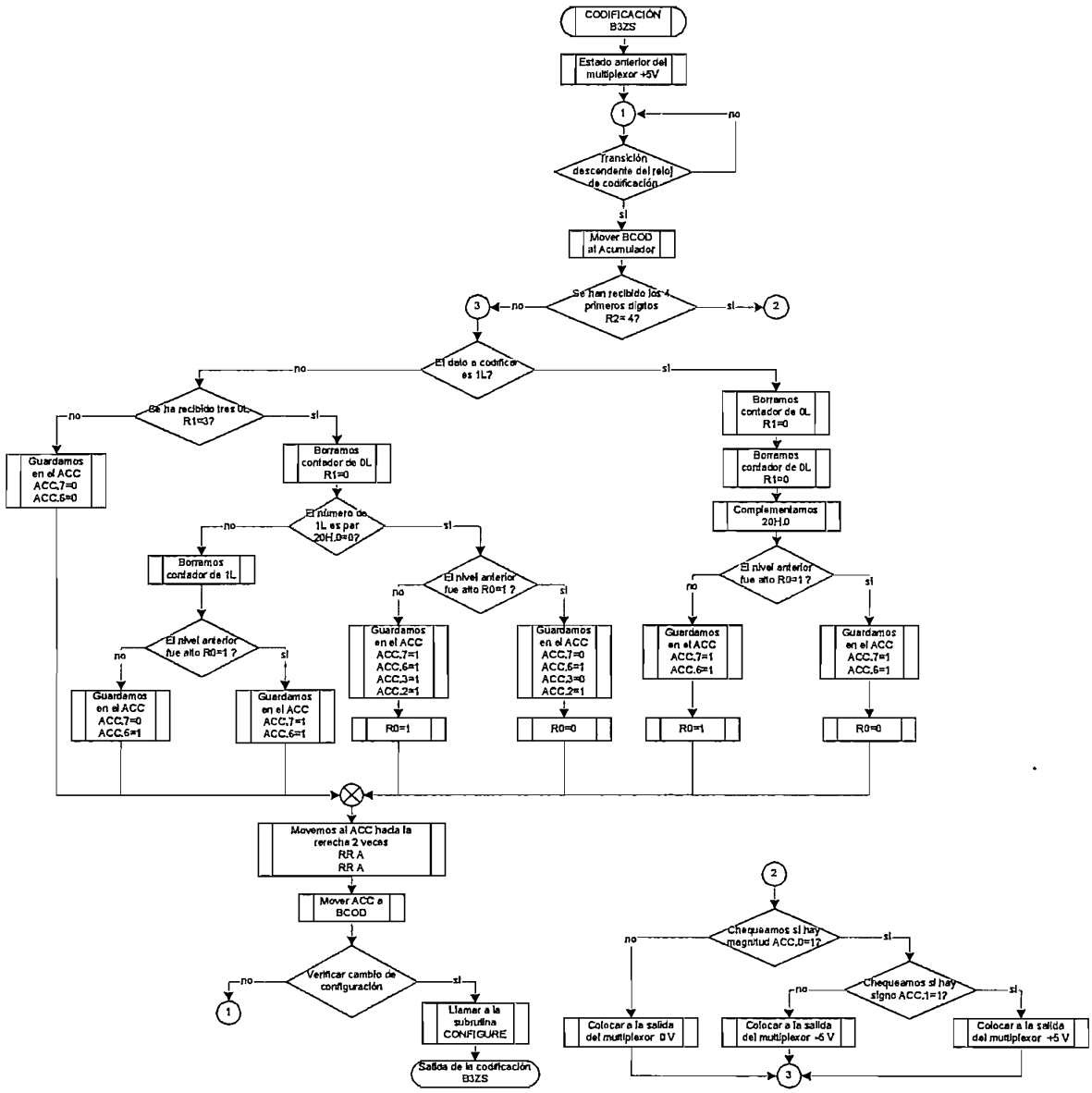


Figura 2.26 Diagrama de flujo de la Codificación B3ZS.

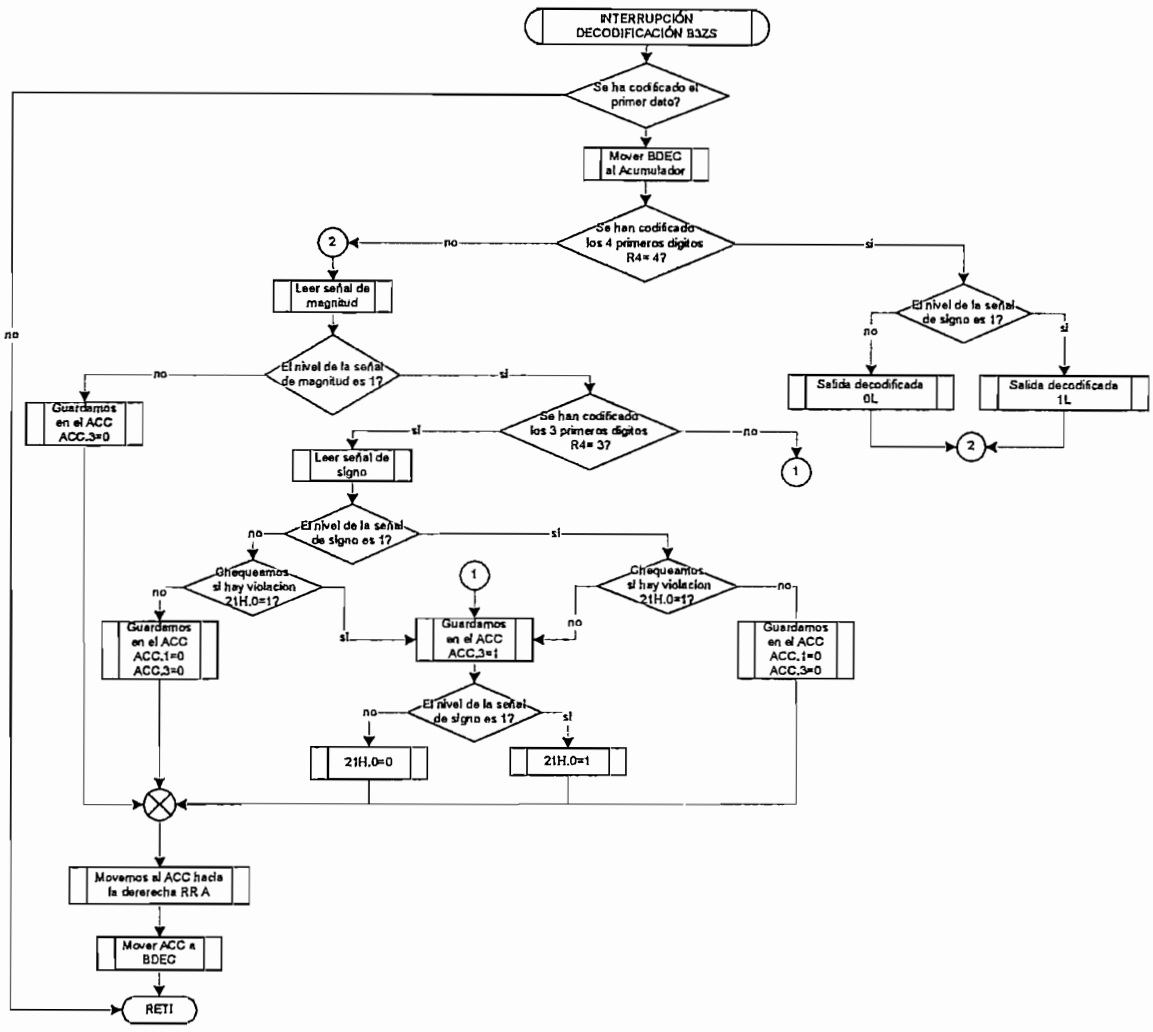


Figura 2.27 Diagrama de flujo de la Decodificación B3ZS.

El proceso de decodificación HDB3 es semejante al proceso de decodificación B3ZS por lo que no se lo va a realizar, pues lo que cambia es el chequeo del número de niveles entrantes al decodificador, que para HDB3 son 4 y se sigue el mismo proceso de verificar la existencia de violación a la regla de AMI.

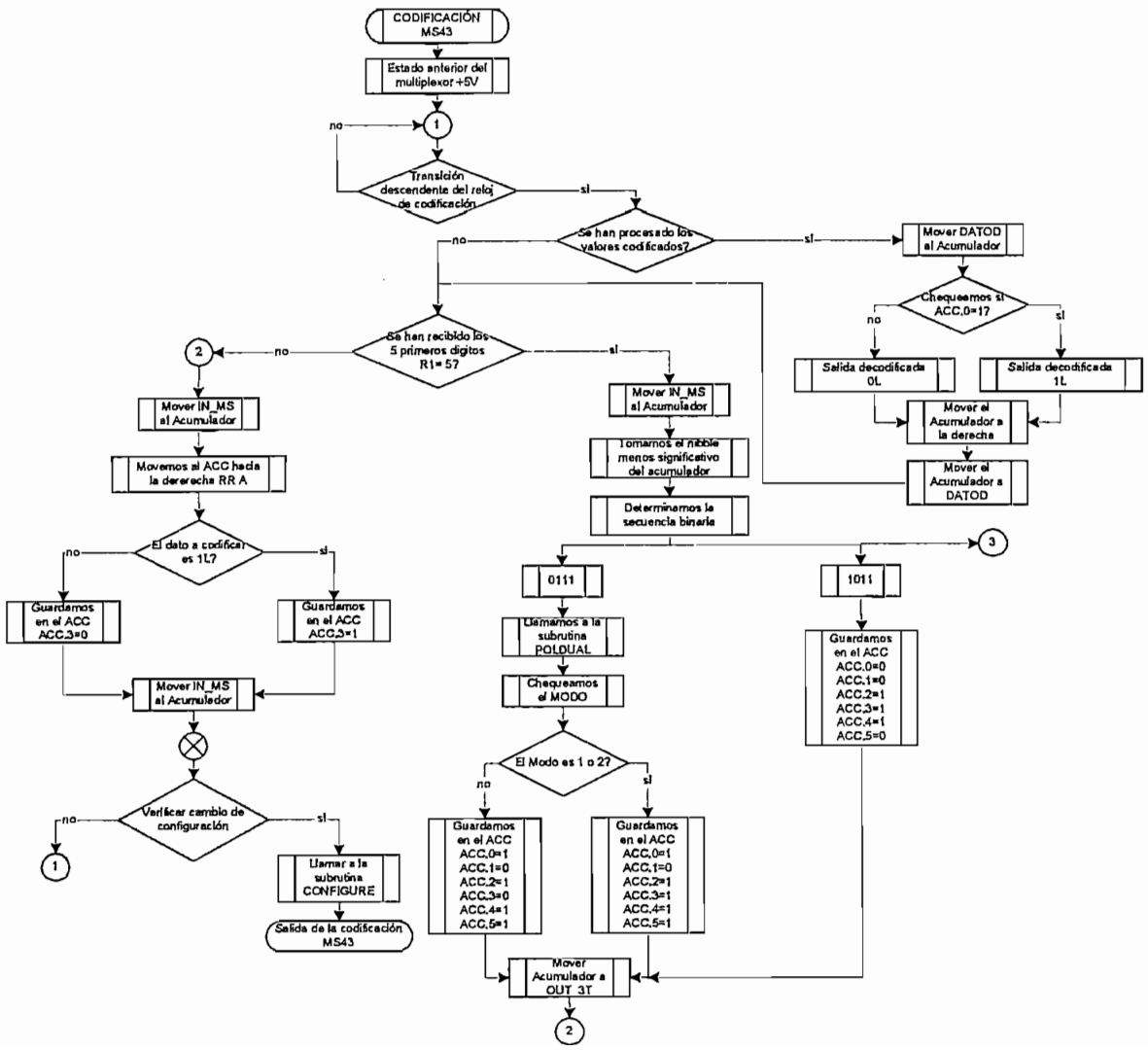
2.2.6.10 Código MS43

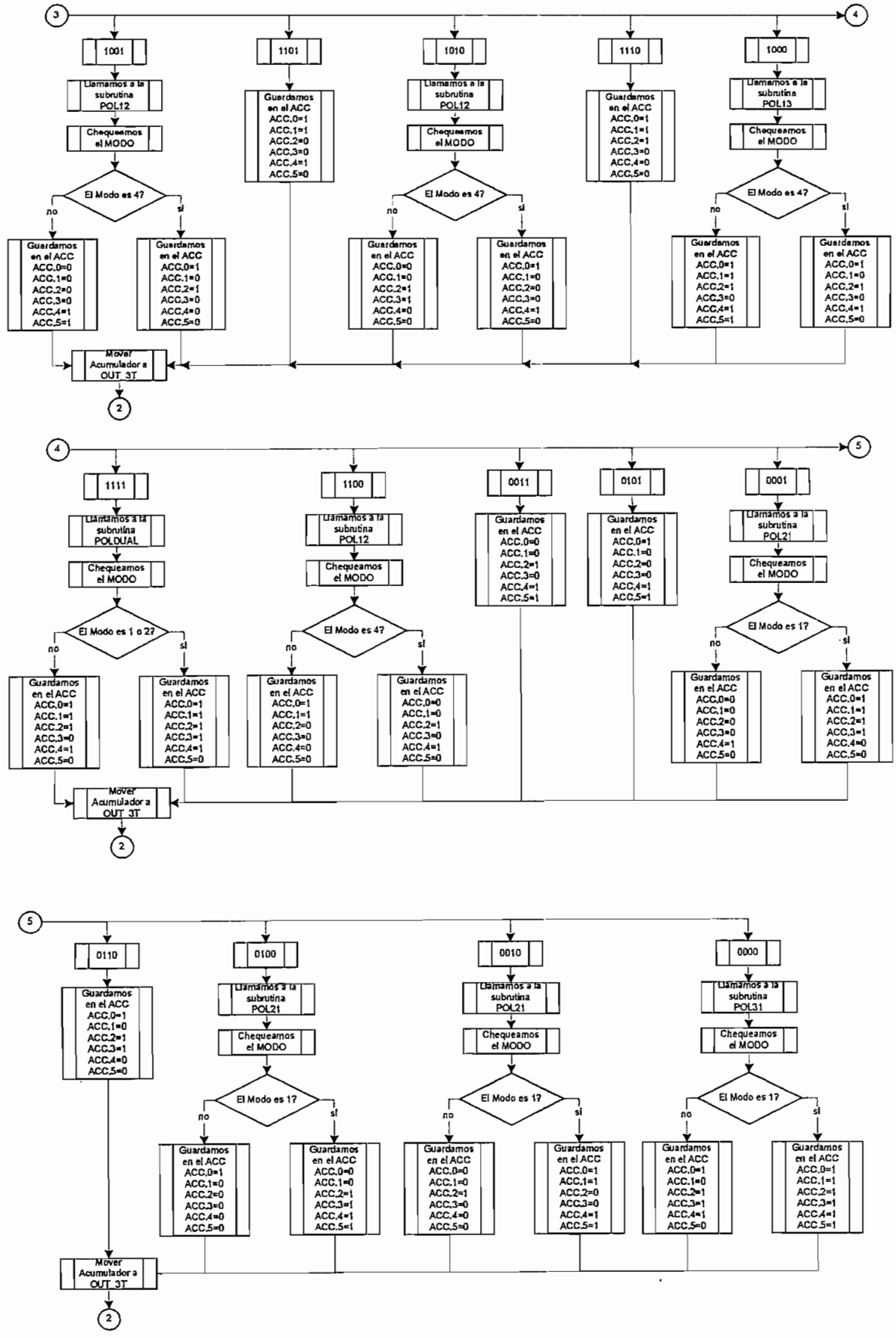
- **Codificación MS43**

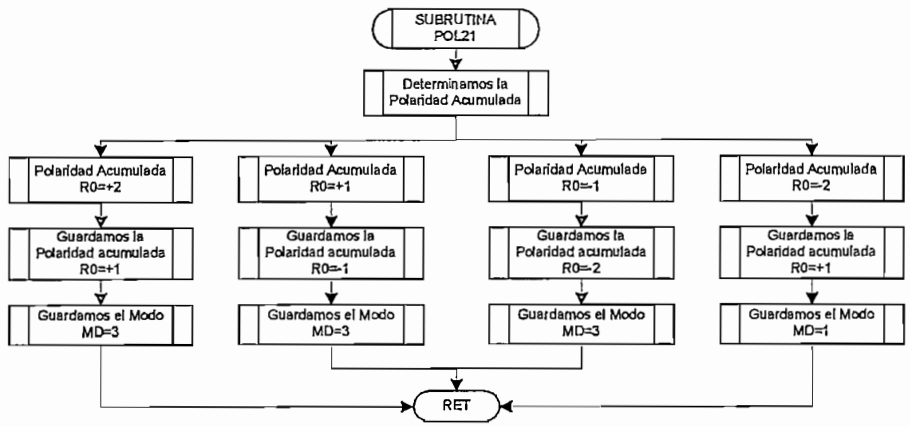
El diagrama de flujo de la figura 2.28 muestra el algoritmo de codificación MS43, al aplicar este algoritmo se tiene un retardo de 4 bits, este retardo de

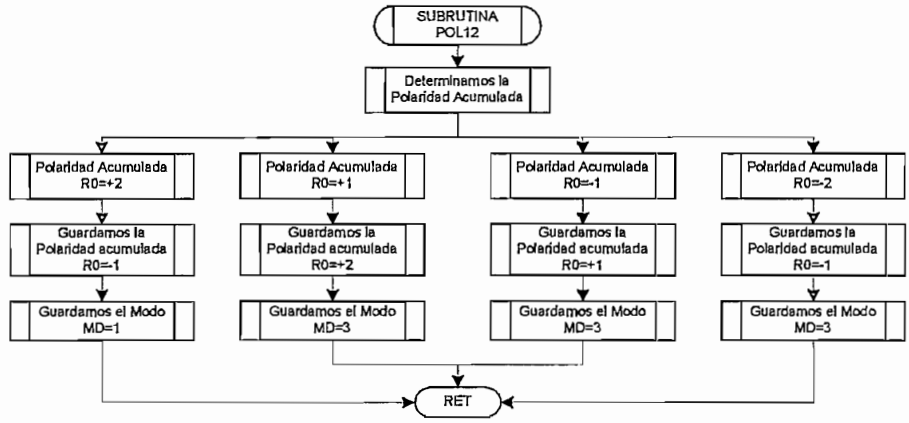
procesamiento se debe a que para codificar la información entrante, se debe tomar grupos de 4 dígitos binarios para así reemplazarlos por tres dígitos ternarios. El algoritmo de codificación debe comparar la información entrante con los existentes en la tabla guardada en el microcontrolador y a su vez tomar en cuenta la disparidad acumulada para la selección de los tres dígitos ternarios que reemplazarán a estos cuatro dígitos binarios.

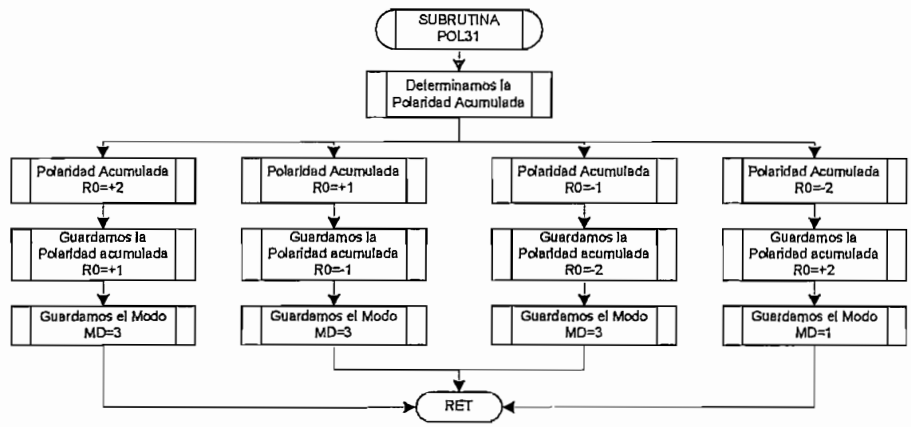
La particularidad de esta codificación es que necesita dos señales de reloj, uno de los relojes debe ser los ¾ de la frecuencia del otro reloj, los dos relojes son importantes en la codificación pues con el un reloj permite capturar los datos entrantes, mientras que con el reloj de ¾ de la frecuencia se va a procesar la salida de los datos del codificador.

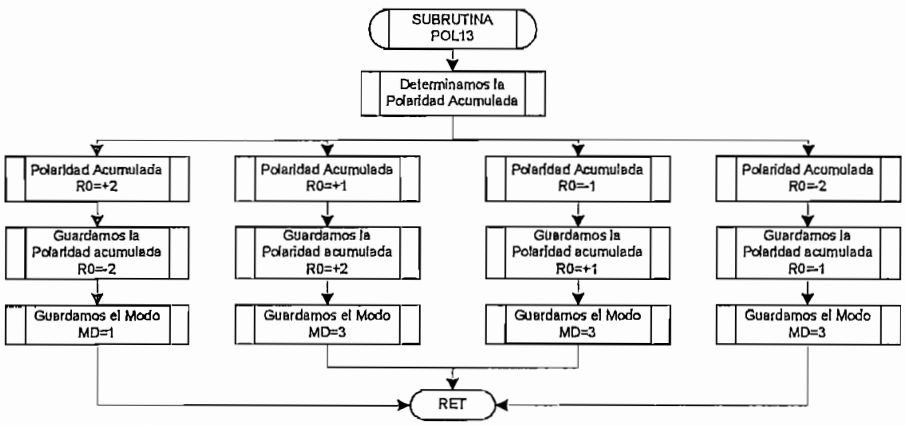



3
4
1001
Llamamos a la subrutina POL12
Chequeamos el MODO
El Modo es 4?
no
si
Guardamos en el ACC ACC,0=0 ACC,1=0 ACC,2=0 ACC,3=0 ACC,4=1 ACC,5=1
Guardamos en el ACC ACC,0=1 ACC,1=0 ACC,2=1 ACC,3=0 ACC,4=0 ACC,5=0
1101
Guardamos en el ACC ACC,0=1 ACC,1=1 ACC,2=0 ACC,3=0 ACC,4=1 ACC,5=0
1010
Llamamos a la subrutina POL12
Chequeamos el MODO
El Modo es 4?
Guardamos en el ACC ACC,0=0 ACC,1=0 ACC,2=1 ACC,3=1 ACC,4=0 ACC,5=0
Guardamos en el ACC ACC,0=1 ACC,1=0 ACC,2=0 ACC,3=0 ACC,4=1 ACC,5=0
1110
Guardamos en el ACC ACC,0=1 ACC,1=1 ACC,2=1 ACC,3=0 ACC,4=0 ACC,5=0
1000
Llamamos a la subrutina POL13
Chequeamos el MODO
El Modo es 4?
Guardamos en el ACC ACC,0=1 ACC,1=1 ACC,2=0 ACC,3=0 ACC,4=0 ACC,5=1
Guardamos en el ACC ACC,0=1 ACC,1=0 ACC,2=1 ACC,3=0 ACC,4=1 ACC,5=0
Mover Acumulador a OUT_3T
2
4
5
1111
Llamamos a la subrutina POLDUAL
Chequeamos el MODO
El Modo es 1 o 2?
Guardamos en el ACC ACC,0=1 ACC,1=1 ACC,2=1 ACC,3=0 ACC,4=1 ACC,5=0
Guardamos en el ACC ACC,0=1 ACC,1=1 ACC,2=1 ACC,3=1 ACC,4=1 ACC,5=0
1100
Llamamos a la subrutina POL12
Chequeamos el MODO
El Modo es 4?
Guardamos en el ACC ACC,0=1 ACC,1=1 ACC,2=0 ACC,3=0 ACC,4=0 ACC,5=0
Guardamos en el ACC ACC,0=0 ACC,1=0 ACC,2=1 ACC,3=0 ACC,4=1 ACC,5=0
0011
Guardamos en el ACC ACC,0=0 ACC,1=0 ACC,2=1 ACC,3=0 ACC,4=1 ACC,5=1
0101
Guardamos en el ACC ACC,0=1 ACC,1=0 ACC,2=0 ACC,3=0 ACC,4=1 ACC,5=1
0001
Llamamos a la subrutina POL21
Chequeamos el MODO
El Modo es 1?
Guardamos en el ACC ACC,0=0 ACC,1=0 ACC,2=0 ACC,3=0 ACC,4=1 ACC,5=0
Guardamos en el ACC ACC,0=1 ACC,1=1 ACC,2=1 ACC,3=1 ACC,4=0 ACC,5=0
Mover Acumulador a OUT_3T
2
5
0110
Guardamos en el ACC ACC,0=1 ACC,1=0 ACC,2=1 ACC,3=1 ACC,4=0 ACC,5=0
0100
Llamamos a la subrutina POL21
Chequeamos el MODO
El Modo es 1?
Guardamos en el ACC ACC,0=1 ACC,1=0 ACC,2=0 ACC,3=0 ACC,4=0 ACC,5=0
Guardamos en el ACC ACC,0=0 ACC,1=0 ACC,2=1 ACC,3=1 ACC,4=1 ACC,5=1
0010
Llamamos a la subrutina POL21
Chequeamos el MODO
El Modo es 1?
Guardamos en el ACC ACC,0=0 ACC,1=0 ACC,2=1 ACC,3=0 ACC,4=0 ACC,5=0
Guardamos en el ACC ACC,0=1 ACC,1=1 ACC,2=0 ACC,3=0 ACC,4=1 ACC,5=1
0000
Llamamos a la subrutina POL31
Chequeamos el MODO
El Modo es 1?
Guardamos en el ACC ACC,0=1 ACC,1=0 ACC,2=1 ACC,3=1 ACC,4=1 ACC,5=0
Guardamos en el ACC ACC,0=1 ACC,1=1 ACC,2=1 ACC,3=1 ACC,4=1 ACC,5=1
Mover Acumulador a OUT_3T
2


SUBRUTINA
POL21
Determinamos la
Polaridad Acumulada
Polaridad Acumulada
R0=+2
Polaridad Acumulada
R0=+1
Polaridad Acumulada
R0=-1
Polaridad Acumulada
R0=-2
Guardamos la
Polaridad acumulada
R0=+1
Guardamos la
Polaridad acumulada
R0=-1
Guardamos la
Polaridad acumulada
R0=-2
Guardamos la
Polaridad acumulada
R0=+1
Guardamos el Modo
MD=3
Guardamos el Modo
MD=3
Guardamos el Modo
MD=3
Guardamos el Modo
MD=1
RET


SUBRUTINA
POL12
Determinamos la
Polaridad Acumulada
Polaridad Acumulada
R0=+2
Polaridad Acumulada
R0=+1
Polaridad Acumulada
R0=-1
Polaridad Acumulada
R0=-2
Guardamos la
Polaridad acumulada
R0=-1
Guardamos la
Polaridad acumulada
R0=+2
Guardamos la
Polaridad acumulada
R0=+1
Guardamos la
Polaridad acumulada
R0=-1
Guardamos el Modo
MD=1
Guardamos el Modo
MD=3
Guardamos el Modo
MD=3
Guardamos el Modo
MD=3
RET


SUBRUTINA
POL31
Determinamos la
Polaridad Acumulada
Polaridad Acumulada
R0=+2
Polaridad Acumulada
R0=+1
Polaridad Acumulada
R0=-1
Polaridad Acumulada
R0=-2
Guardamos la
Polaridad acumulada
R0=+1
Guardamos la
Polaridad acumulada
R0=-1
Guardamos la
Polaridad acumulada
R0=-2
Guardamos la
Polaridad acumulada
R0=+2
Guardamos el Modo
MD=3
Guardamos el Modo
MD=3
Guardamos el Modo
MD=3
Guardamos el Modo
MD=1
RET


SUBRUTINA
POL13
Determinamos la
Polaridad Acumulada
Polaridad Acumulada
R0=+2
Polaridad Acumulada
R0=+1
Polaridad Acumulada
R0=-1
Polaridad Acumulada
R0=-2
Guardamos la
Polaridad acumulada
R0=-2
Guardamos la
Polaridad acumulada
R0=+2
Guardamos la
Polaridad acumulada
R0=+1
Guardamos la
Polaridad acumulada
R0=-1
Guardamos el Modo
MD=1
Guardamos el Modo
MD=3
Guardamos el Modo
MD=3
Guardamos el Modo
MD=3
RET

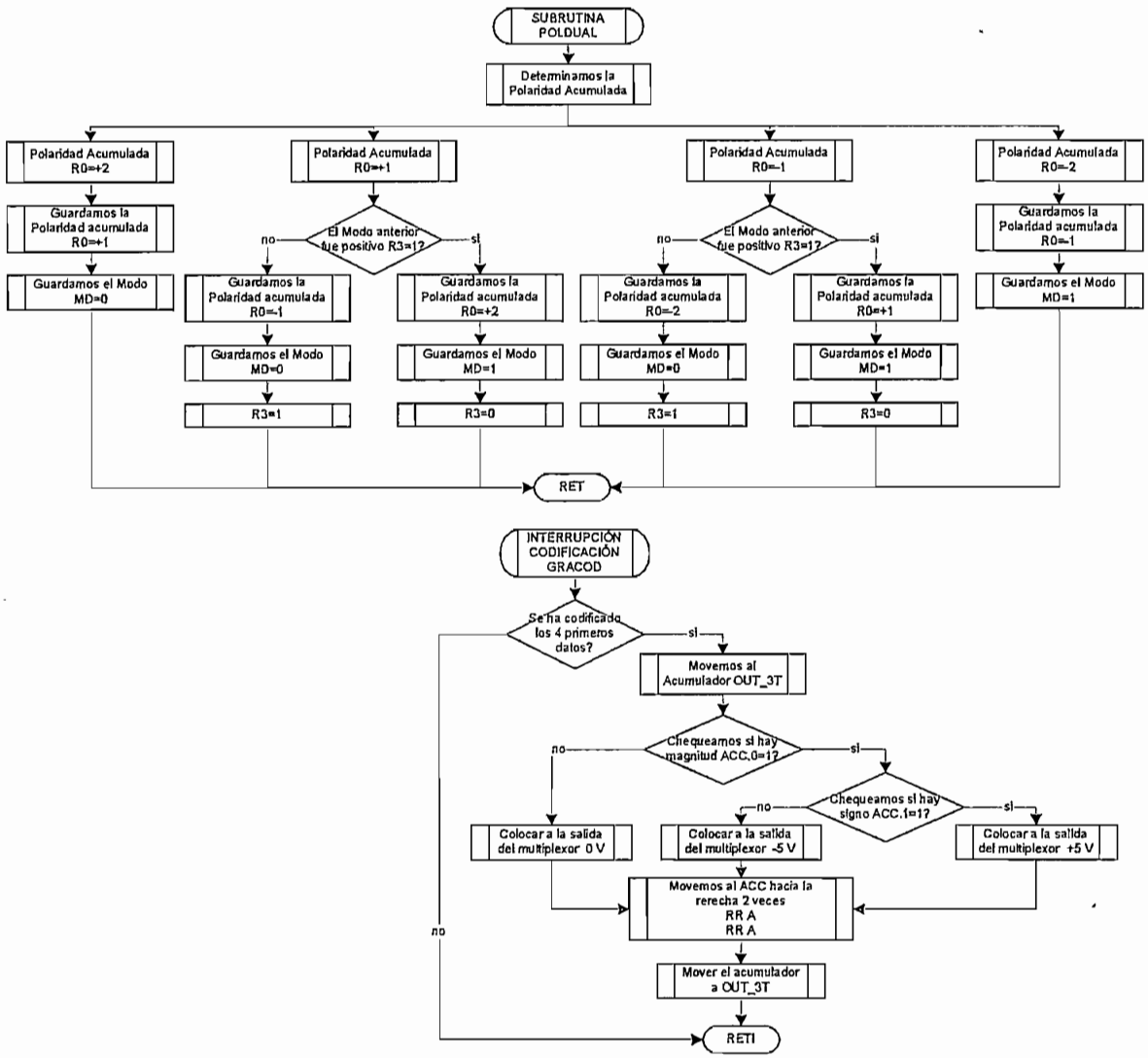


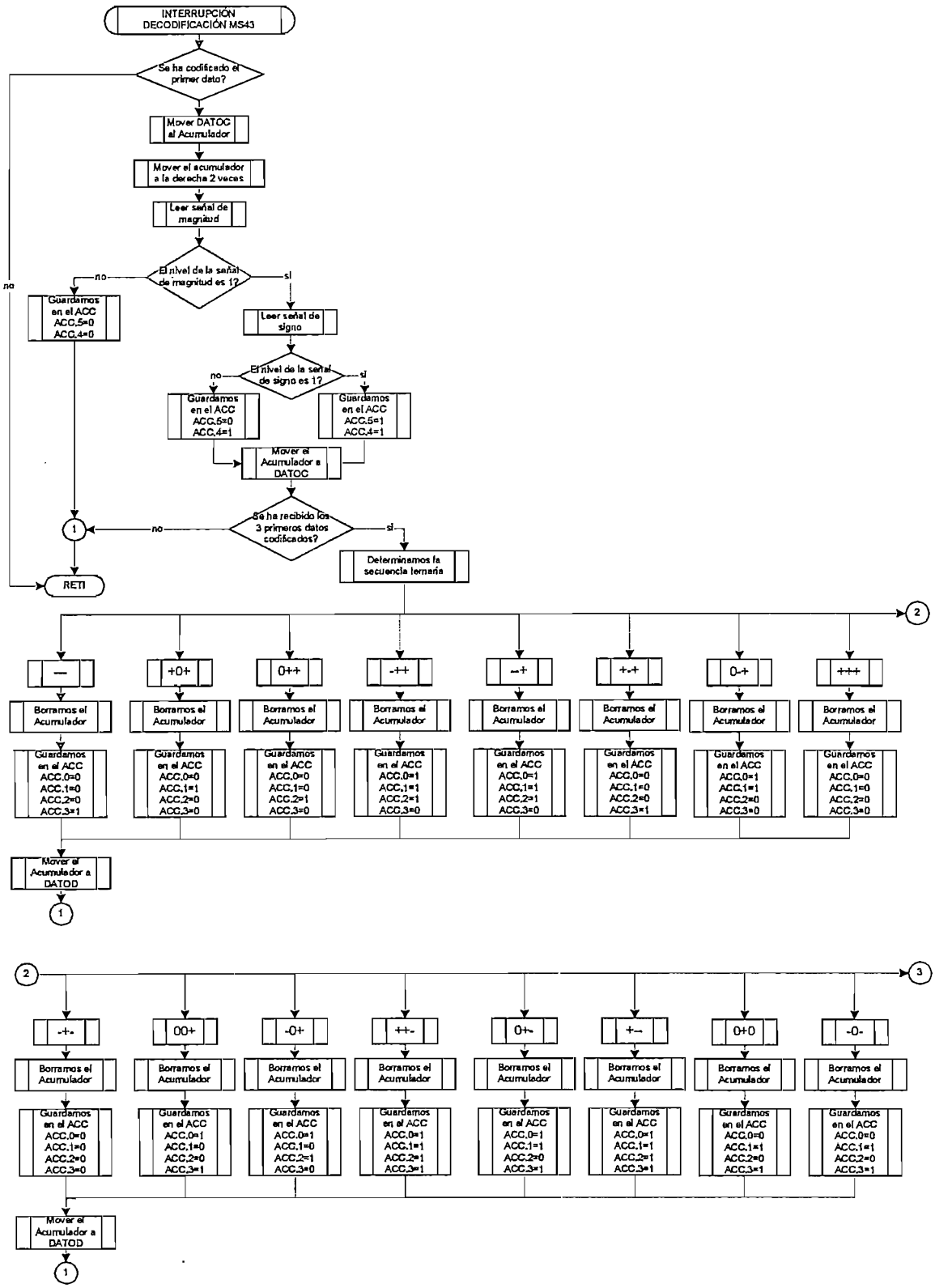
Figura 2.28 Diagrama de flujo de la Codificación MS43.

La codificación 4B3T tiene un proceso similar a la codificación, pues lo único que cambia es la tabla de codificación y las disparidades, por lo que no se expondrá el diagrama de flujo de codificación 4B3T.

➢ Decodificación MS43

Para la decodificación se debe tomar 3 dígitos ternarios de la señal codificada los mismos que serán reemplazados por 4 dígitos binarios, para lo cual estos tres dígitos ternarios son comparados con la tabla previamente guardada en el microprocesador. Al igual que la etapa de codificación, la decodificación también

necesita de los dos relojes anteriormente descritos. El algoritmo se expone en el diagrama de flujo de la figura 2.29.



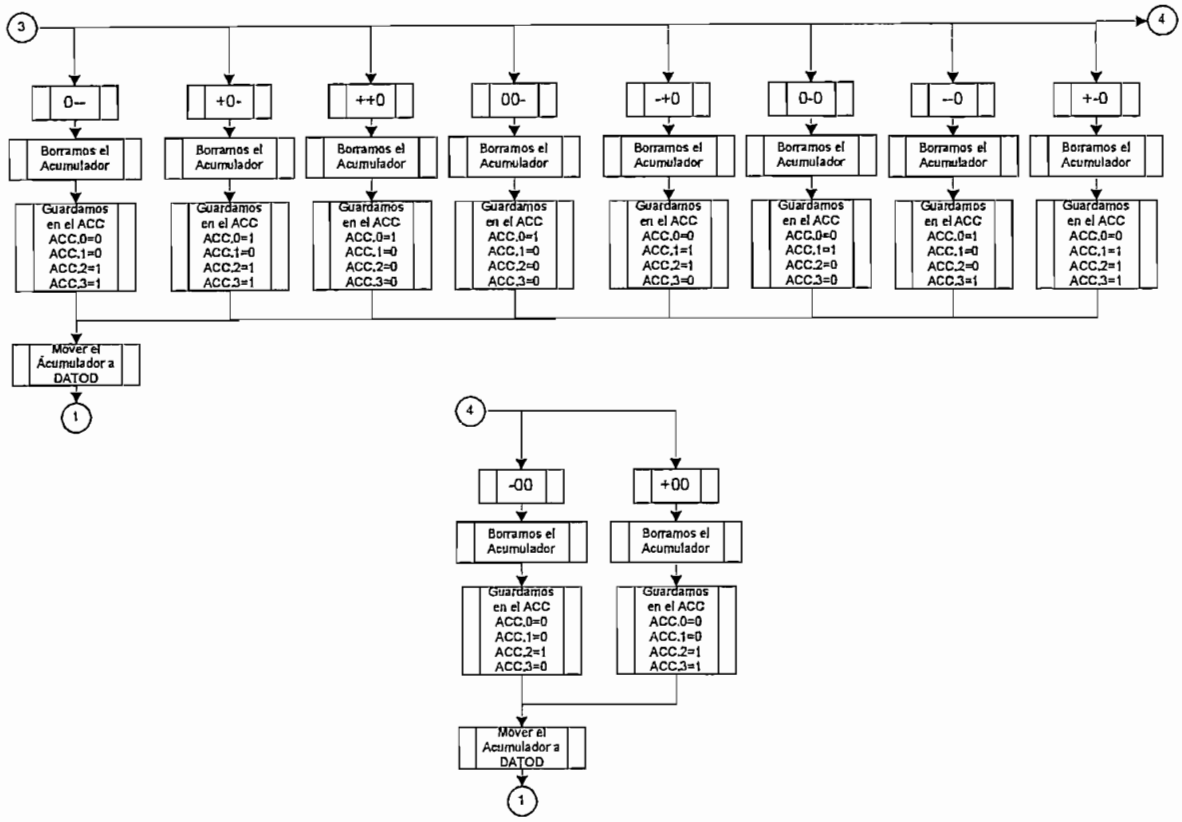


Figura 2.29 Diagrama de flujo de la Decodificación MS43.

El proceso de decodificación 4B3T es similar al proceso de decodificación MS43, pues tiene el mismo principio de comparar los datos que entrar al decodificador con los existentes en la tabla, por lo que no se va a presentar dicho algoritmo.

2.2.6.11 Subrutinas

Una vez concluida la presentación de los algoritmos de codificación y decodificación se presentan las subrutinas utilizadas en el programa, las cuales permiten mantener la comunicación con el computador, estas subrutinas son:

➢ **Subrutina TEMPO:** Permite el envío al computador de un uno lógico, siempre y cuando se produzca la interrupción debido al desbordamiento del *timer 0* además permite parar el envío de datos al computador. En la figura 2.30 se muestra el diagrama de flujo de esta subrutina.

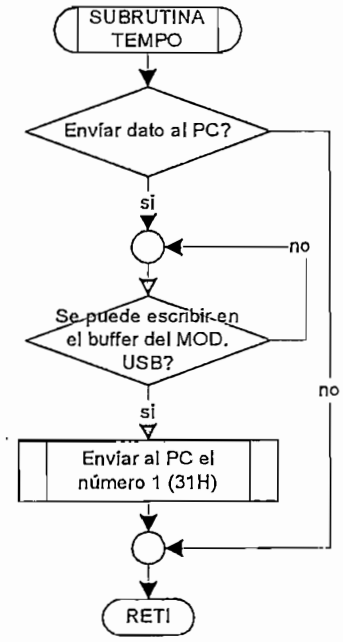


Figura 2.30 Subrutina TEMPO

Otra de las subrutinas presentes en cada uno de los algoritmos es la **subrutina CONFIGURE**, la cual se encarga de almacenar los valores de velocidad y de código a más de permitir configurar los relojes. En el diagrama de flujo de la figura 2.31 se muestra esta subrutina.

Dentro de la subrutina CONFIGURE se tiene otra subrutina llamada **TRANSMITIR**, esta subrutina realiza la comunicación serial con el PIC 16F870 para configurar la frecuencia de la señal de reloj de codificación y decodificación. El diagrama de flujo de la subrutina TRANSMITIR se muestra en la figura 2.32, esta subrutina envía los datos de configuración a una velocidad de 9600 bits por segundo.

Una vez explicado los algoritmos utilizados en la generación de los distintos códigos de línea, así como los algoritmos de las comunicaciones vía las interfaces USB y Serial, estos algoritmos son interpretados en mnemónicos, los mismos que serán almacenados en la memoria FLASH del microcontrolador que cumple la función de Unidad Central de Procesos. El diagrama de conexiones de la Unidad Central de Procesos se muestra en la Figura 2.33.

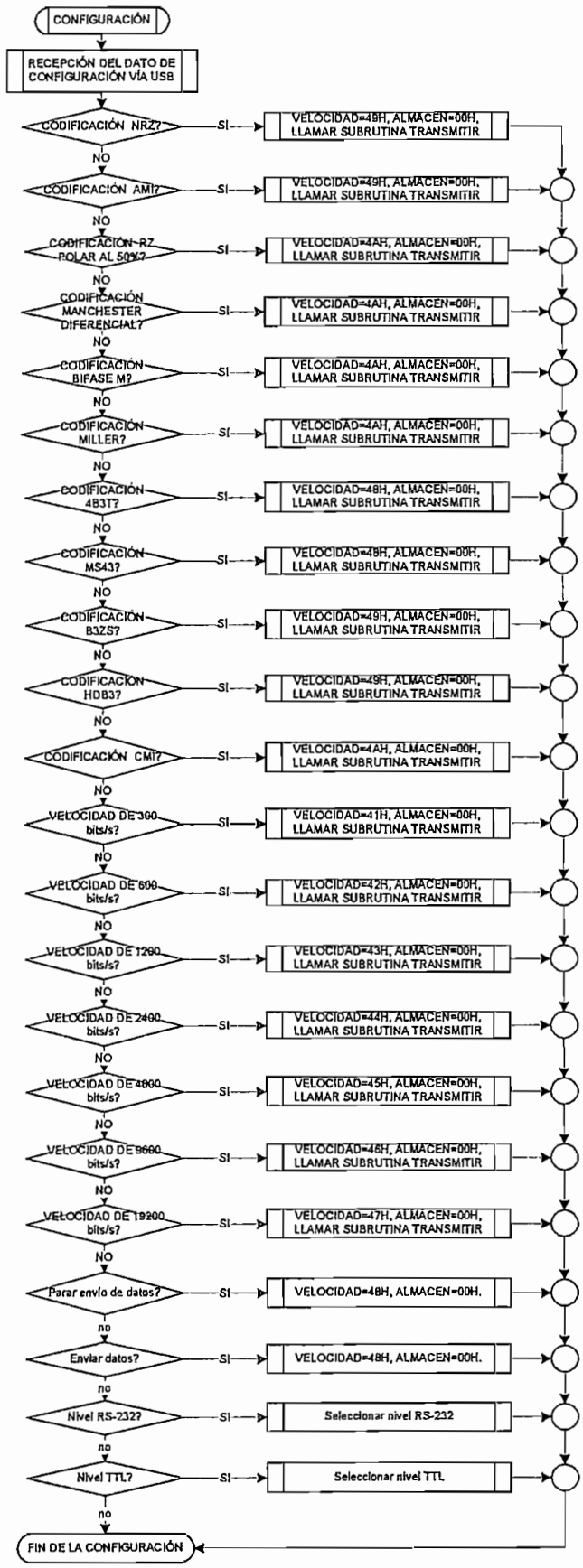


Figura 2.31 Subrutina CONFIGURE

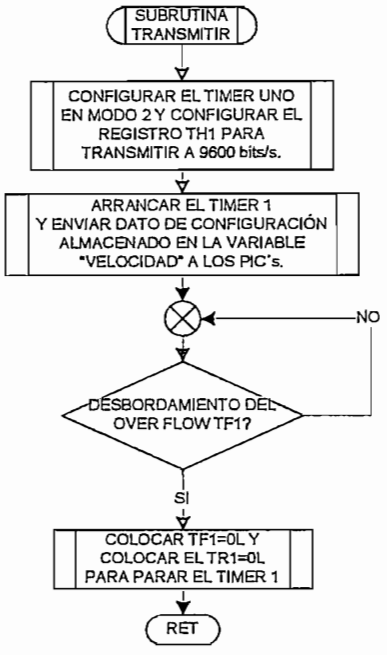


Figura 2.32 Subrutina TRANSMITIR

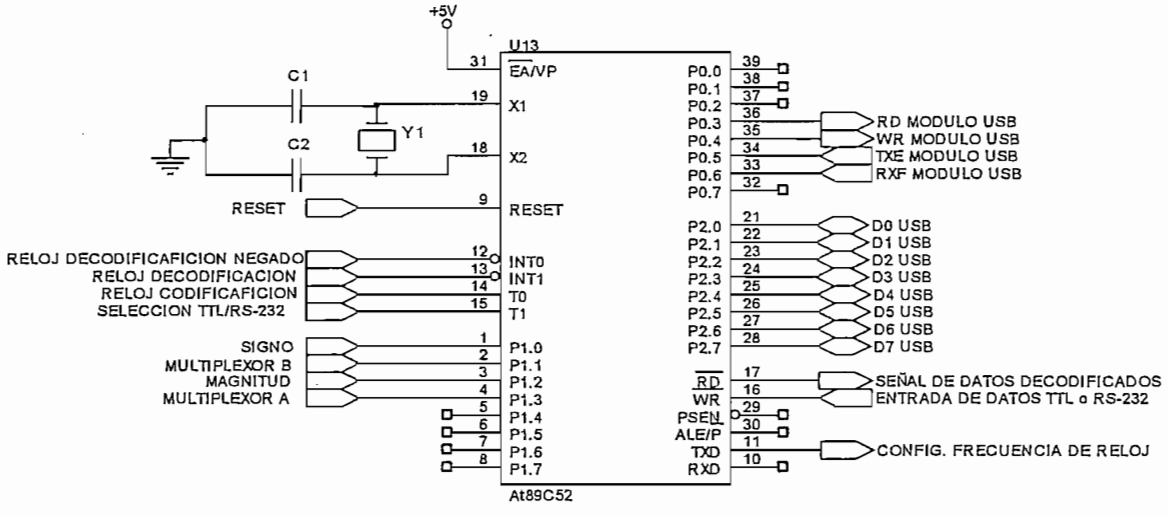


Figura 2.33 Conexiones de la Unidad Central de Procesos del CODEC.

Un aspecto importante del circuito es su polarización, la cual se la hace en base a la energía tomada del bus. Además varios elementos del circuito necesitan una polarización negativa de 5voltios, por lo que para polarizarlos se a utililizado el conversor DC – DC Max765, el cual permitió convertir el voltaje de +5V en un voltaje de –12V y luego mediante circuitería convertirlo en –5V. En la figura 2.34 se muestra la distribución de pines y los elementos que se utilizan para la conversión DC – DC.

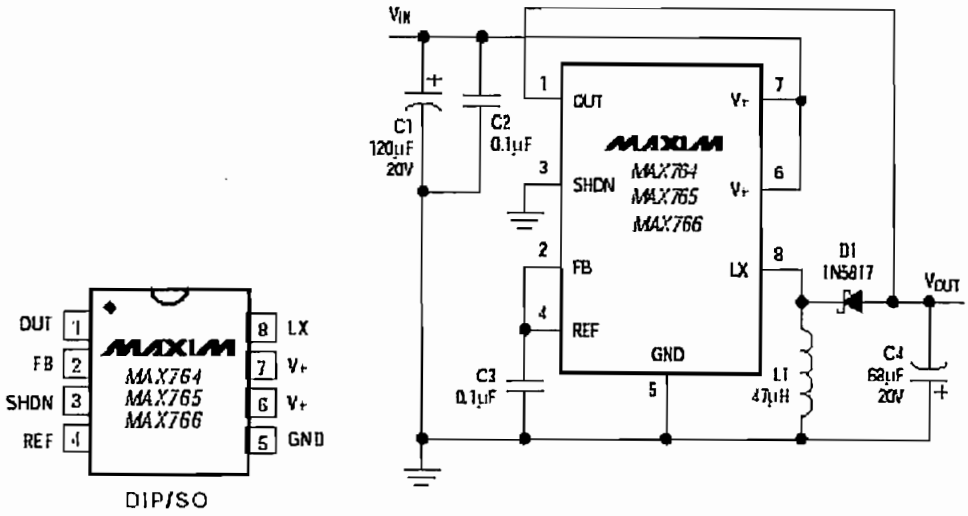
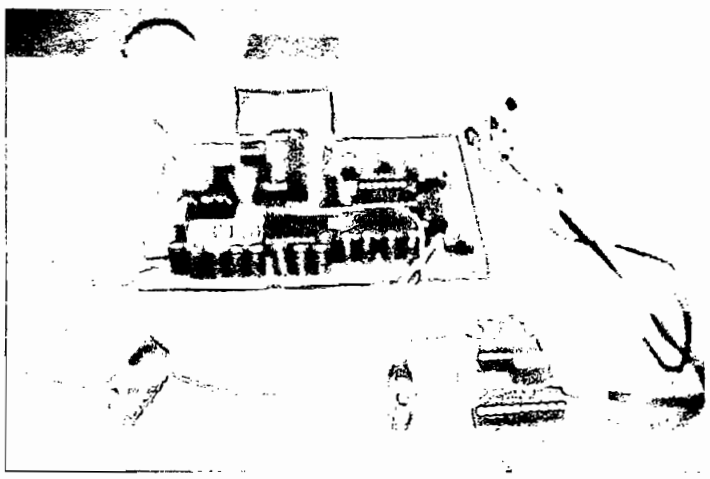


Figura 2.34 Circuito Integrado MAX765[1].

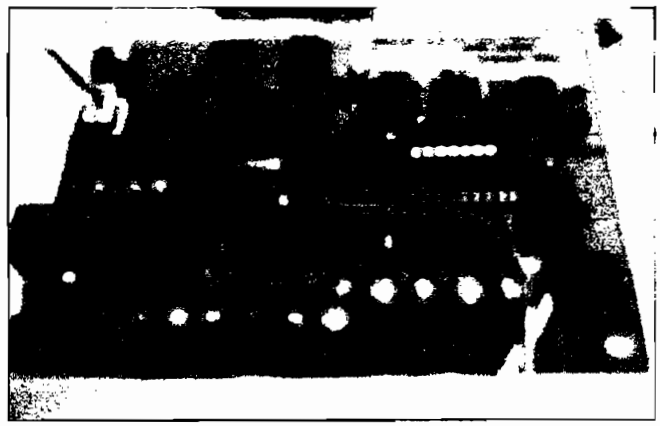
El Diagrama de todas las conexiones para la construcción del CODEC se muestra en el Anexo B.

Además se presentan en la figura 2.34 fotos del prototipo terminado, es decir la parte interna del CODEC.

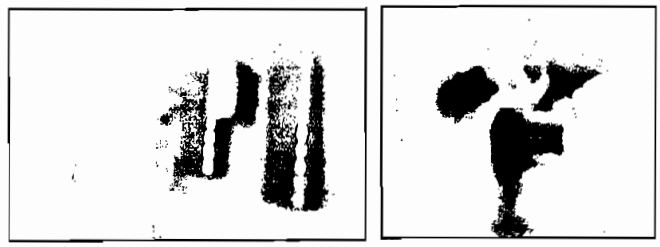


(a)

[1] El circuito y la distribucón de pines tomados de la hoja de datos del MAX765. Ver Anexo B.



(b)



(c)

Figura 2.35 Circuito Armado del CODEC

CAPÍTULO III

DESARROLLO DEL SOFTWARE PARA LA INTERFAZ GRÁFICA

DESARROLLO DEL SOFTWARE PARA LA INTERFAZ GRÁFICA

3.1 DESCRIPCIÓN Y REQUERIMIENTOS DEL *SOFTWARE.*

En el capítulo anterior se ha descrito tanto el diseño y construcción del hardware como la justificación de la utilización de cada uno de los elementos empleados, por tanto en este capítulo se expondrá una descripción del *software* que se desarrolló para la configuración del CODEC (codificador/decodificador).

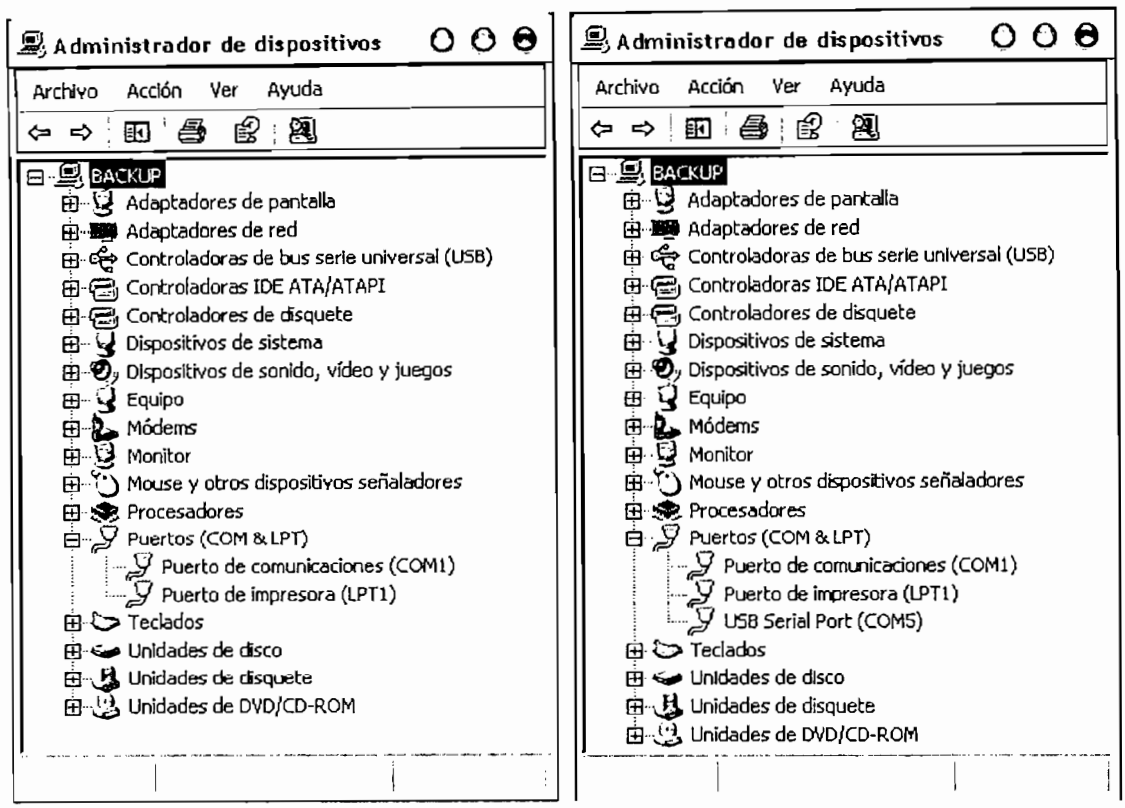
Con la finalidad de presentar una interfaz gráfica amigable al usuario, la cual requiere de un buen manejo de la pantalla en modo gráfico, se ha utilizado para su desarrollo el lenguaje de programación VISUAL BASIC versión 6.0, el cual presenta facilidades para el manejo en modo gráfico.

La comunicación de la interfaz gráfica con el CODEC se la realiza vía USB con la ayuda del módulo USB MOD2, el cual es un módulo integrado para transferencia de datos sobre el interfaz USB a 1 *Megabyte* por segundo, descrito detalladamente en el capitulo anterior.

Cuando se utiliza el módulo USB MOD2, se puede usar un controlador VCP (*Virtual Com Port*), el cual simplifica la programación. Una vez instalado el controlador, Windows agrega el VCP a la pantalla de administración de dispositivos en el panel de control. En la Figura 3.1 se muestra la pantalla de administración de dispositivos, tanto antes como después de conectar el módulo.

Para visualizar el Administrador de dispositivos, en Windows XP, seleccionar Menú Inicio > Panel de Control > Rendimiento y Mantenimiento > Sistema > Hardware > Administrador de Dispositivos. Cuando un usuario desconecta el

módulo, Windows automáticamente quita el VCP de la pantalla de Administrador de Dispositivos.



(a) (b)

Figura 3.1 Administrador de dispositivos en Windows.

El número del puerto COM varía dependiendo del número de puertos COM existentes en la computadora personal y del número de módulos USB MOD2 conectados.

Una vez que el VCP está instalado, éste puede ser programado exactamente como un puerto serial COM usando el control *MSComm* desde Visual Basic, para lo cual debemos establecer el puerto COM al mismo número que aparece en el Administrador de dispositivos.

De esta forma podemos enviar datos desde el computador personal al CODEC para su configuración, como lo es el código de línea, la velocidad de transmisión y el nivel de señal, a utilizarse en la codificación.

La información a ser codificada será proporcionada en general por un dispositivo de comunicación serial, como es el pórtico de un terminal de comunicaciones o un computador personal. La Figura 3.2 muestra un esquema en el que se presenta la comunicación del CODEC con el computador personal vía USB y del computador personal con el CODEC vía tanto USB como RS-232.

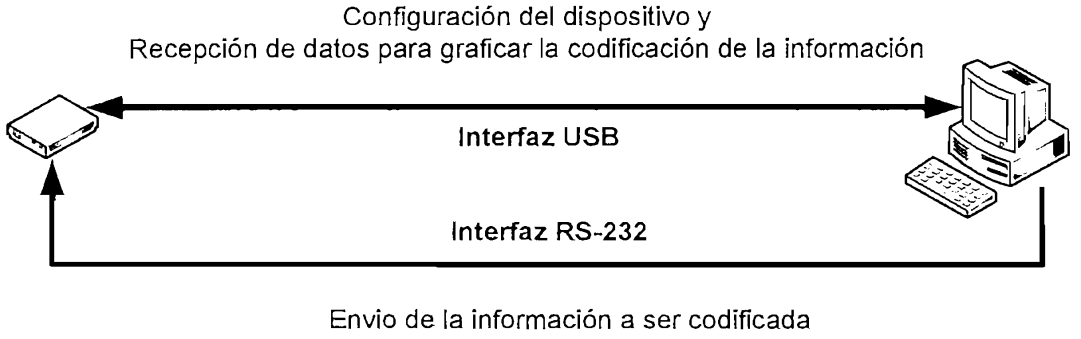


Figura 3.2 Comunicación desde el computador personal con el CODEC y viceversa.

Por otro lado, el software desarrollado presenta una ayuda para el usuario, en la cual se presenta información de cada uno de los códigos de línea con los cuales trabaja el CODEC, como lo es su algoritmo de generación, sus características, su regla de codificación, su diagrama de estado; además cuenta con aplicaciones para codificar y decodificar con sus respectivos gráficos de la señal de reloj, señal de datos y señal codificada, y de una aplicación para visualizar los espectros de potencia aproximados.

Los códigos de línea para los cuales se desarrolló la ayuda respectiva son:

1) El código AMI;
2) El código B3ZS;
3) El código Bifase-M;
4) El código CMI;
5) El código HDB3;
6) El código Manchester Diferencial;
7) El código Miller;
8) El código MLT-3;
9) El código MS43;

10) El código NRZ Polar;
11) El código RZ Polar y;
12) El código 4B3T.

La utilización del software desarrollado demanda de un procedimiento interactivo simple que se resume en los términos presentados a continuación.

3.2 RUTINAS DE INICIALIZACIÓN Y VERIFICACIÓN DEL ESTADO DE LA CONEXIÓN CON EL CODEC.

3.2.1 PRESENTACIÓN DEL SOFTWARE DESARROLLADO.

Al ejecutar el programa se mostrará el mensaje de presentación por un intervalo de tiempo determinado como se visualiza en la Figura 3.3.

Figura 3.3 Presentación del programa.

Después de la presentación del programa, la ventana que aparece la primera vez que se inicia el programa, permitirá al usuario seleccionar el VCP o USB *Serial Port* que *Windows* le ha asignado al *hardware* instalado y que fue observado en el administrador de dispositivos, ver Figura 3.1. La Figura 3.4 muestra la ventana en la cual seleccionaremos el VCP.

Figura 3.4 Selección del VCP (USB *Serial Port*)

Si se ha seleccionado un VCP equivocado, debemos buscar el archivo de texto CodecPortatil.txt en C:\Windows\System32\, para luego editarlo y cambiar el número encontrado por 0; de esta manera al iniciar el programa, se podrá seleccionar nuevamente el VCP.

Luego de seleccionar el VCP, se carga una pantalla con las opciones: Información de los Códigos de Línea y Configuración de los Parámetros del CODEC, éste último se habilitará siempre y cuando se haya detectado la conexión del CODEC. La Figura 3.5 muestra la pantalla anteriormente mencionada.

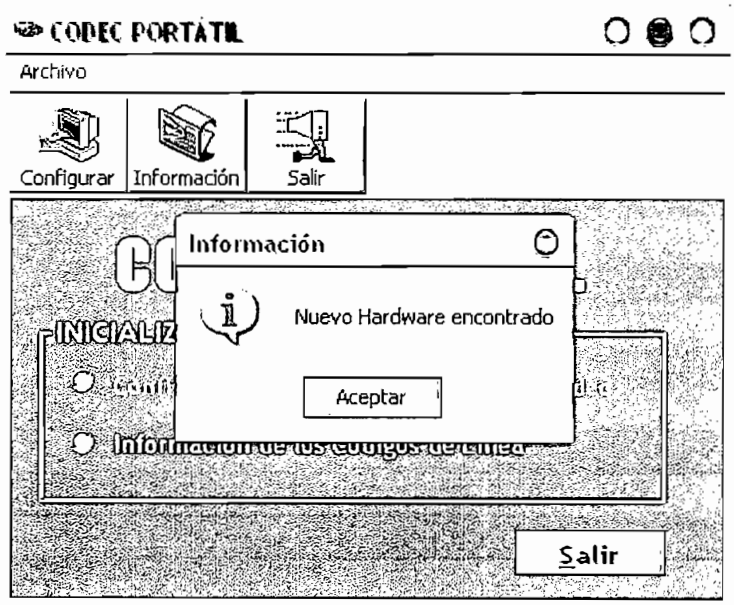
Figura 3.5 Pantalla para determinar la tarea a realizarse.

3.2.2 VERIFICACIÓN DEL ESTADO DE LA CONEXIÓN.

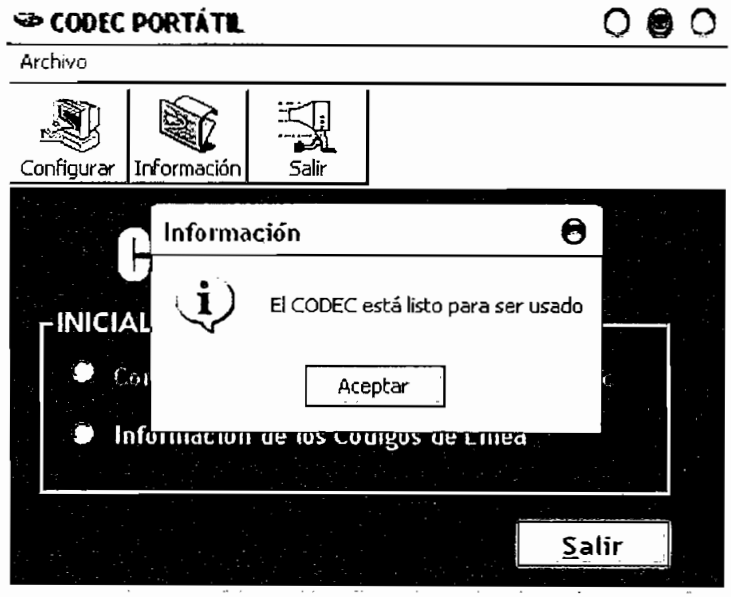
Para detectar la conexión del CODEC usamos el control *SysInfo* de Visual Basic, el cual permite responder a ciertos mensajes del sistema enviados a todas las aplicaciones por el sistema operativo, para nuestro caso utilizaremos los eventos

de *Plug and Play* (por ejemplo, el evento *DeviceArrival*) que nos ayudará a determinar modificaciones en la configuración del *hardware*.

De esta manera cuando ocurra el evento *DeviceArrival,* el programa indicará al usuario que el CODEC ha sido conectado y que está listo para ser usado. En la Figura 3.6 se muestra los diferentes mensajes a darse al usuario cuando se ha conectado el CODEC al computador personal.



(a)



(b)

Figura 3.6 Detección de la conexión del CODEC.

Una vez detectada la conexión del CODEC se verifica constantemente el estado de conexión con el CODEC, para lo cual el programa permanentemente estará chequeando la llegada de datos al VCP, y cuando se presente ausencia de datos en el mismo, esperará un tiempo de 1 segundo para indicar al usuario que el CODEC ha sido desconectado. En la Figura 3.7 se muestra el mensaje presentado al usuario cuando el CODEC esté desconectado.

Figura 3.7 Detección de la desconexión del CODEC.

En Visual Basic se desarrolló una subrutina que permite detectar la conexión del dispositivo con la ayuda del control *SysInfo,* como lo muestra la Figura 3.8.a, una subrutina para detectar datos en el VCP con la ayuda del control *MSComm*, como lo muestra la figura 3.8.b y una subrutina para determinar la desconexión del dispositivo con la ayuda del control *Timer*, como lo muestra la Figura 3.8.c.

Cuando se ha detectado la conexión del dispositivo se guarda en la variable global CODEC el número 1 que indicará a las siguientes pantallas que el dispositivo está conectado y así habilitar la opción de Configurar los parámetros del CODEC. El control *MSComm* desactiva el *timer* cada vez que recibe un dato en el VCP y al no detectar datos en el VCP por un intervalo de tiempo de 1 segundo, activa el *timer*, el cual mostrará que el dispositivo ha sido desconectado,

guardando en la variable global CODEC el valor de 0 para indicar a las siguientes pantallas que el dispositivo no está conectado y por lo tanto no debe habilitar la opción de Configurar los parámetros del CODEC y debe mantener desactivado el *timer*.

```
Private Sub SysInfo1 DeviceArrival(ByVal tipoDispositivo As Long, ByVal IdDispositivo As Long, ByVal
                                   ByVal nombreDispositivo As String, ByVal datosDispositivo As Long)
 'Se produce cuando se agrega un nuevo dispositivo al sistema
    p = MsgBox("Nuevo Hardware encontrado", vbInformation, "Información")
    p = MsgBox("El CODEC esta listo para ser usado", vbInformation, "Información")
    mnuDispositivoDetectado.Enabled = True
  'Configura y abre el puerto para la comunicación con el dispositivo
    MSComm1.CommPort = COM
    MSComm1.PortOpen = True ' ABRO EL PUERTO
    MSComm1.Output = "Z"
    CODEC = 1
 End Sub
```

(a)

```
Private Sub MSComm1_OnComm()
'SE EJECUTA SIEMPRE HAY UN DATO EN PUERTO SERIAL
'Permite la recepción de un dato para detectar si el codec esta o no conectado
    Timer1.Enabled = False 'Detiene el timer
    Timer1.Enabled = True   'Inicializa el timer
End Sub
```

(b)

```
Private Sub Timer1_Timer()
'Timer que permite detectar si el codec portátil fue desconectado
    CODEC = 0
    p = MsgBox("El CODEC ha sido Desconectado", vbInformation, "Información")
    MSComm1.PortOpen = False ' CIERRO EL PUERTO
    Timer1.Enabled = False
    mnuDispositivoDetectado.Enabled = False
End Sub
```

(c)

Figura 3.8 Rutinas de programación en Visual Basic.

El procedimiento descrito para presentar el programa y verificar el estado de la conexión con el CODEC se resume en el diagrama de flujo que se muestra en la Figura 3.9.

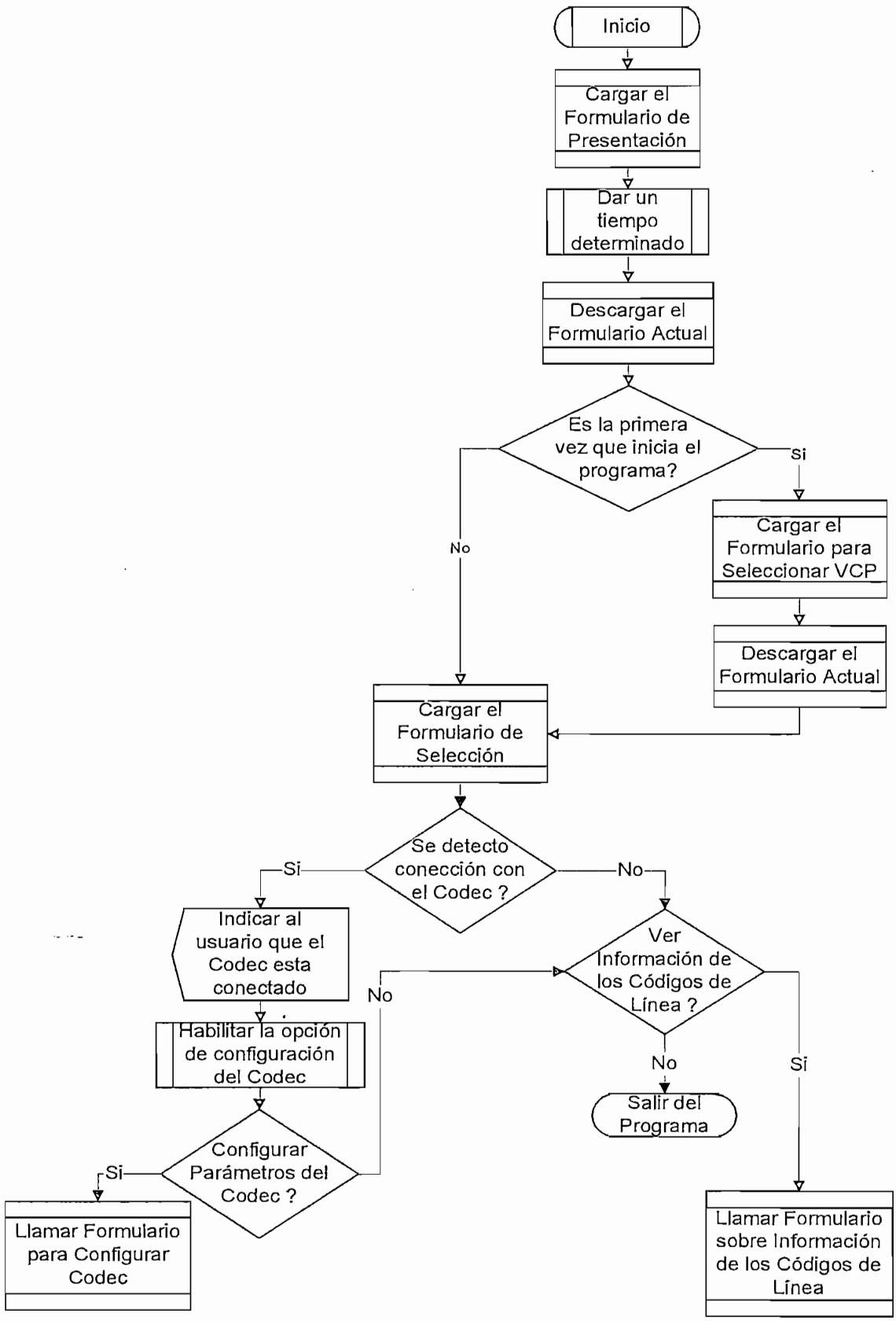


Figura 3.9 Diagrama de flujo de la rutina de inicialización del programa.

3.3 RUTINA DE CONFIGURACIÓN DEL DISPOSITIVO.

Los parámetros básicos que presenta el CODEC para ser configurado son: el tipo de código, la velocidad de transmisión estandarizada a la que le es posible trabajar y el nivel de señal. La interfaz gráfica que permitirá seleccionar los parámetros básicos del CODEC se muestra en la Figura 3.10, tomando en cuenta que la pantalla estará habilitada siempre y cuando el CODEC esté conectado.

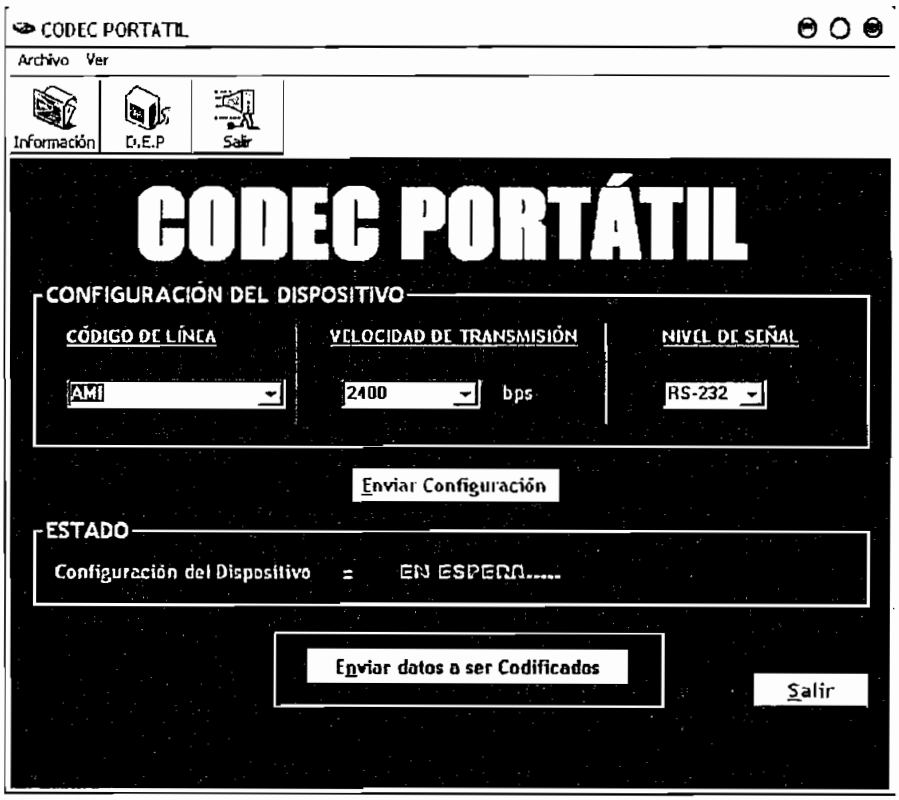


Figura 3.10 Pantalla para configurar el CODEC portátil.

Una vez que se haya enviado la configuración al dispositivo a través del VCP, la opción de "Enviar Datos a ser Codificados" se habilitará y la opción de "Enviar Configuración" seleccionada, cambiará a "Cambiar Parámetros" para configurar al CODEC de una manera diferente. Con la opción de Enviar Datos a ser Codificados podremos enviar los datos a ser codificados a través del interfaz RS-232 y visualizar la señal de reloj, la señal de datos y la señal de datos codificados.

El procedimiento para configurar el CODEC portátil se presenta en el diagrama de flujo que se muestra en la Figura 3.11.

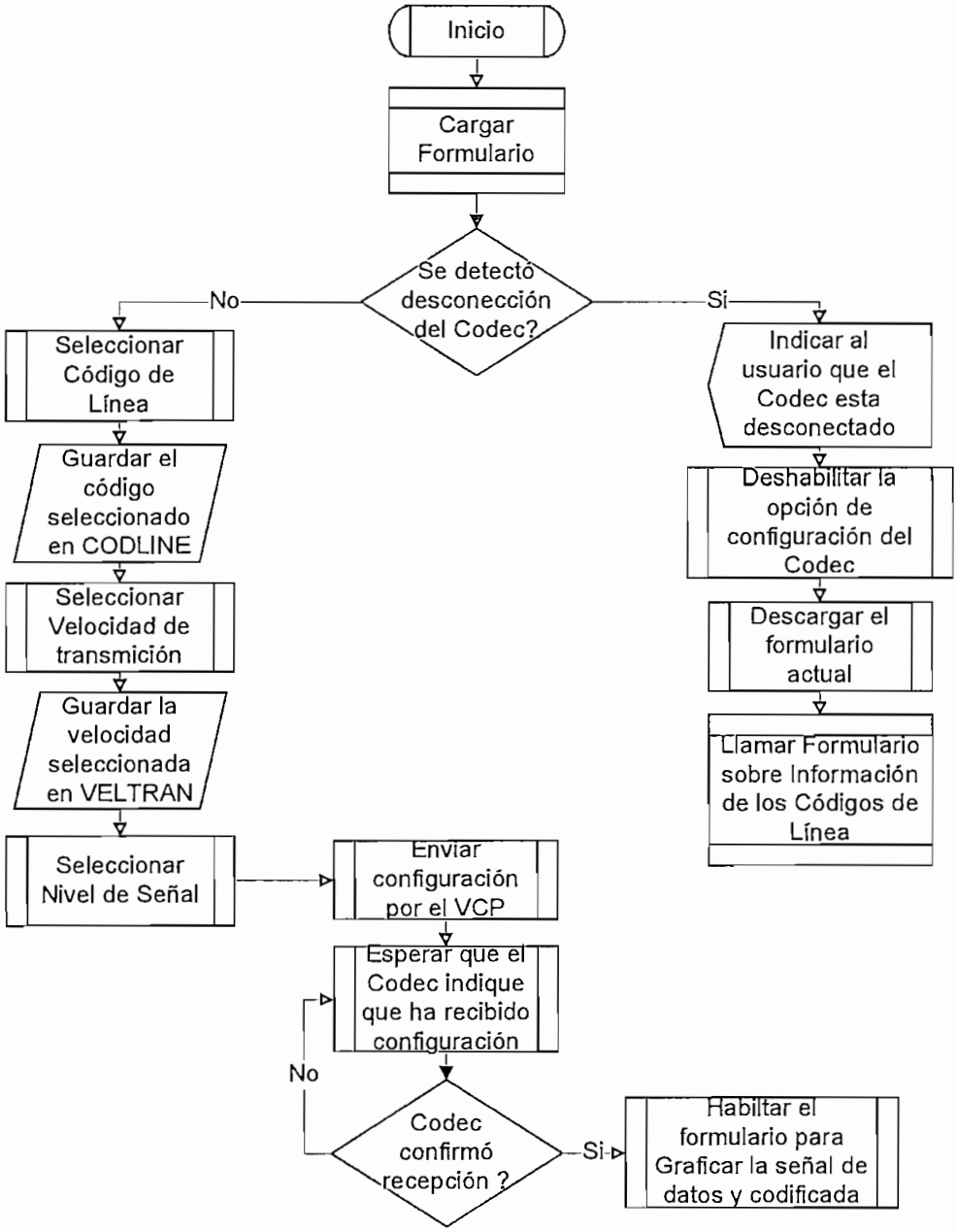


Figura 3.11 Diagrama de flujo de la rutina de configuración del CODEC.

La subrutina desarrollada en Visual Basic para enviar la configuración del CODEC se la realiza con la ayuda del control *MSComm*, enviando los datos de configuración al CODEC a través del VCP. En esta subrutina guardamos en las variables globales CODLINE y VELTRAN un dato que identifique el Código de Línea y la Velocidad de Transmisión seleccionados, respectivamente. Estas variables nos ayudarán a graficar la señal de reloj, la señal de datos y la señal codificada.

3.4 RUTINA PARA ENVIAR LOS DATOS A SER CODIFICADOS Y GRAFICAR LA SEÑAL DE RELOJ, SEÑAL DE DATOS Y SEÑAL CODIFICADA.

La rutina de Configurar Dispositivo habilitará la rutina para enviar datos a través del interfaz serial y graficar la señal codificada.

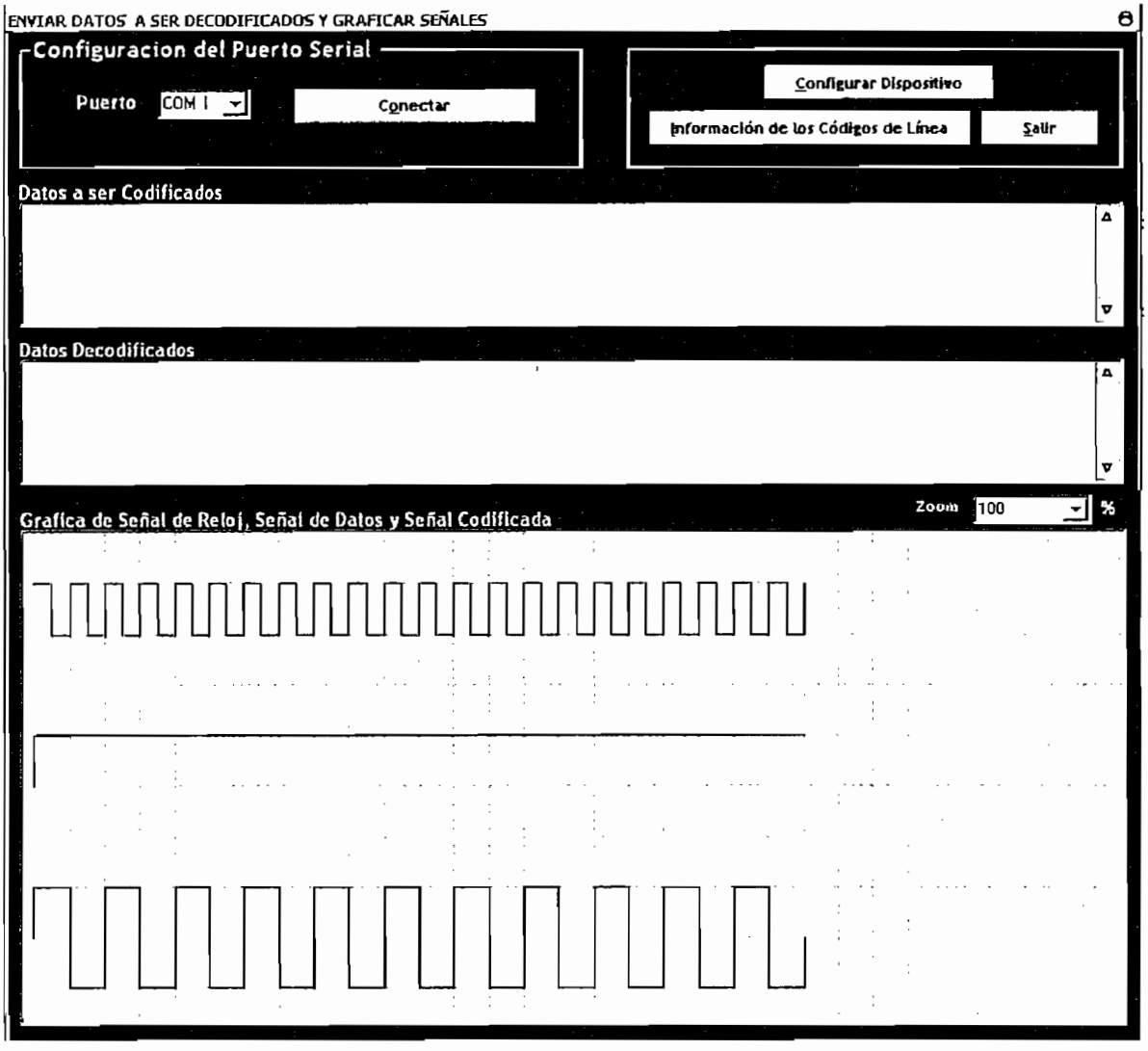
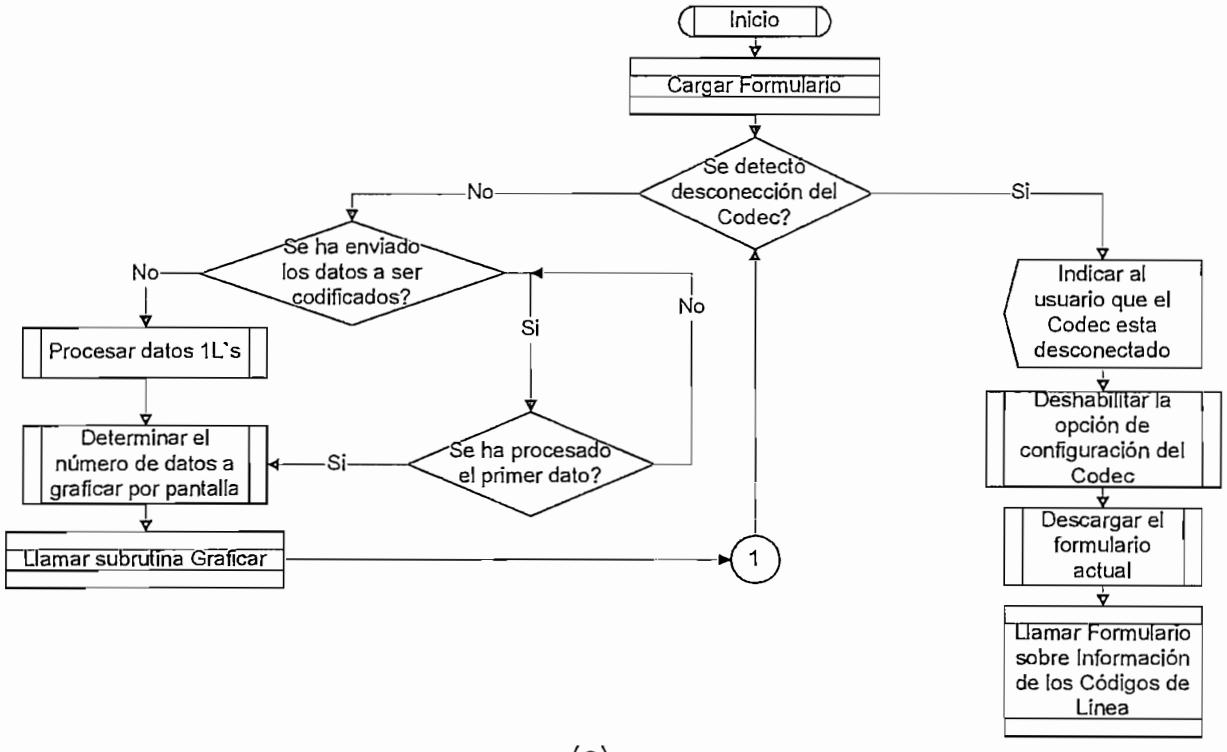


Figura 3.12 Pantalla para enviar datos a ser codificados y visualizar las señales de Reloj, Datos y Codificada.

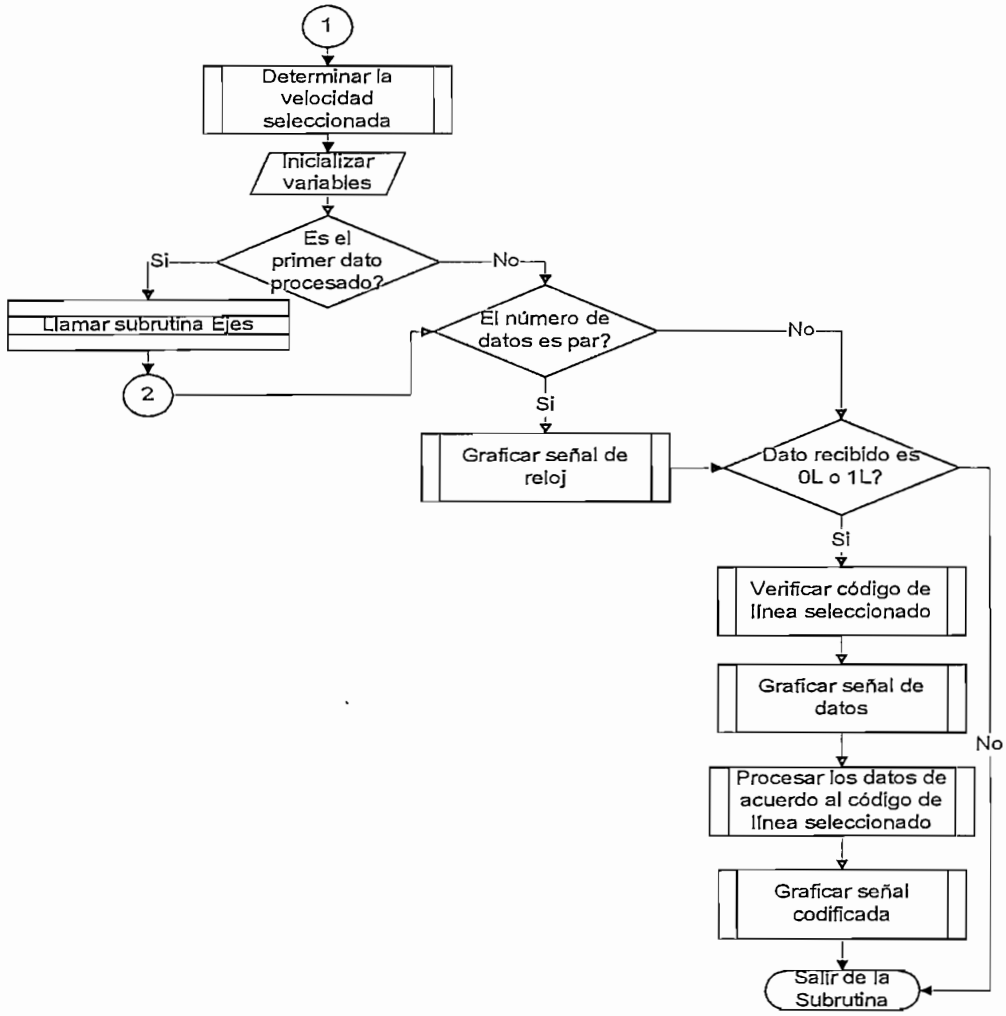
En la Figura 3.12 se puede observar la pantalla que nos ayudará a enviar los datos a ser codificados y graficar la señal de reloj, la señal de datos y la señal

codificada, como ejemplo hemos seleccionado el código de línea AMI, el cual está codificando los 1L que el CODEC por defecto procesa. Las señales a visualizarse en pantalla, se desarrollarán con la ayuda de controles gráficos como lo son el Control *Line* y Control *Picture Box* de Visual Basic.

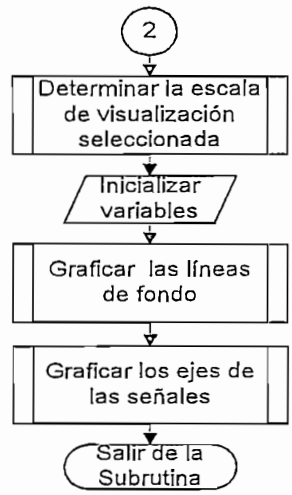
Los pasos a seguirse para realizar la representación de las señales mencionadas se resume en el diagrama de flujo expuesto en la Figura 3.13. En la Figura 3.13.a tenemos la rutina principal que determinará el estado de la conexión del dispositivo como el número de datos a graficar por pantalla; en la Figura 3.13.b tenemos la subrutina que procesará los datos a ser codificados, para graficar las señales respectivas; en la Figura 3.13.c tenemos la subrutina que graficará los ejes de las señales, las líneas de fondo, de acuerdo a la escala de visualización seleccionada.



(a)



(b)



(c)

Figura 3.13 Diagrama de flujo de la rutina para graficar la señales de: reloj, datos y codificada.

3.5 RUTINAS DE INFORMACIÓN DE LOS CÓDIGOS DE LÍNEA.

Después de la presentación del programa en la rutina de inicialización, se carga una pantalla con dos opciones: Información de los Códigos de Línea y Configurar Parámetros del Codec Portátil, este último se habilitará siempre y cuando se haya detectado la conexión del CODEC. Al elegir la opción correspondiente a la Información de los Códigos de Línea se cargará una pantalla con la opción de seleccionar el código de línea del cual se requiera conocer sus características o trabajar con las aplicaciones para codificar y decodificar. Además la pantalla permite utilizar la aplicación para ver los espectros de potencia aproximados como lo muestra la Figura 3.14.

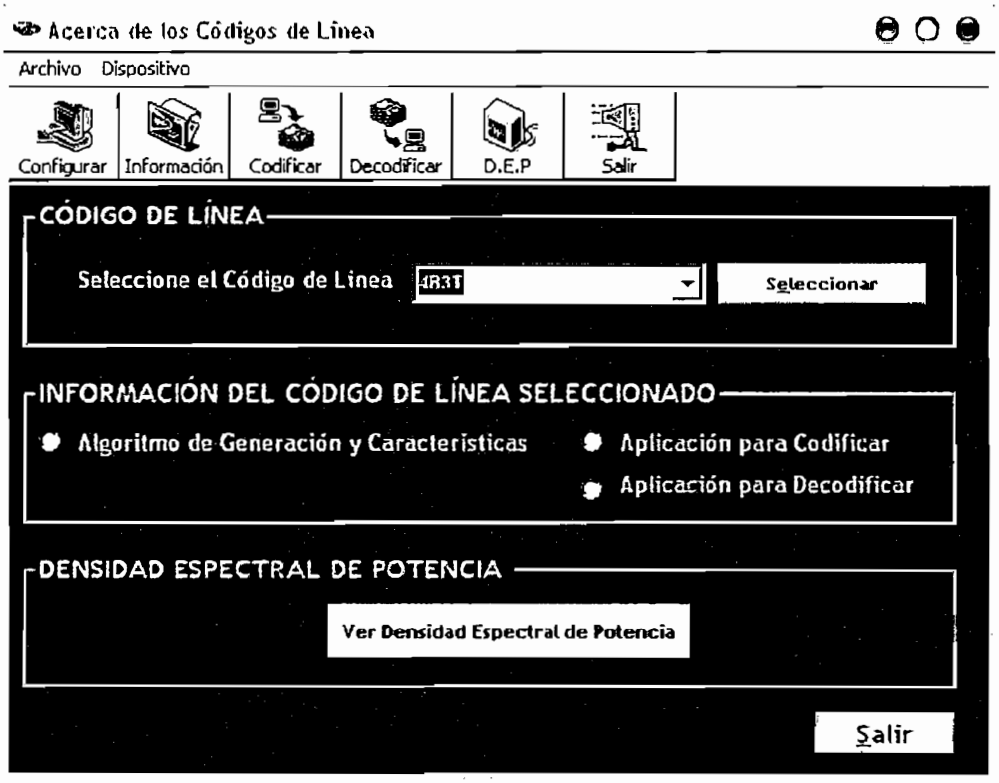


Figura 3.14 Pantalla sobre la información de los Códigos de Línea

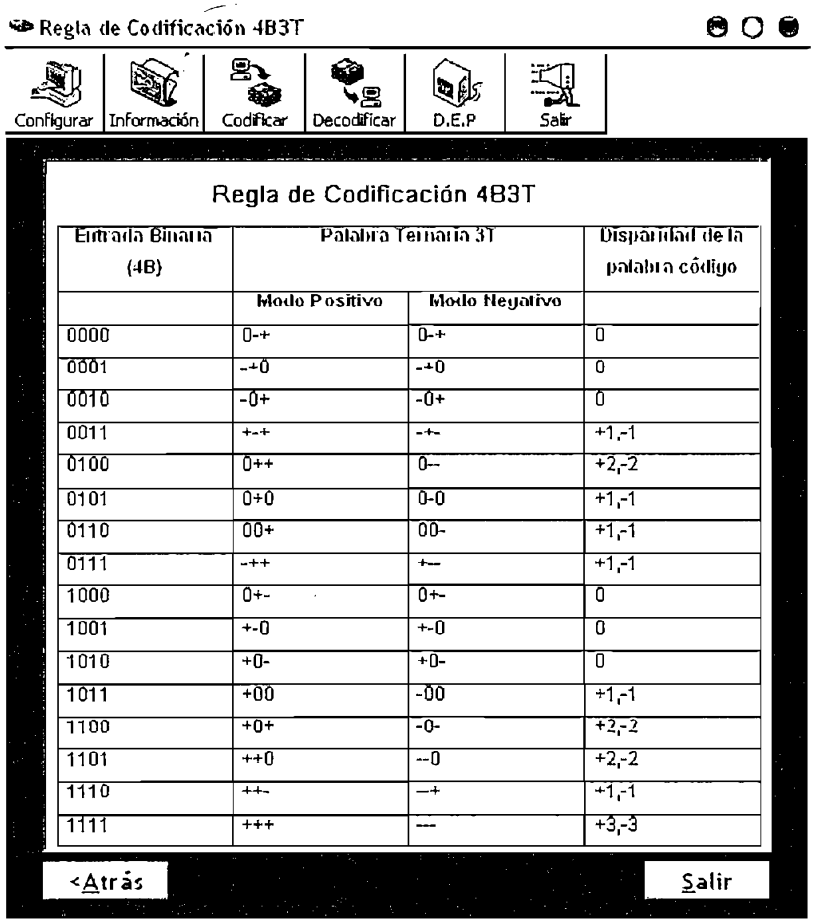
Una vez seleccionado el código de línea, la opción de "Seleccionar" cambiará a "Cambiar" para permitir cambiar el código de línea seleccionado y se habilitarán las opciones correspondientes a la información del código de línea. Como información del código de línea seleccionado tenemos tres opciones descritas a continuación.

3.5.1 ALGORÍTMO DE GENERACIÓN Y CARACTERÍSTICAS.

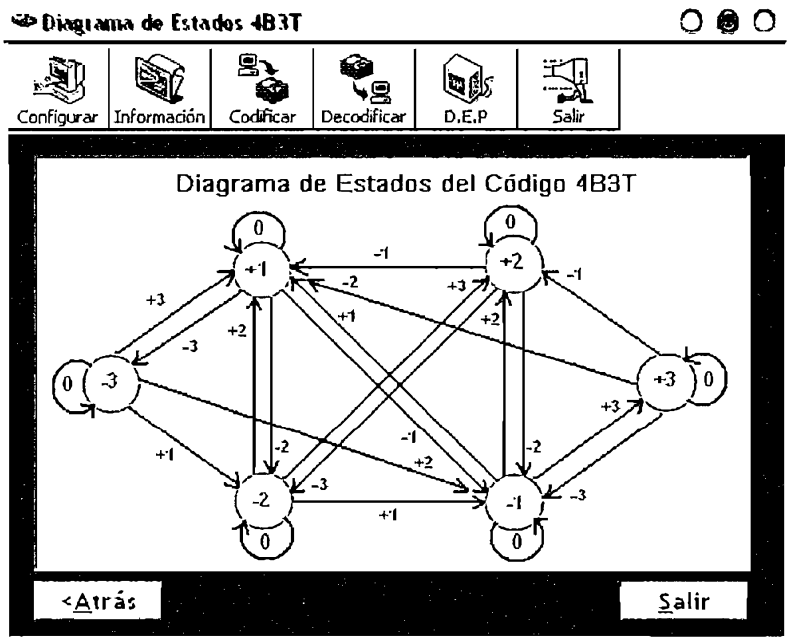
Esta opción cargará una pantalla que permitirá visualizar las características principales del código de línea como lo es su algoritmo de generación, su regla de codificación y su diagrama de estados, estas dos últimas opciones estarán presentes de acuerdo al código seleccionado como lo muestra la Figura 3.15.

Figura 3.15 Pantalla de las características principales del código de línea.

En este caso se ha seleccionado el código de línea 4B3T, el cual presenta las opciones de visualizar su regla de codificación y visualizar su diagrama de estados como lo muestra la Figura 3.16.



(a)



(b)

Figura 3.16 (a) Regla de codificación del código de 4B3T, (b) Diagrama de estados del código 4B3T.

3.5.2 APLICACIÓN VISUALIZAR LOS ESPECTROS DE POTENCIA APROXIMADOS

Si optamos por visualizar el espectro de potencia aproximado, se cargará una pantalla que permitirá seleccionar el código de línea, la escala de visualización y la probabilidad de 1L´s, esta última opción no se habilitará para los códigos de línea HDB3 y Miller como lo muestra la Figura 3.17. .

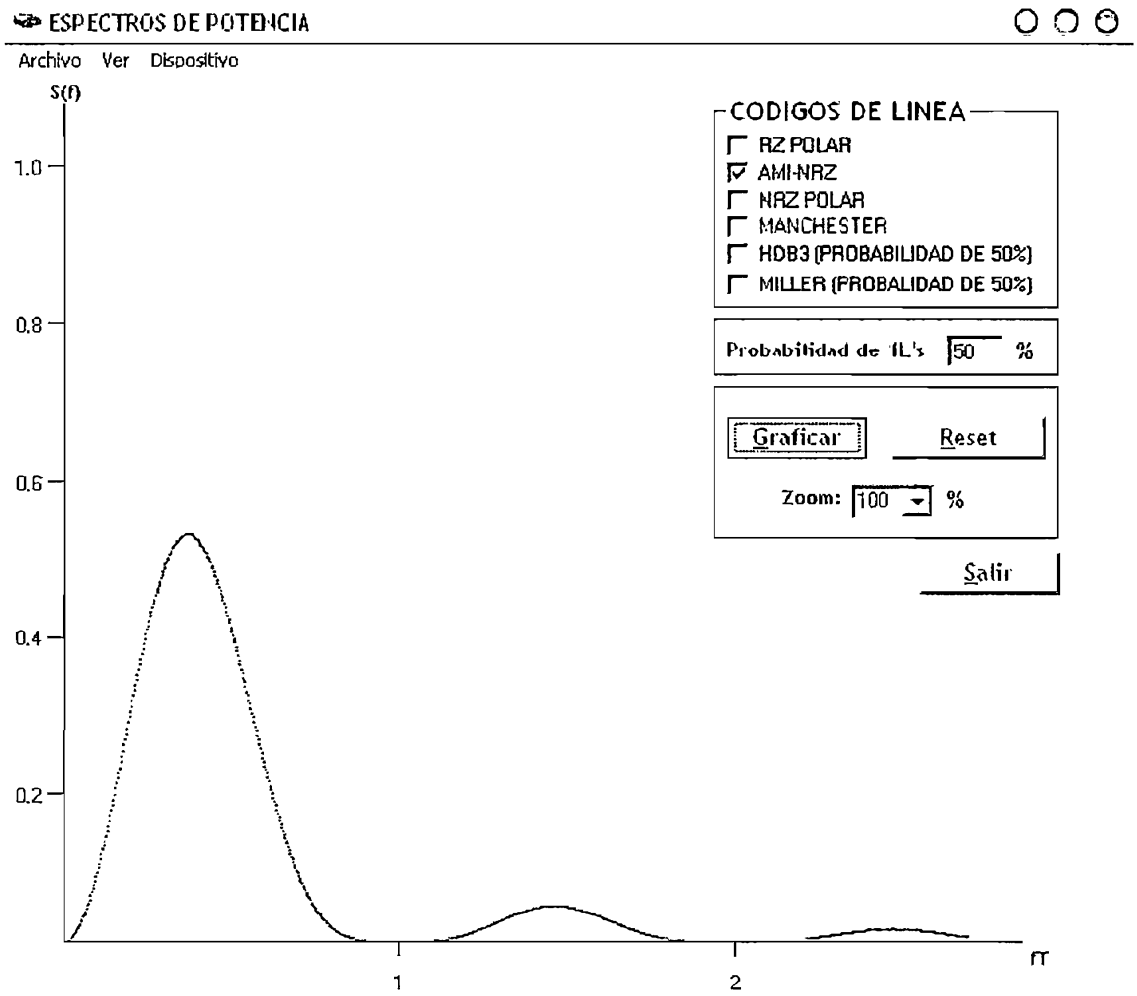


Figura 3.17 Espectro de Potencia Aproximado del Código de Línea AMI.

3.5.3 APLICACIÓN PARA CODIFICAR

Esta opción cargará una pantalla que permitirá codificar una secuencia de datos binarios, de acuerdo al código de línea seleccionado. Además tenemos la opción

de elegir la velocidad de transmisión, una escala de visualización y la polaridad inicial, esta última se habilitará dependiendo del código de línea seleccionado. Como resultado tenemos la visualización de las señales de reloj, datos y codificada, los niveles de la señal codificada y los niveles de polaridad acumulada, este último dependerá del código de línea seleccionado como se muestra en la Figura 3.18.

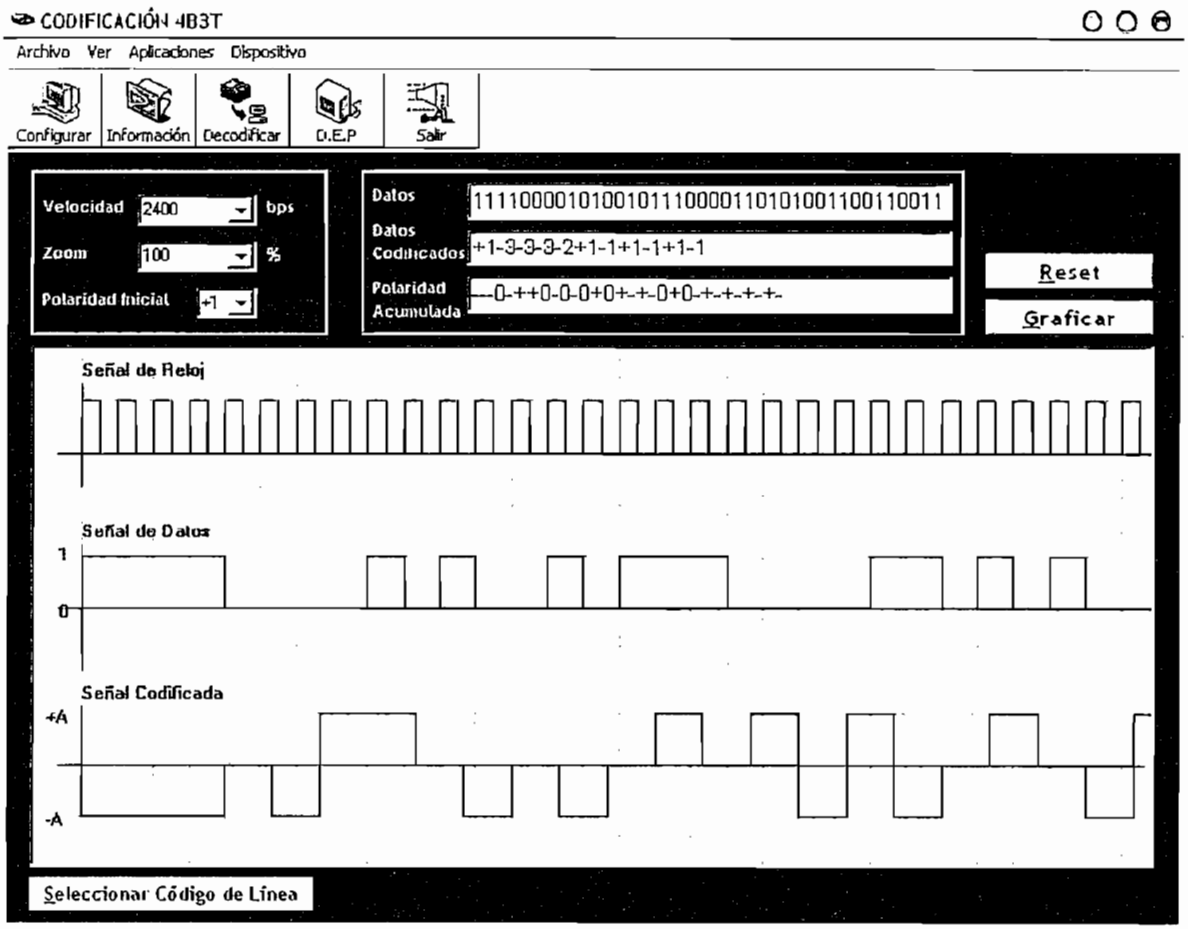


Figura 3.18 Pantalla correspondiente a la aplicación para codificar.

3.5.4 APLICACIÓN PARA DECODIFICAR

Esta opción cargará una pantalla que permitirá decodificar una secuencia de niveles de señal (señal codificada), de acuerdo al código de línea seleccionado. Para permitir el ingreso correcto de los niveles de señal debemos leer las instrucciones presentadas en la parte superior de la pantalla para evitar decodificaciones erróneas, luego de aceptar estas instrucciones se habilitarán las

opciones de ingresar los niveles de señal, elegir la velocidad de transmisión y una escala de visualización. Como resultado tenemos la visualización de las señales de reloj, codificada y decodificada, los datos binarios de la señal de datos como se muestra en la Figura 3.19.

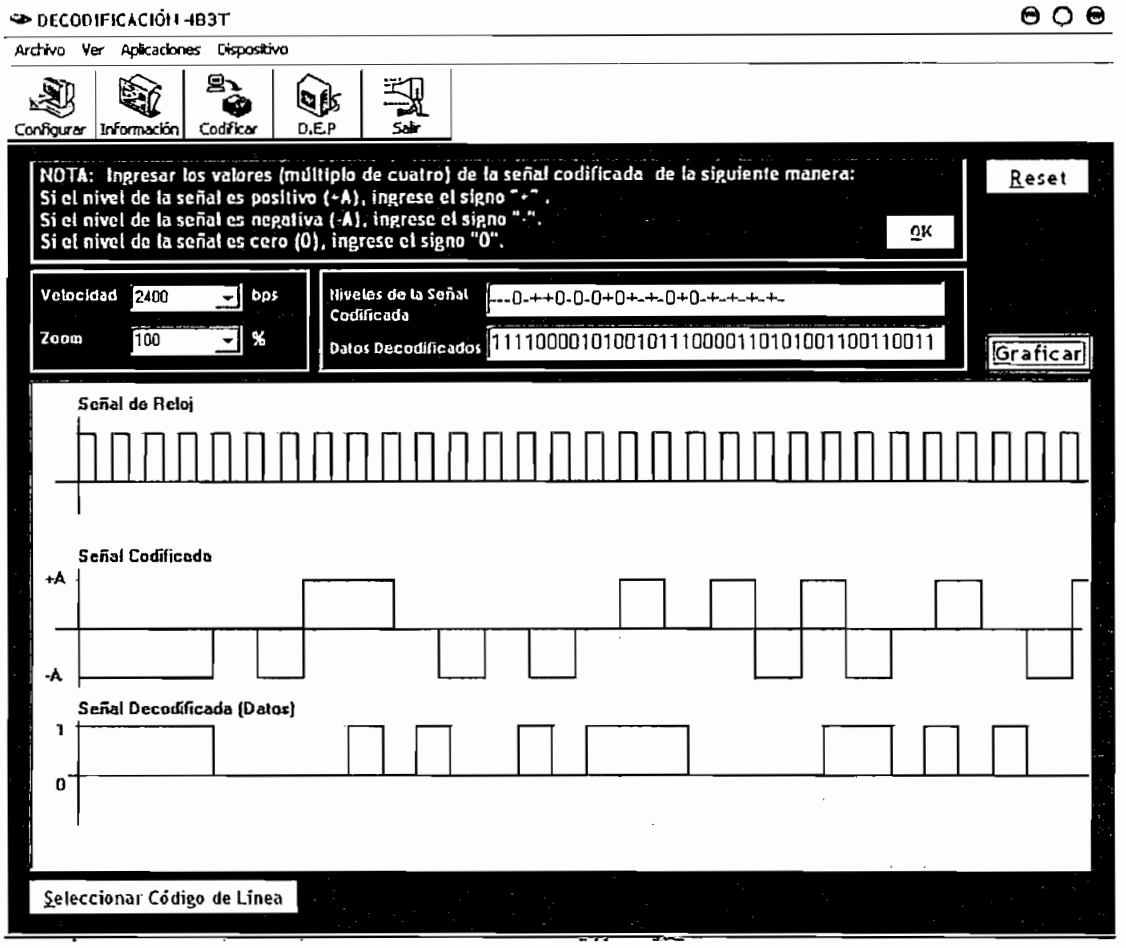


Figura 3.19 Pantalla correspondiente a la aplicación para decodificar.

3.6 RUTINAS DE CODIFICACION Y DECODIFICACION PARA LOS CODIGOS DE LINEA.

Las rutinas de codificación y decodificación para los diferentes códigos de línea inician chequeando la velocidad de transmisión, la escala de visualización y la polaridad inicial (opción presente dependiendo del código de línea) que se ha seleccionado, además cada rutina previo a la codificación verifica que los datos

ingresados sean correctos y correspondan a datos binarios, si se trata de una decodificación la rutina verifica que los niveles de señal ingresados correspondan a una señal codificada de acuerdo al código seleccionado, si se encuentra un error la rutina indicará el error.

3.6.1 Rutina para el código AMI

a. Codificación AMI

La codificación AMI inicia chequeando si el dato ingresado corresponde a 1L o 0L, si es 0L escribirá un nivel de señal cero, si el dato es 1L chequeará si el número de 1L es impar o par, si es impar escribirá un nivel de señal positivo, si es par escribirá un nivel de señal negativo. Una vez determinada la secuencia de niveles de señal, procederá a graficar las señales de reloj, datos y codificada como se muestra en la Figura 3.20.

b. Decodificación AMI

La decodificación de señales AMI inicia chequeando el nivel de señal ingresado, si el nivel de señal es cero escribirá como dato decodificado 0L, si el nivel de señal es positivo o negativo chequeará si el nivel anterior, diferente a cero, fue diferente al nivel actual, para escribir como dato decodificado 1L, caso contrario informará que existe un error en la secuencia de niveles de señal a decodificarse. Una vez determinada la secuencia de datos decodificados, se procederá a graficar las señales de reloj, datos y codificada como se muestra en la Figura 3.21.

3.6.2 Rutina para el código B3ZS

a. Codificación B3ZS

La codificación B3ZS inicia chequeando si el dato ingresado corresponde a 1L o 0L, si es 1L se aplicará la regla AMI; si el dato es 0L chequeará el número de 0L,

si es diferente a tres se escribirá un nivel de señal cero, si es igual a tres se chequeará si el número de 1L es par o impar, para reemplazar la secuencia de tres ceros por la secuencia B0V o 00V, donde B es un pulso de relleno y V es el pulso de violación de la regla de AMI. Una vez determinada la secuencia de niveles de señal, procederá a graficar las señales de reloj, datos y codificada como se muestra en la Figura 3.22.

b. Decodificación B3ZS

La decodificación de señales B3ZS inicia chequeando tres niveles de señal ingresados, si el nivel de señal es cero se escribirá como dato decodificado 0L, si el nivel de señal es positivo o negativo chequeará si el nivel es una violación, si es una violación se escribirá como datos decodificados tres 0L, caso contrario se escribirá 1L. Una vez determinada la secuencia de datos decodificados, se procederá a graficar las señales de reloj, datos y codificada como se muestra en la Figura 3.23.

3.6.3 Rutina para el código Bifase M

a. Codificación Bifase M

La codificación Bifase M inicia chequeando si el dato ingresado corresponde a 1L o 0L, si es 1L se escribirá un nivel de señal positivo que implicará graficar la señal codificada con transición al inicio y a la mitad del periodo de bit, si es 0L se escribirá un nivel se señal negativo que implicará graficar la señal codificada con transición al inicio del periodo de bit. Una vez determinada la secuencia de niveles de señal, procederá a graficar las señales de reloj, datos y codificada como se muestra en la Figura 3.24.

b. Decodificación Bifase M

La decodificación de señales Bifase M inicia chequeando el nivel de señal ingresado, si es un nivel de señal positivo se escribirá como dato decodificado 1L,

si es un nivel de señal negativo se escribirá como dato decodificado 0L. Una vez determinada la secuencia de datos decodificados, se procederá a graficar las señales de reloj, datos y codificada como se muestra en la Figura 3.25.

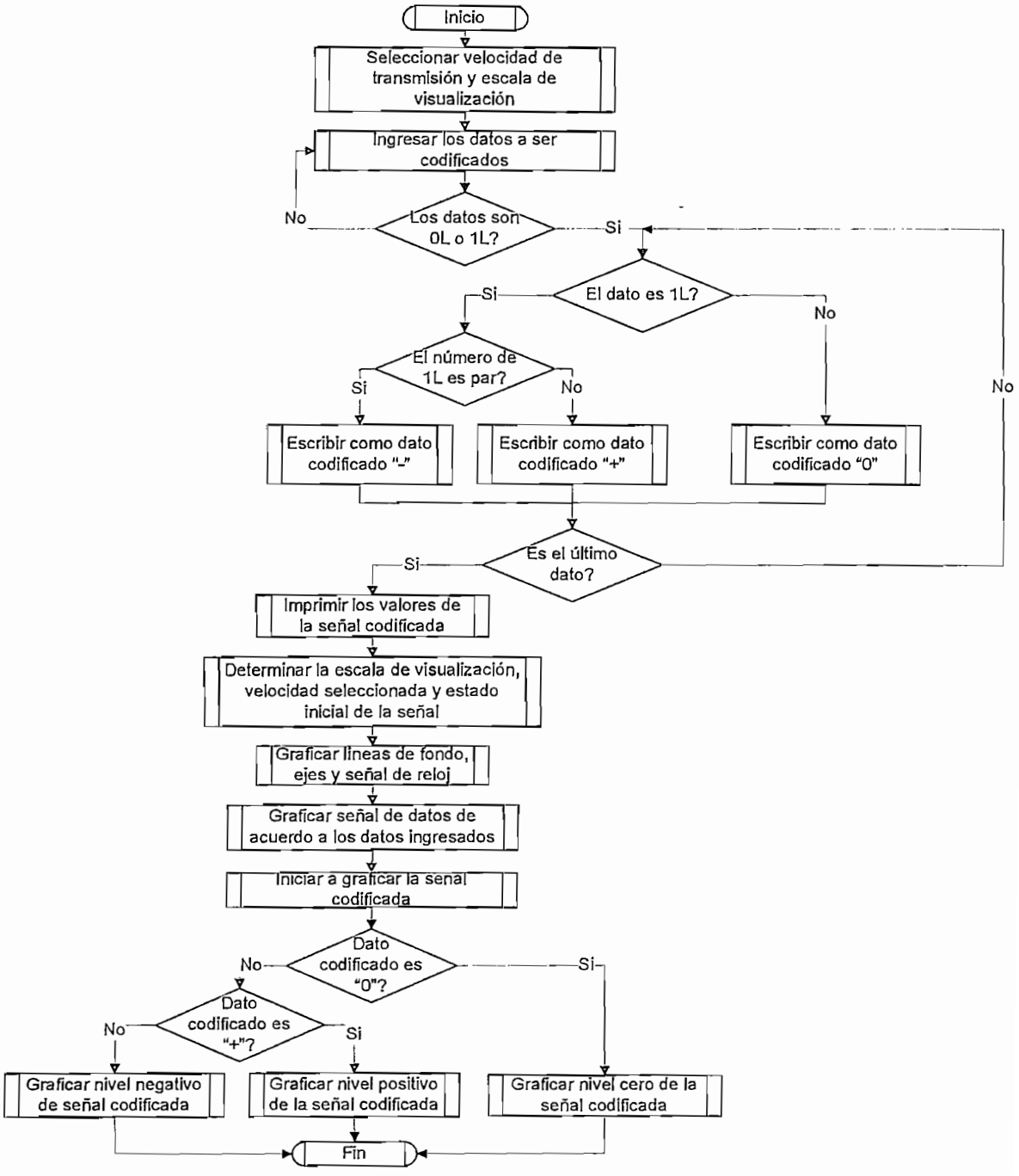


Figura 3.20 Diagrama de flujo de la rutina para la codificación AMI.

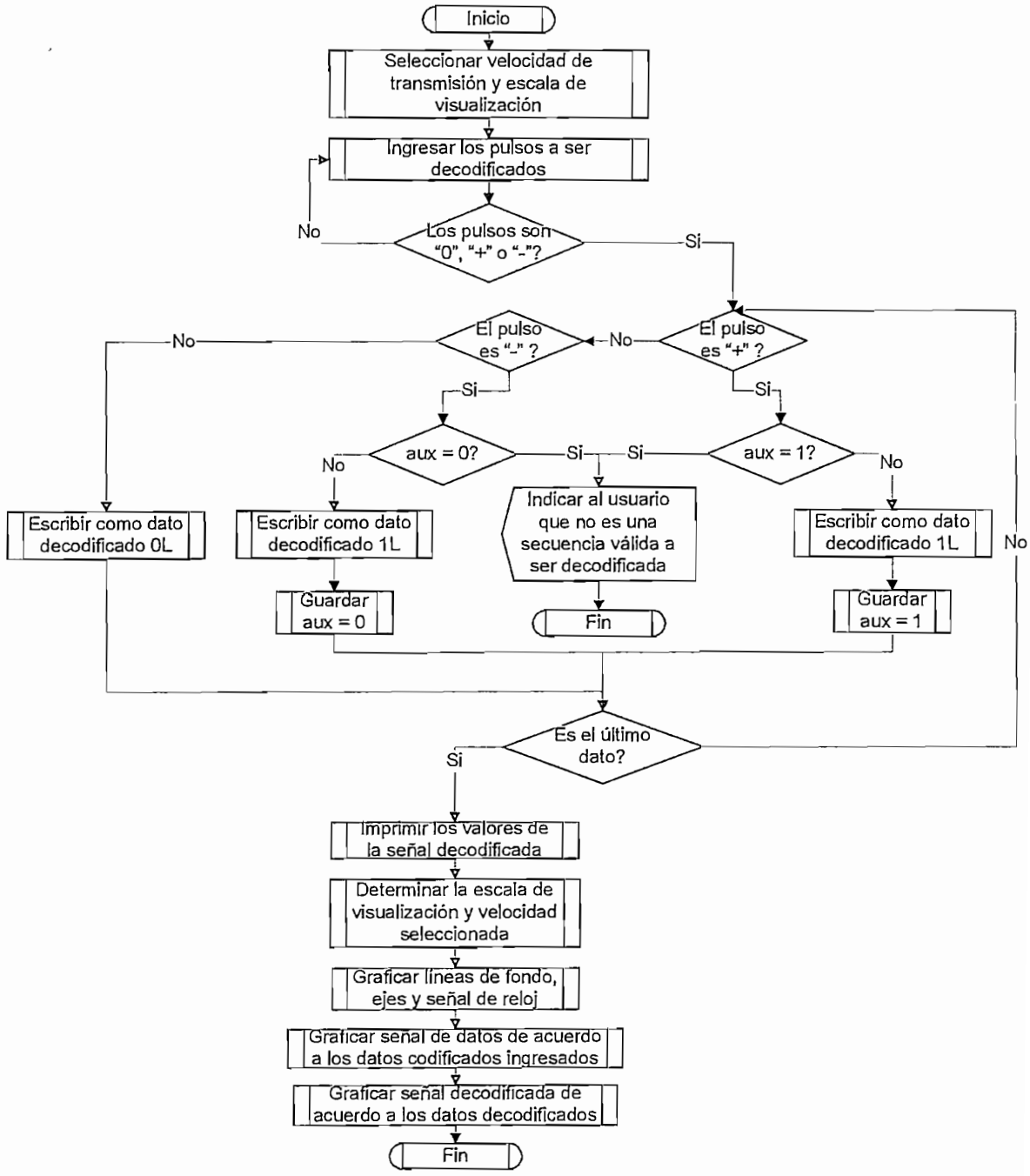


Figura 3.21 Diagrama de flujo de la rutina para la decodificación AMI.

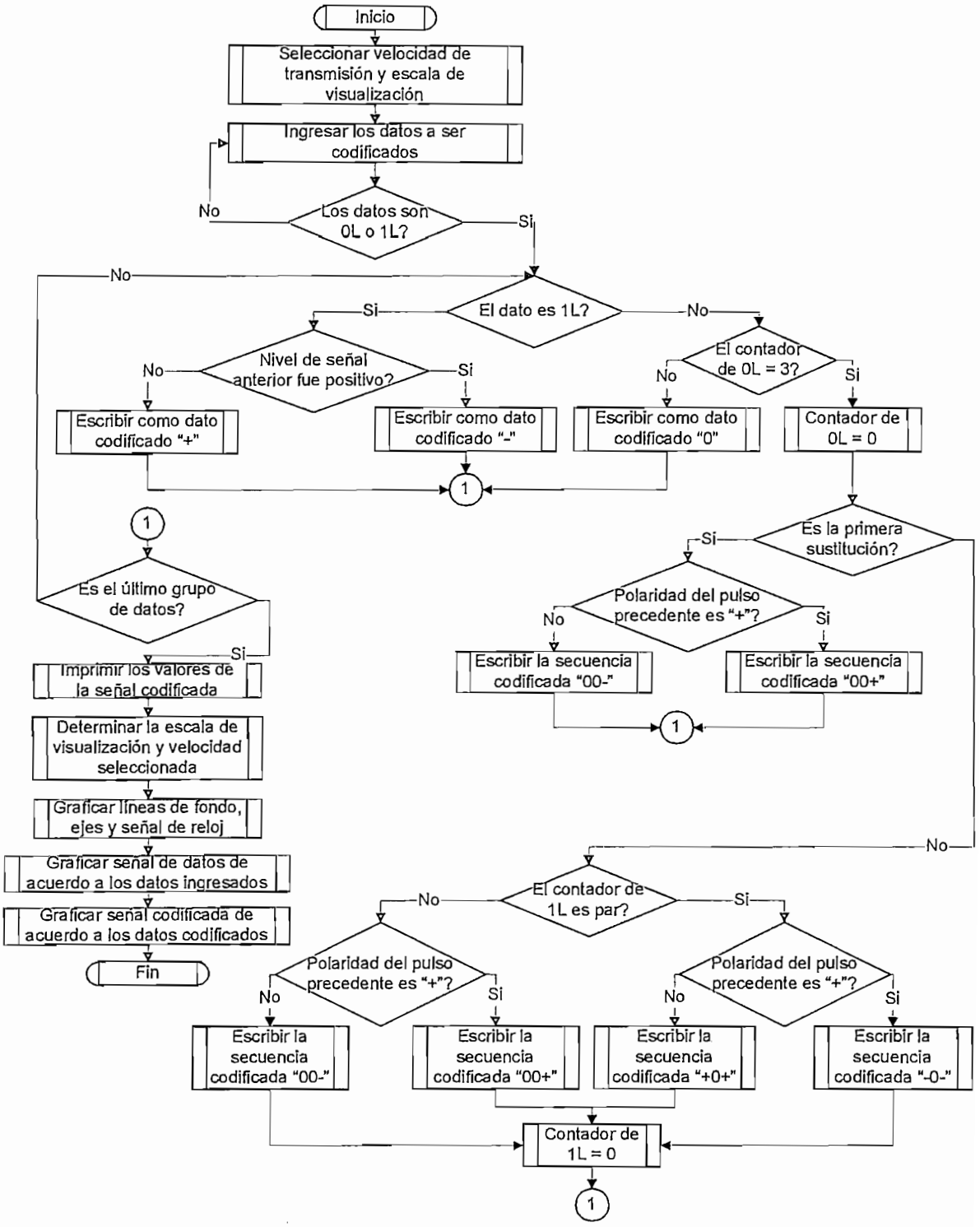


Figura 3.22 Diagrama de flujo de la rutina para la codificación B3ZS.

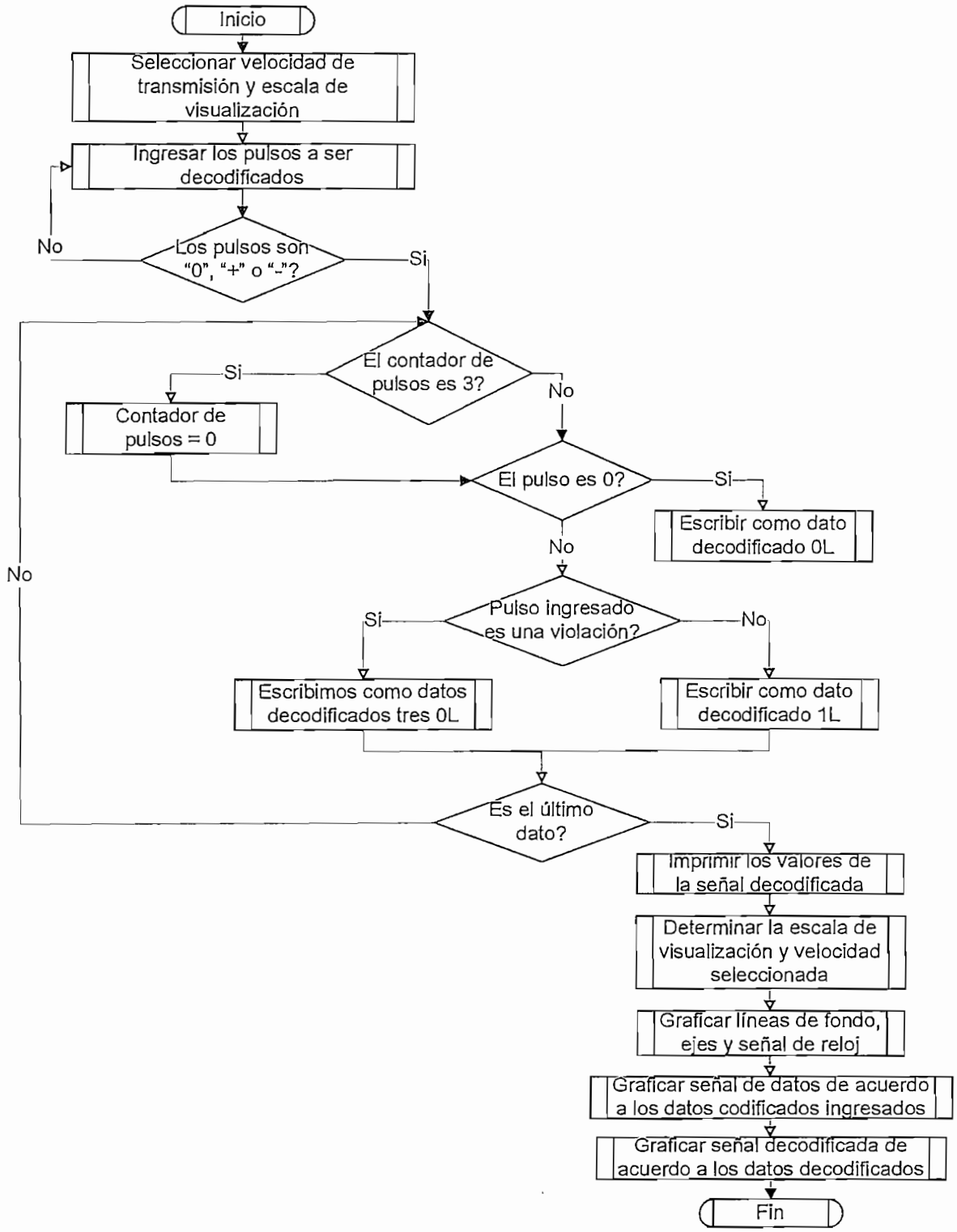


Figura 3.23 Diagrama de flujo de la rutina para la decodificación B3ZS.

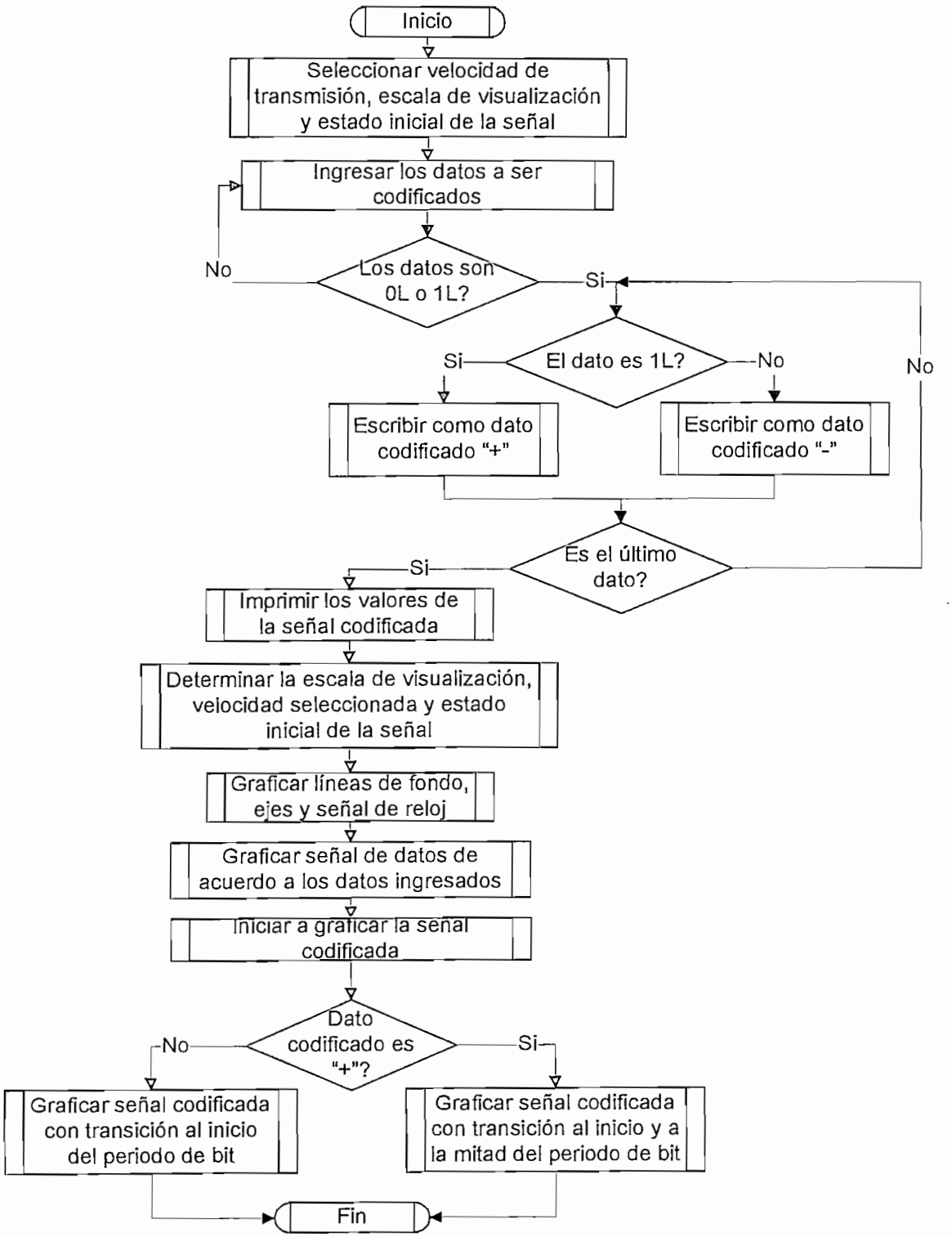


Figura 3.24 Diagrama de flujo de la rutina para la codificación Bifase M.

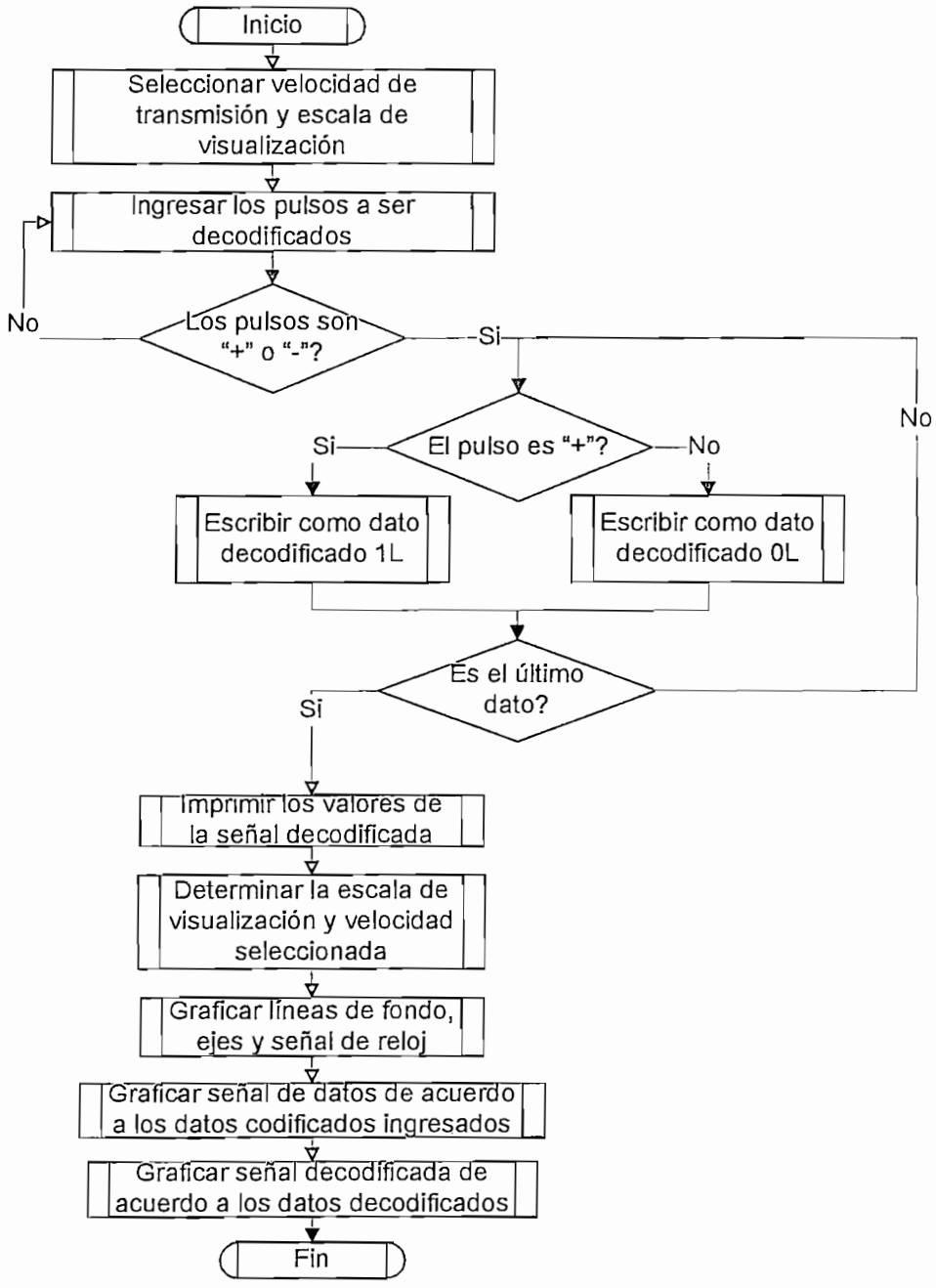


Figura 3.25 Diagrama de flujo de la rutina para la decodificación Bifase M.

3.6.4 Rutina para el código CMI

a. Codificación CMI

La codificación de señales CMI es similar a la codificación de señales AMI, con la diferencia que un nivel de señal cero implicará graficar la señal codificada con una transición de nivel negativo a nivel positivo a la mitad del periodo de bit.

b. Decodificación CMI

La decodificación de señales CMI se la realiza con las mismas consideraciones expuestas para la decodificación AMI.

3.6.5 Rutina para el código HDB3

a. Codificación HDB3

La codificación HDB3 inicia chequeando si el dato ingresado corresponde a 1L o 0L, si es 1L se aplicará la regla AMI; si el dato es 0L chequeará el número de 0L, si es diferente de tres se escribirá un nivel de señal cero, si es igual a cuatro se chequeará si el numero de 1L es par o impar, para reemplazar la secuencia de cuatro ceros por la secuencia B00V o 000V, donde B es un pulso de relleno y V es el pulso de violación de la regla de AMI. Una vez determinada la secuencia de niveles de señal, procederá a graficar las señales de reloj, datos y codificada. Para realizar la codificación se toma pasos similares a los considerados para la codificación B3ZS.

b. Decodificación HDB3

La decodificación de señales HDB3 inicia chequeando cuatro niveles de señal ingresados, si el nivel de señal es cero se escribirá como dato decodificado 0L, si el nivel de señal es positivo o negativo chequeará si es una violación, si es una violación se escribirá como datos decodificados cuatro 0L, caso contrario se

escribirá 1L. Una vez determinada la secuencia de datos decodificados, se procederá a graficar las señales de reloj, datos y codificada. Para realizar la decodificación se toma pasos similares a los considerados para la codificación B3ZS.

3.6.6 Rutina para el código Manchester Diferencial

a. Codificación Manchester Diferencial

La codificación Manchester Diferencial inicia chequeando si el dato ingresado corresponde a 1L o 0L, si es 1L se escribirá un nivel de señal positivo que implicará graficar la señal codificada con transición a la mitad del periodo de bit opuesta a la transición anterior, si es 0L se escribirá un nivel de señal negativo que implicará graficar la señal codificada con transición a la mitad del periodo de bit igual a la transición anterior. Una vez determinada la secuencia de niveles de señal, procederá a graficar las señales de reloj, datos y codificada como se muestra en la Figura 3.26.

b. Decodificación Manchester Diferencial

La decodificación de señales Manchester Diferencial se la realiza con las mismas consideraciones expuestas para la decodificación Bifase M.

3.6.7 Rutina para el código Miller

a. Codificación Miller

La codificación Miller inicia chequeando si el dato ingresado corresponde a 1L o 0L, si es 1L se escribirá un nivel de señal positivo que implicará graficar la señal codificada con transición a la mitad del periodo de bit, si es 0L se escribirá un nivel se señal negativo que implicará graficar la señal codificada con una transición al inicio del periodo de bit si el nivel de señal anterior fue negativo, caso contrario graficará la señal codificada sin transición al inicio del periodo de bit.

Una vez determinada la secuencia de niveles de señal, procederá a graficar las señales de reloj, datos y codificada como se muestra en la Figura 3.27.

b. Decodificación Miller

La decodificación de señales Miller se la realiza con las mismas consideraciones expuestas para la decodificación Bifase M.

3.6.8 Rutina para el código MLT-3

a. Codificación MLT-3

La codificación MLT-3 inicia chequeando si el dato ingresado corresponde a 1L o 0L, si es 1L se chequea la variable aux, si esta tiene el valor de 2 se escribirá un nivel se señal cero y se tomará las siguientes consideraciones: se almacenará en la variable c el valor de 2 que indicará al siguiente valor de 0L que mantenga un nivel de señal cero y almacenará en la variable aux el valor de 1 o 0 para indicar que el siguiente 1L le corresponde un nivel de señal positivo o un nivel de señal negativo, respectivamente. Si la variable aux tiene el valor de 1 se escribirá un nivel de señal positivo y se almacenará en la variable c el valor de 1 para indicar que el siguiente 0L mantenga un nivel de señal positivo y en la variable aux el valor de 0 para indicar que el siguiente 1L le corresponderá un nivel de señal cero. Si la variable aux tiene el valor de 0 se escribirá un nivel de señal negativo y se almacenará en la variable c el valor de 0 para indicar que el siguiente 0L mantenga un nivel de señal negativo y en la variable aux el valor de 0 para indicar que el siguiente 1L le corresponderá un nivel de señal cero.

Si el dato ingresado es 0L se chequeará la variable c para conocer que nivel de señal debe de mantener como se lo ha expuesto anteriormente. Una vez determinada la secuencia de niveles de señal, procederá a graficar las señales de reloj, datos y codificada como se muestra en la Figura 3.28.

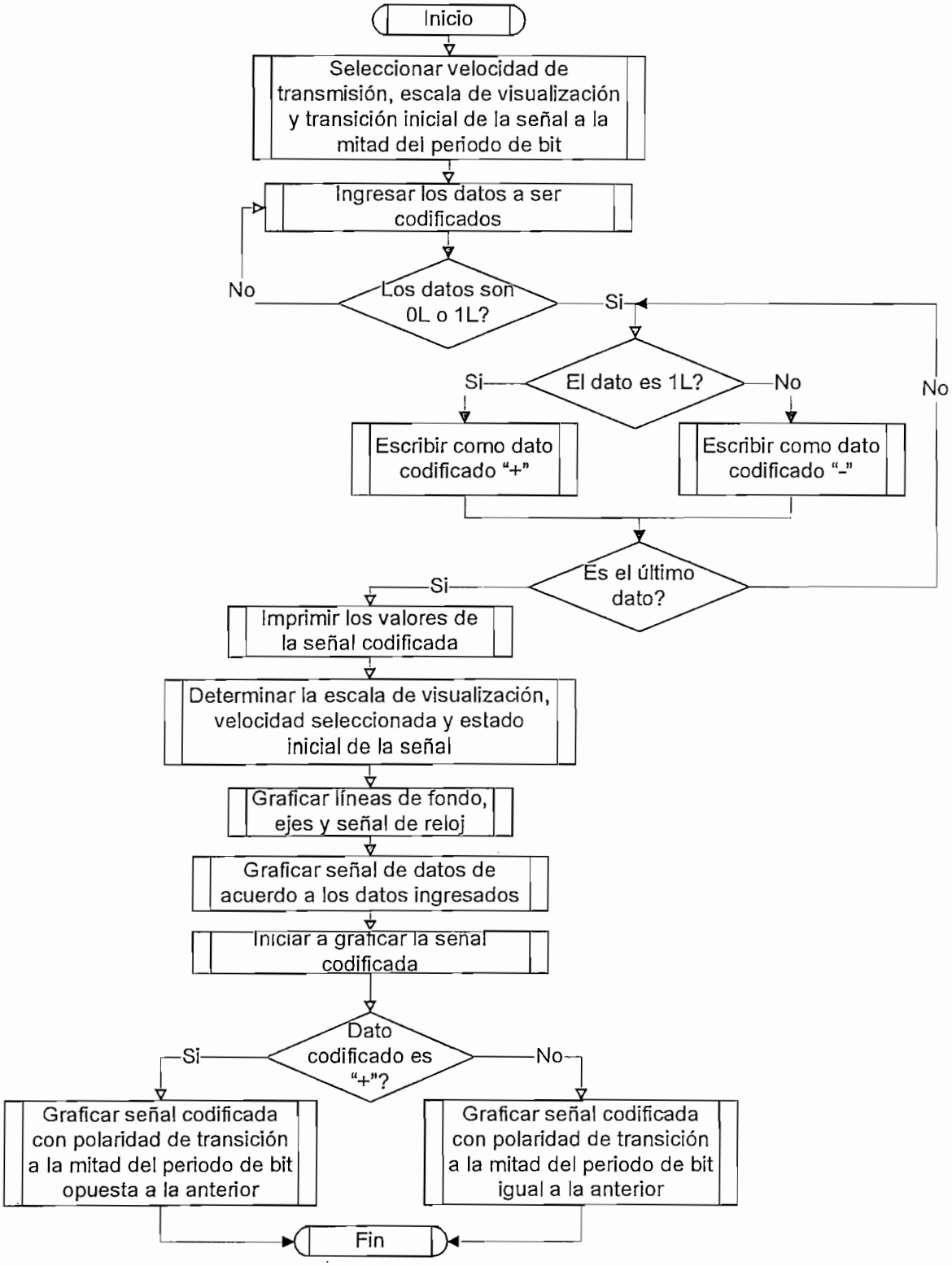


Figura 3.26 Diagrama de flujo de la rutina para la codificación Manchester Diferencial.

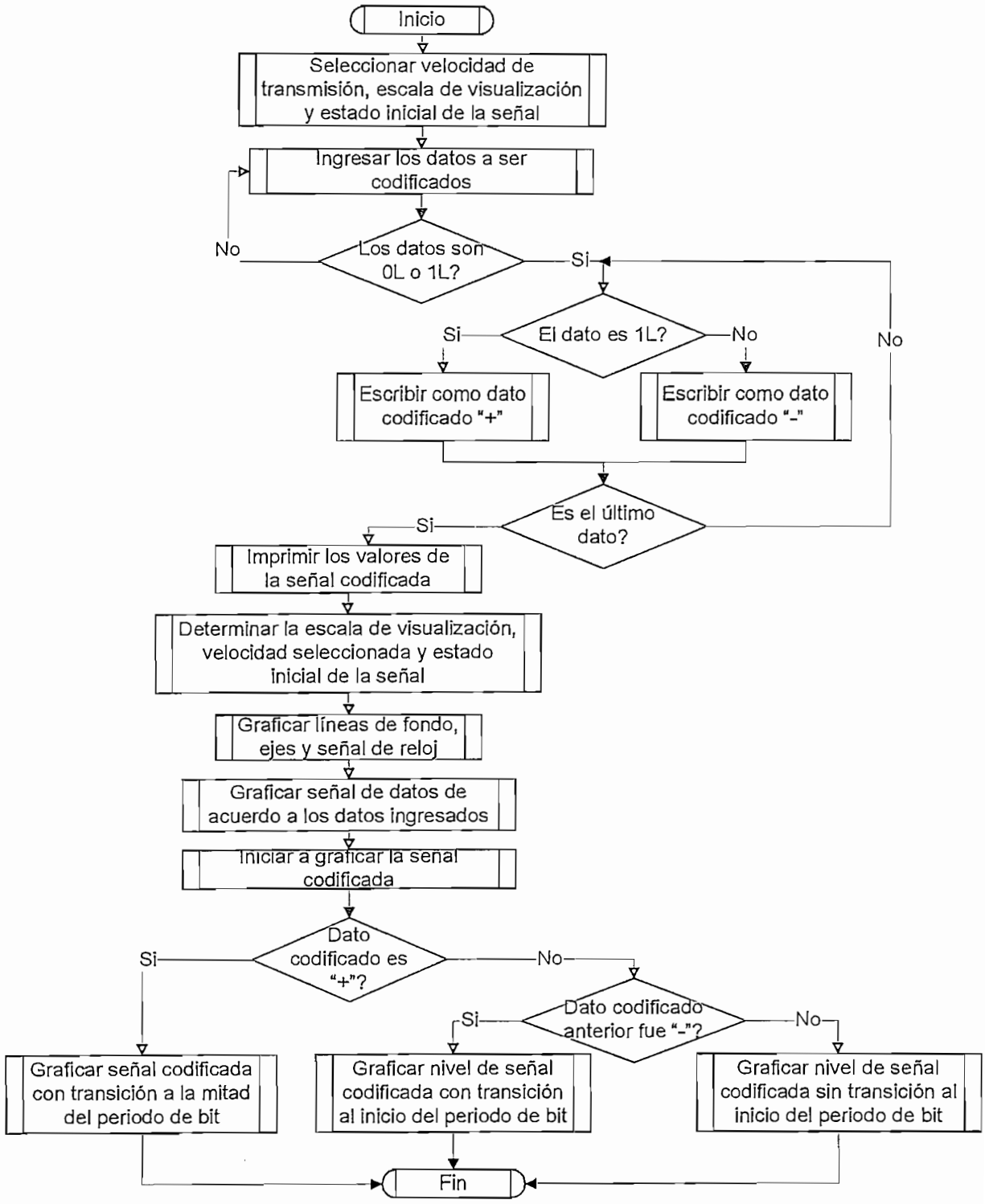


Figura 3.27 Diagrama de flujo de la rutina para la codificación Miller.

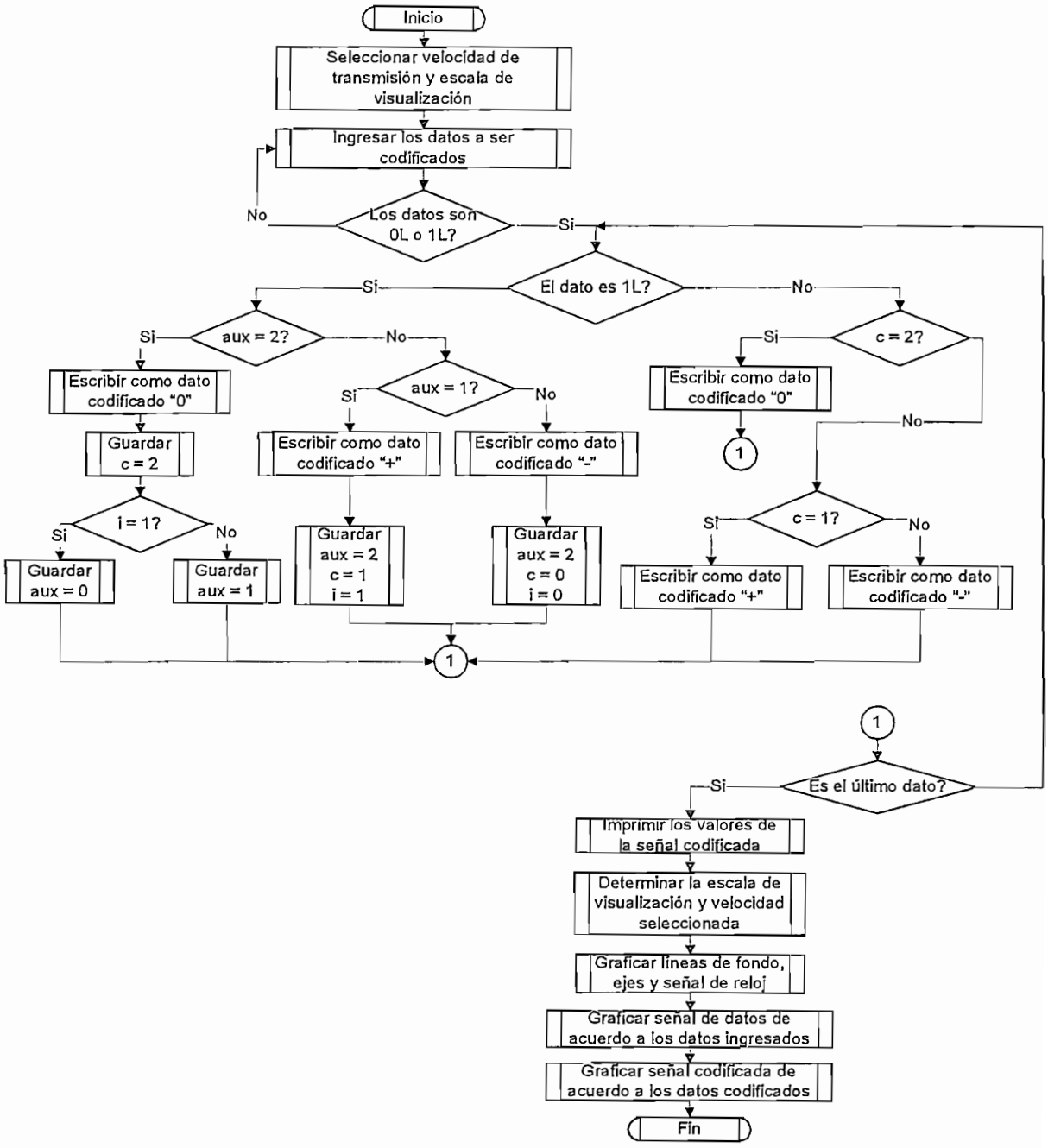


Figura 3.28 Diagrama de flujo de la rutina para la codificación MLT-3.

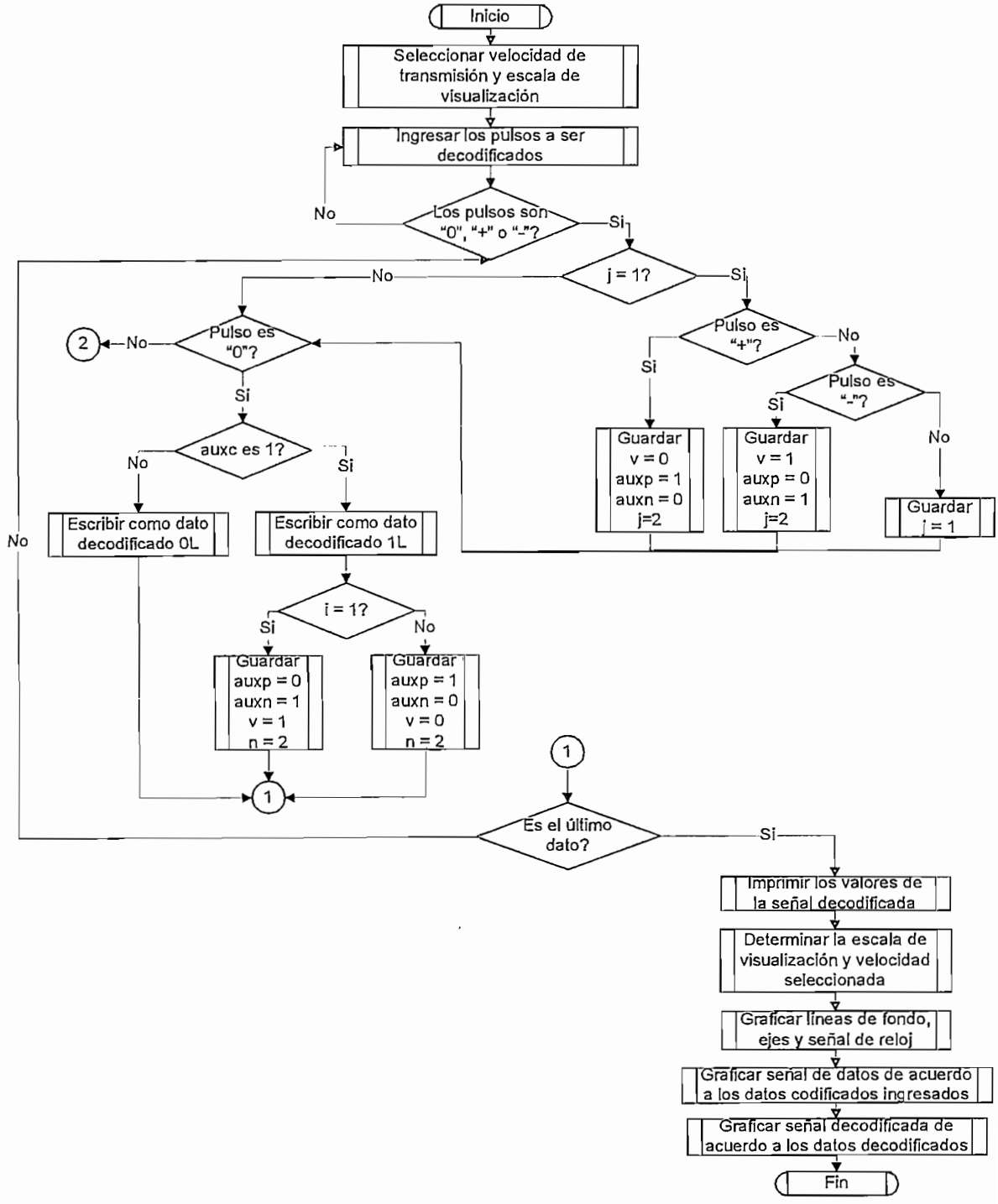
b. Decodificación MLT-3

La decodificación de señales MLT-3 inicia chequeando los niveles de señal ingresados, si el nivel de señal es positivo se almacenará en la variable v el valor de 0 para indicar que el siguiente nivel de señal después del nivel de señal cero deberá ser un nivel negativo, con lo cual nos ayudará a chequear que la

secuencia ingresada sea correcta; en la variable auxp se almacenará el valor de 1 y en la variable auxn se almacenará un valor de 0 para escribir como dato codificado 1L, luego se almacenará en la variable auxp el valor de 0, en la variable auxc el valor de 1 para indicar que el siguiente nivel de señal positivo representará un dato codificado 0L y en la variable n se almacenará el valor de 1 que nos ayudará a chequear que la secuencia de 1L codificados sea 0+0-0+.

Si el nivel de señal es negativo se almacenará en la variable v el valor de 1 para indicar que el siguiente nivel de señal después del nivel de señal cero deberá ser un nivel positivo, con lo cual nos ayudará a chequear que la secuencia ingresada sea correcta; en la variable auxp se almacenará el valor de 0 y en la variable auxn se almacenará un valor de 1 para escribir como dato codificado 1L, luego se almacenará en la variable auxn el valor de 0, en la variable auxc el valor de 1 para indicar que el siguiente nivel de señal negativo representará un dato codificado 0L y en la variable n se almacenará el valor de 0 que nos ayudará a chequear que la secuencia de 1L codificados sea 0+0-0+.

Si el nivel de señal es cero se chequeará el valor de la variable auxc, si tiene el valor de 0 representará un dato 0L, si tiene el valor de 1 representará un dato 1L y almacenará en la variable v el valor de 0 o 1 para indicar que el siguiente nivel de señal deberá ser un nivel negativo o positivo respectivamente, con lo cual nos ayudará a chequear que la secuencia ingresada sea correcta. En la variable auxp se almacenará el valor de 1 y en la variable auxn se almacenará un valor de 0 para indicar que el siguiente nivel de señal positivo representará un dato 1L, o en la variable auxp se almacenará el valor de 0 y en la variable auxn se almacenará un valor de 1 para indicar que el siguiente nivel de señal negativo representará un dato 1L y en la variable n se almacenará el valor de 2 que nos ayudará a chequear que la secuencia de 1L codificados sea 0+0-0+. La Figura 3.29 muestra los pasos para realizar la decodificación MLT-3.


Inicio
Seleccionar velocidad de transmisión y escala de visualización
Ingresar los pulsos a ser decodificados
Los pulsos son "0", "+" o "-"?
No
Si
j = 1?
No
Si
Pulso es "0"?
2
No
Si
Pulso es "+"?
No
Si
Pulso es "-"?
No
Si
auxc es 1?
No
Si
Guardar v = 0 auxp = 1 auxn = 0 j=2
Guardar v = 1 auxp = 0 auxn = 1 j=2
Guardar j = 1
Escribir como dato decodificado 0L
Escribir como dato decodificado 1L
No
i = 1?
Si
No
Guardar auxp = 0 auxn = 1 v = 1 n = 2
Guardar auxp = 1 auxn = 0 v = 0 n = 2
1
1
Es el último dato?
Si
Imprimir los valores de la señal decodificada
Determinar la escala de visualización y velocidad seleccionada
Graficar líneas de fondo, ejes y señal de reloj
Graficar señal de datos de acuerdo a los datos codificados ingresados
Graficar señal decodificada de acuerdo a los datos decodificados
Fin

(a)

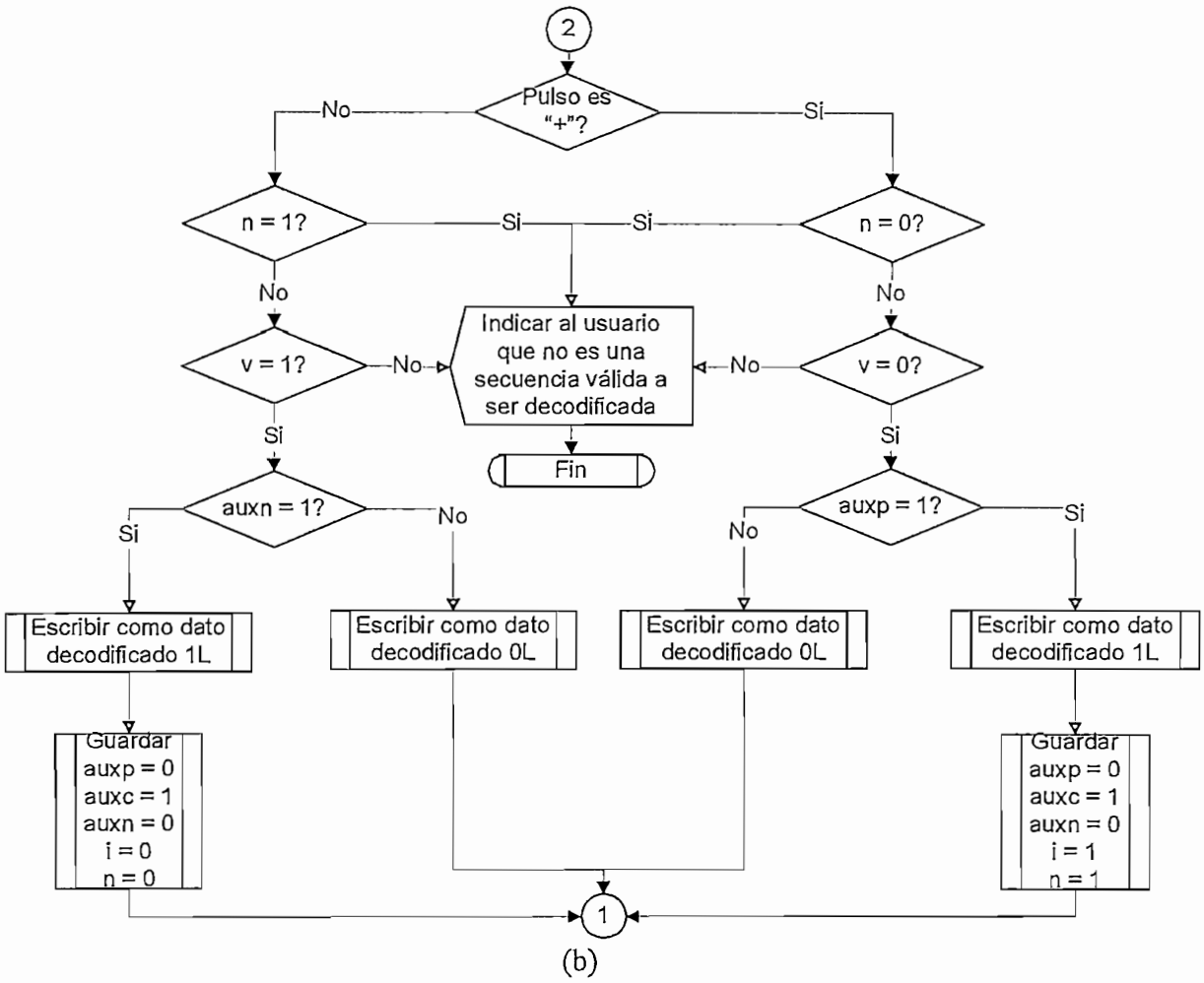


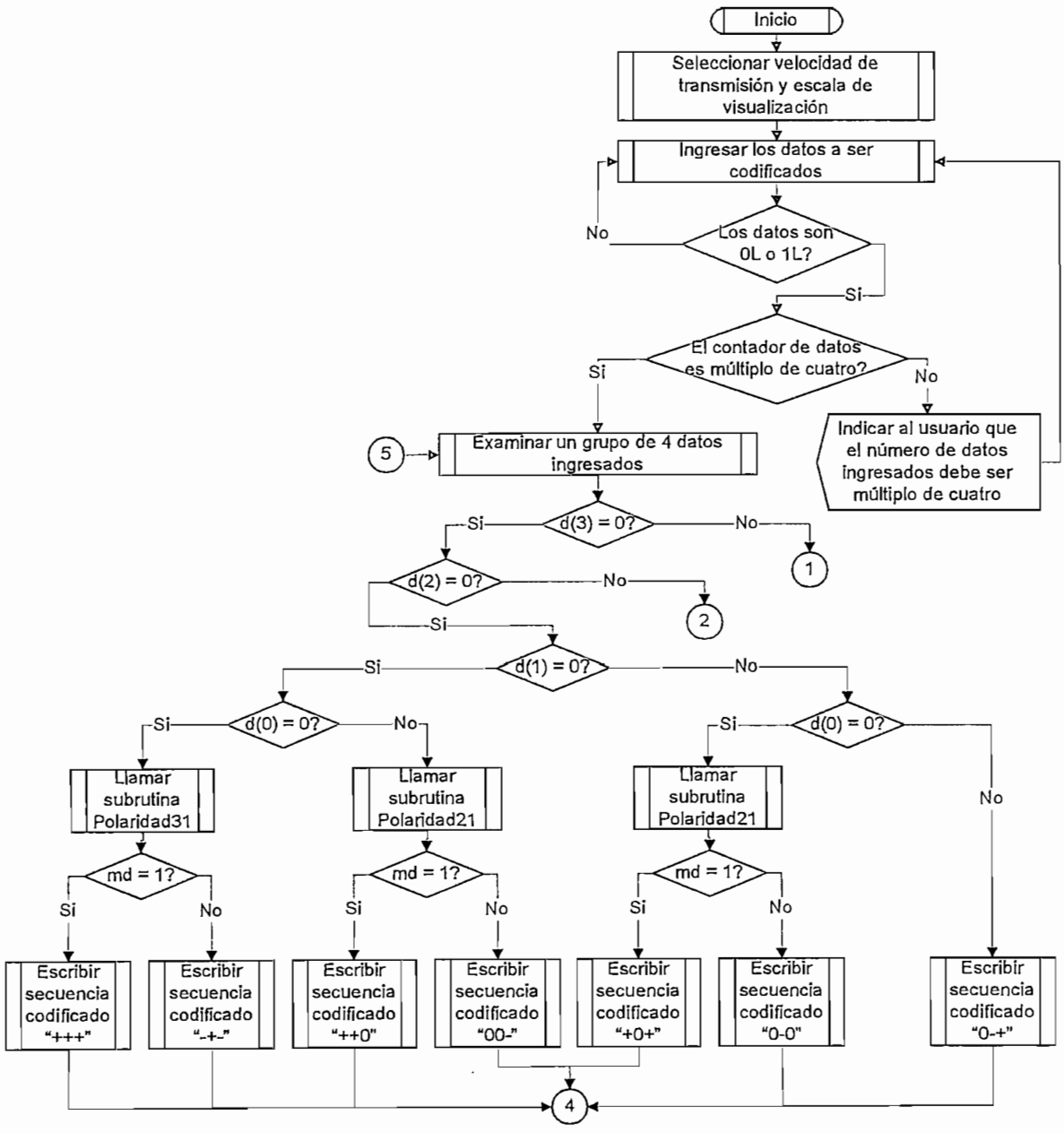
Figura 3.29 Diagrama de flujo de la rutina para la decodificación MLT-3.

3.6.9 Rutina para el código MS43

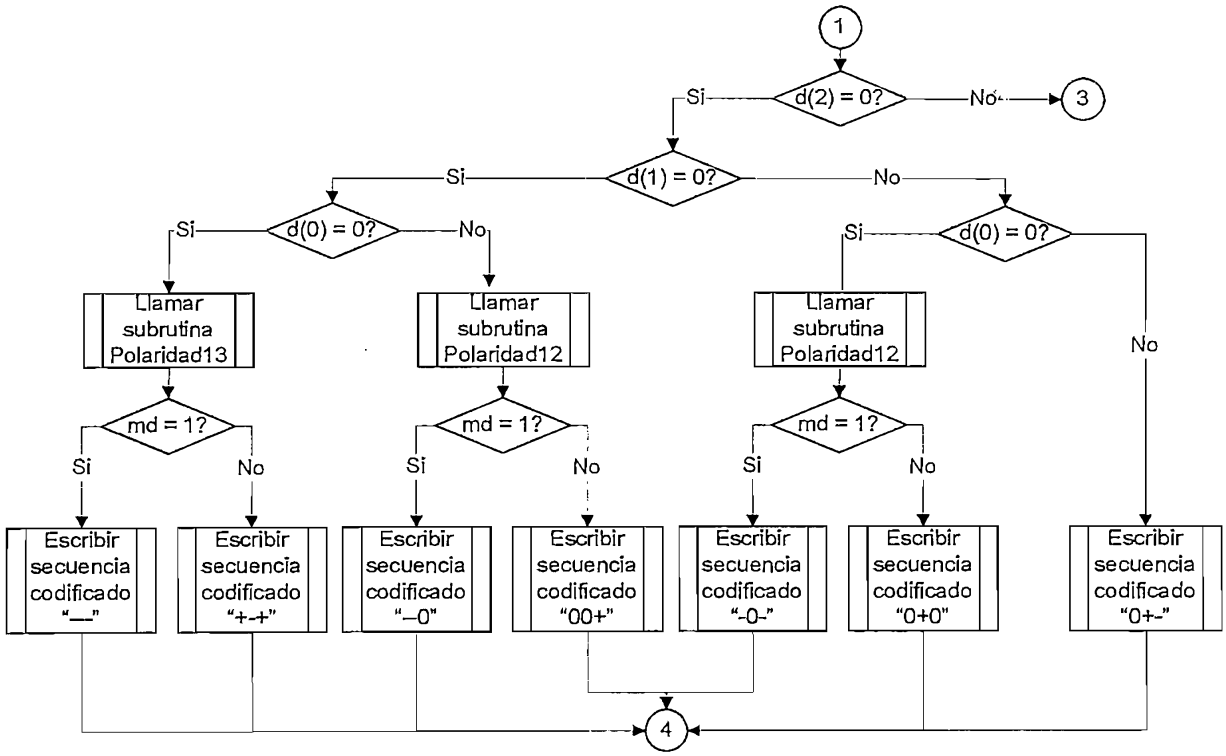
a. Codificación MS43

La codificación MS43 inicia chequeando si el número de datos ingresados es múltiplo de 4, luego chequea cuatro datos ingresados, iniciando por el más significativo hasta el menos significativo para determinar la entrada binaria, luego llamamos a la subrutina de polaridad correspondiente, para determinar la polaridad acumulada y el modo al cual corresponde la palabra ternaria. Una vez determinada la secuencia de niveles de señal, procederá a graficar las señales de reloj, datos y codificada. En la Figura 3.30 se presenta el diagrama de flujo de la rutina para la codificación MS43. La Figura 3.31 presenta el diagrama de flujo

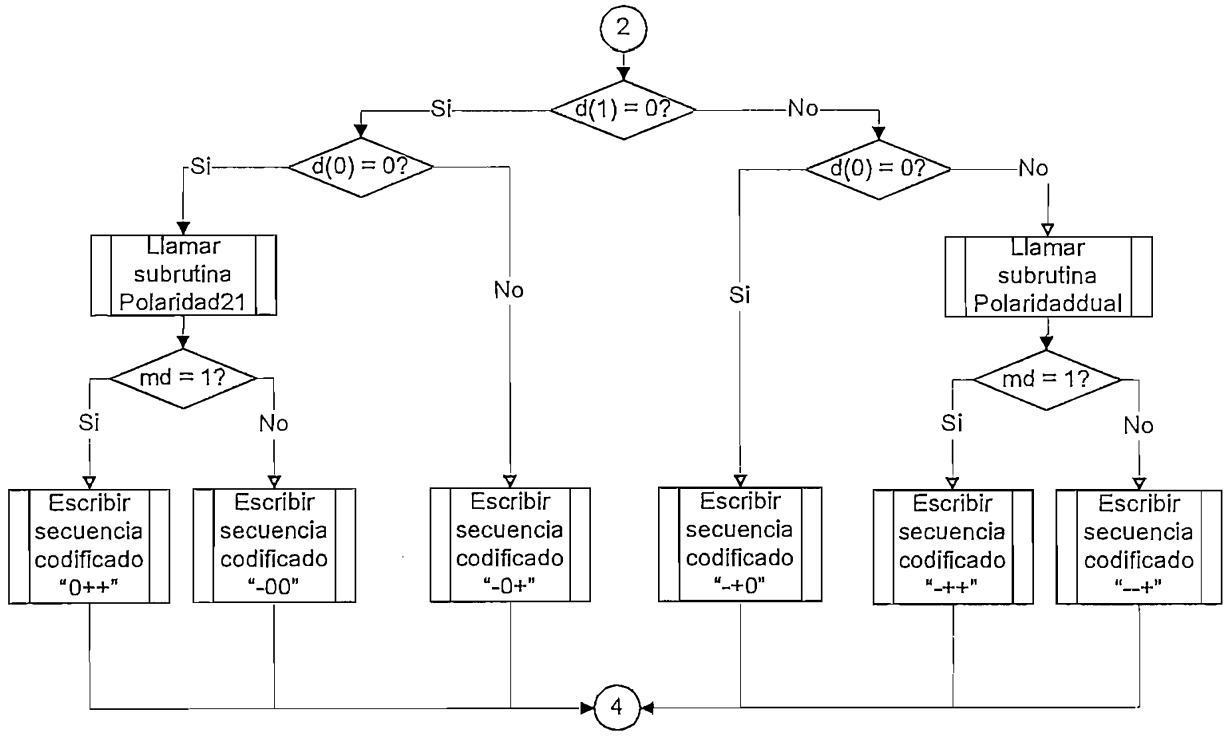
para determinar la polaridad acumulada y el modo al cual corresponde la palabra ternaria.



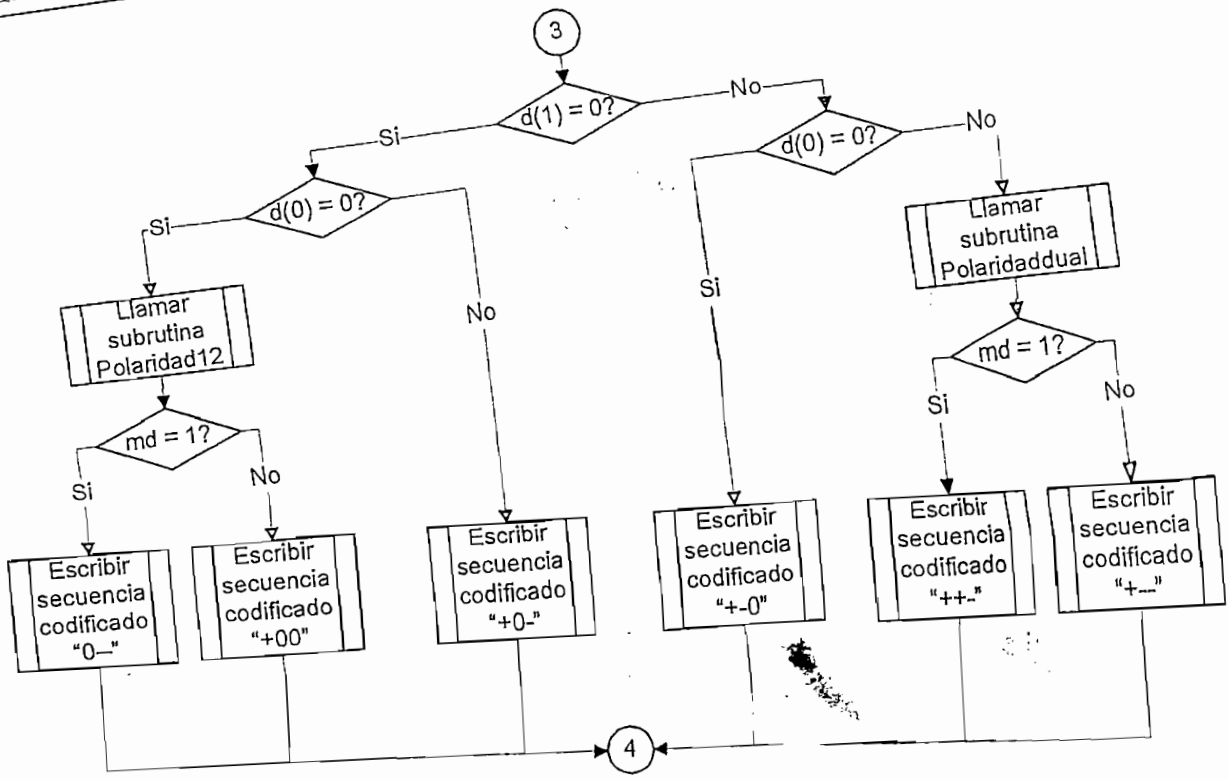
(a)


1
d(2) = 0?
Si
No
3
d(1) = 0?
Si
No
d(0) = 0?
Si
No
d(0) = 0?
Si
No
Llamar subrutina Polaridad13
Llamar subrutina Polaridad12
Llamar subrutina Polaridad12
md = 1?
md = 1?
md = 1?
Si
No
Escribir secuencia codificado "—"
Escribir secuencia codificado "+-+"
Escribir secuencia codificado "–0"
Escribir secuencia codificado "00+"
Escribir secuencia codificado "-0-"
Escribir secuencia codificado "0+0"
Escribir secuencia codificado "0+-"
4

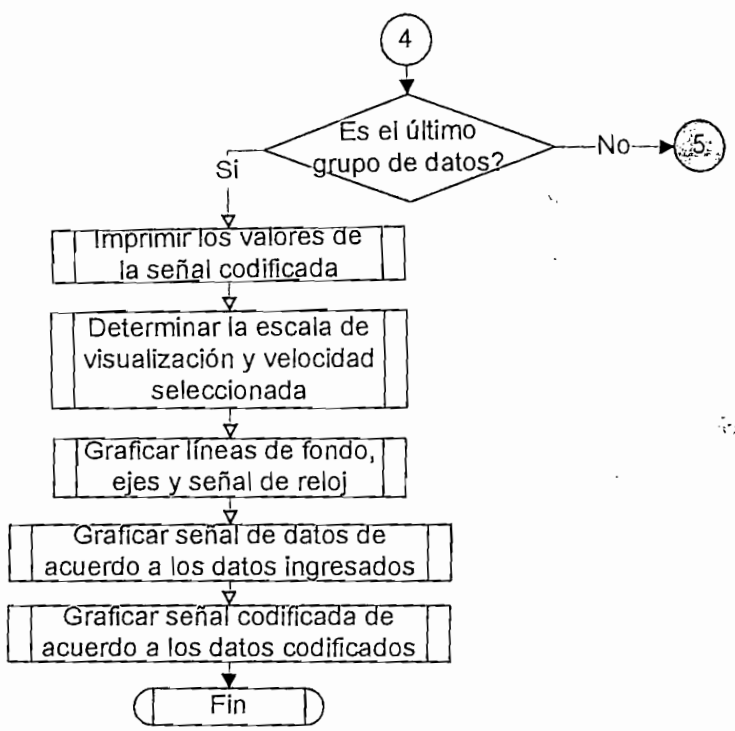
(b)


2
d(1) = 0?
Si
No
d(0) = 0?
Si
No
d(0) = 0?
Si
No
Llamar subrutina Polaridad21
Llamar subrutina Polaridaddual
md = 1?
md = 1?
Si
No
Escribir secuencia codificado "0++"
Escribir secuencia codificado "-00"
Escribir secuencia codificado "-0+"
Escribir secuencia codificado "-+0"
Escribir secuencia codificado "-++"
Escribir secuencia codificado "--+"
4

(c)

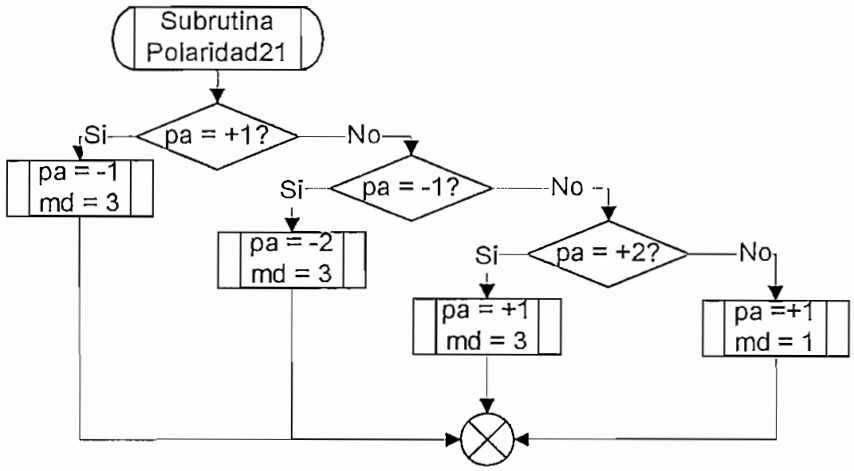


(d)

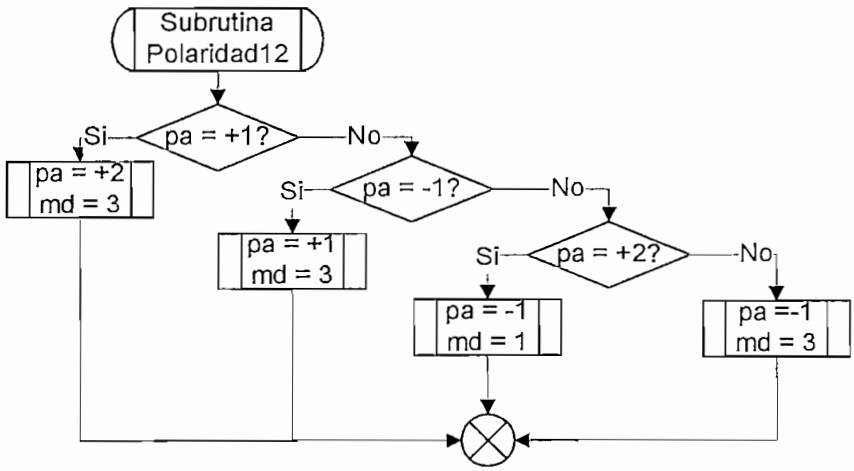


(e)

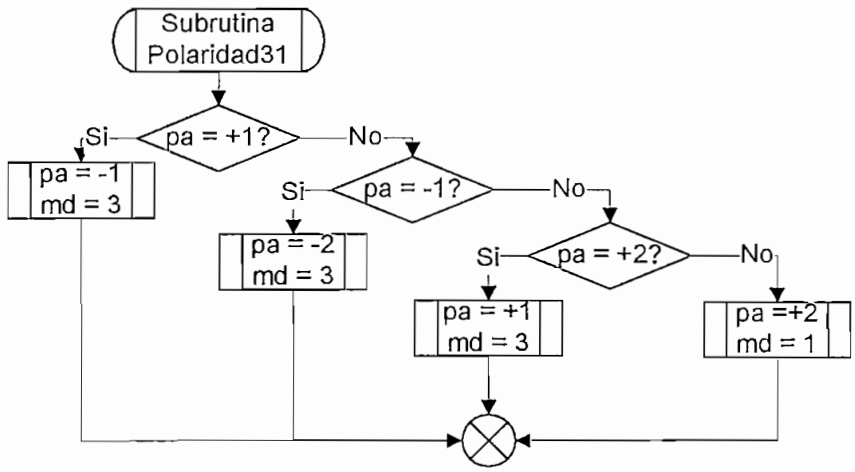
Figura 3.30 Diagrama de flujo de la rutina para la codificación MS43.


Subrutina
Polaridad21
pa = +1?
Si
No
pa = -1
md = 3
pa = -1?
Si
No
pa = -2
md = 3
pa = +2?
Si
No
pa = +1
md = 3
pa =+1
md = 1

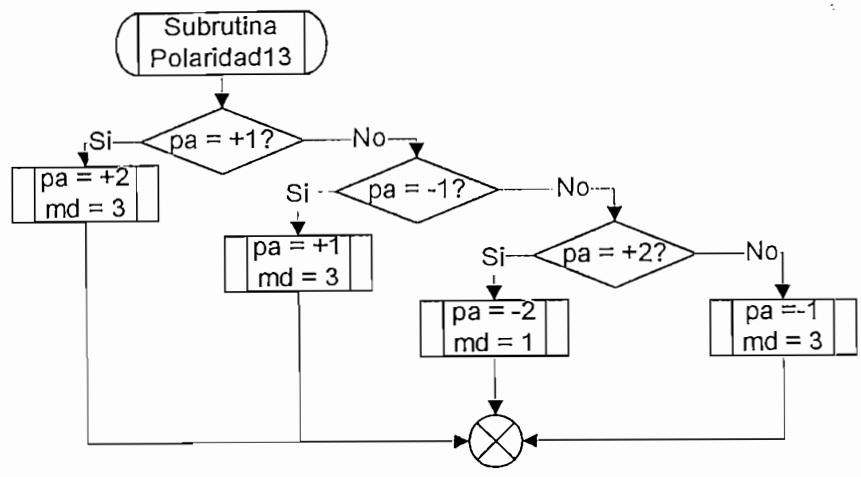
(a)


Subrutina
Polaridad12
pa = +1?
Si
No
pa = +2
md = 3
pa = -1?
Si
No
pa = +1
md = 3
pa = +2?
Si
No
pa = -1
md = 1
pa =-1
md = 3

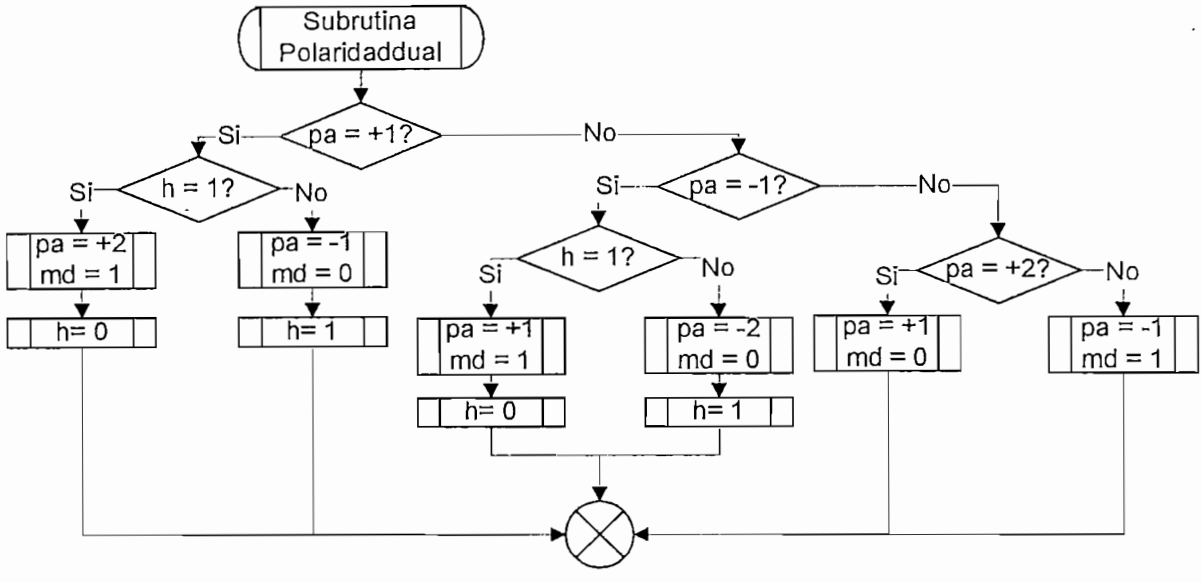
(b)


Subrutina
Polaridad31
pa = +1?
Si
No
pa = -1
md = 3
pa = -1?
Si
No
pa = -2
md = 3
pa = +2?
Si
No
pa = +1
md = 3
pa =+2
md = 1

(c)



(d)



(e)

Figura 3.31 Diagrama de flujo de las subrutinas para la codificación MS43.

b. Decodificación MS43

La decodificación de señales MS43 inicia chequeando si el número de niveles de señal ingresados es múltiplo de 3, luego chequea tres niveles de señal ingresados, iniciando por el más significativo hasta el menos significativo para determinar la palabra ternaria y qué entrada binaria le corresponde. Una vez determinada la secuencia de niveles de señal, procederá a graficar las señales de reloj, datos y codificada como se muestra en la Figura 3.32.

3.6.10 Rutina para el código NRZ Polar

a. Codificación NRZ Polar

La codificación NRZ Polar inicia chequeando si el dato ingresado corresponde a 1L o 0L, si es 0L escribirá un nivel de señal negativo, si el dato es 1L escribirá un nivel de señal positivo. Una vez determinada la secuencia de niveles de señal, procederá a graficar las señales de reloj, datos y codificada como se muestra en la Figura 3.33.

b. Decodificación NRZ Polar

La decodificación de señales NRZ Polar se la realiza con las mismas consideraciones expuestas para la decodificación Bifase M.

3.6.11 Rutina para el código RZ Polar

a. Codificación RZ Polar

La codificación de señales RZ Polar es similar a la codificación de señales NRZ Polar, con la diferencia que un nivel de señal positivo implicará graficar la señal codificada con una transición a la mitad del periodo de bit.

b. Decodificación RZ Polar

La decodificación de señales RZ Polar se la realiza con las mismas consideraciones expuestas para la decodificación Bifase M.

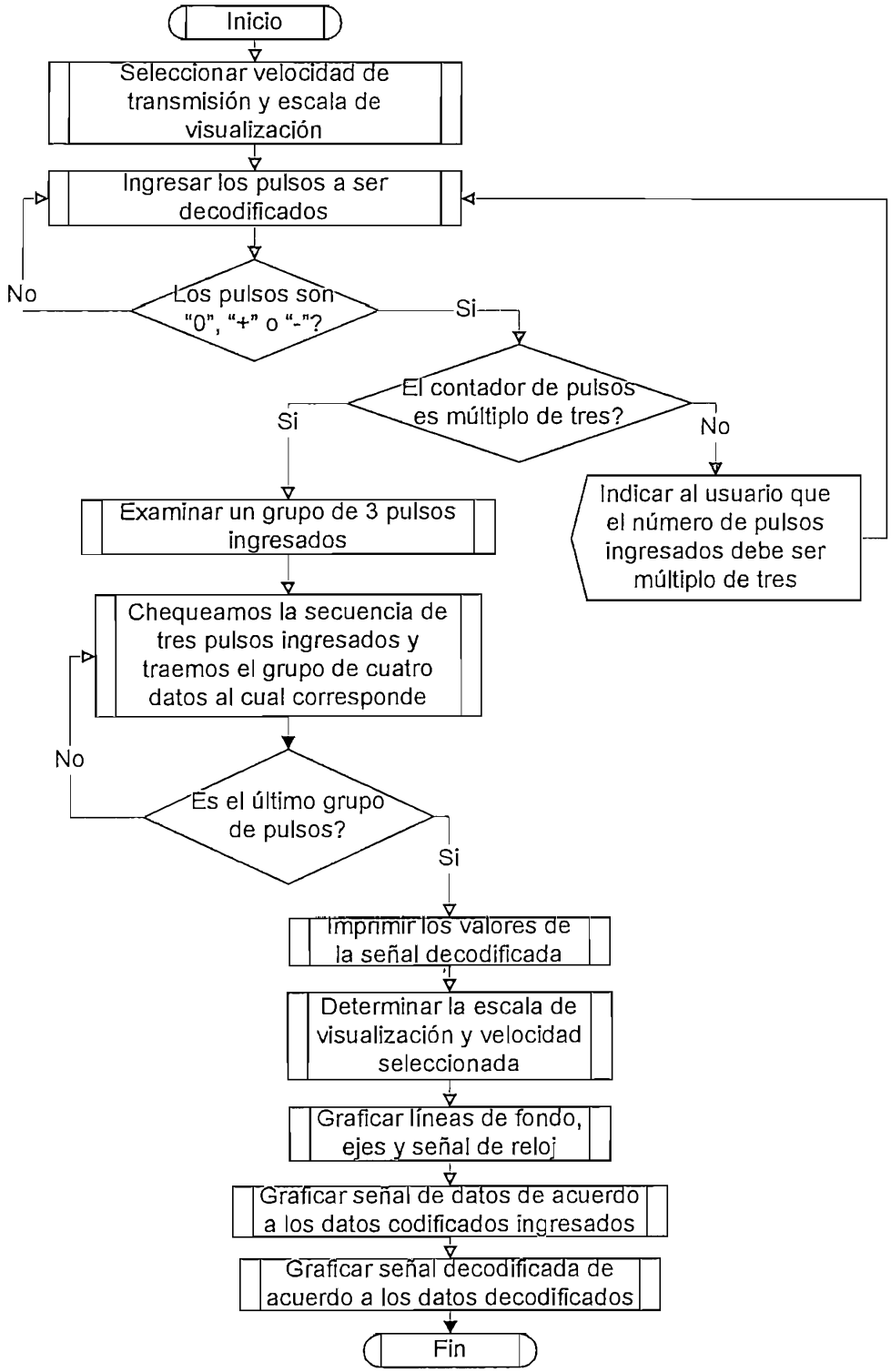


Figura 3.32 Diagrama de flujo de la rutina para la decodificación MS43.

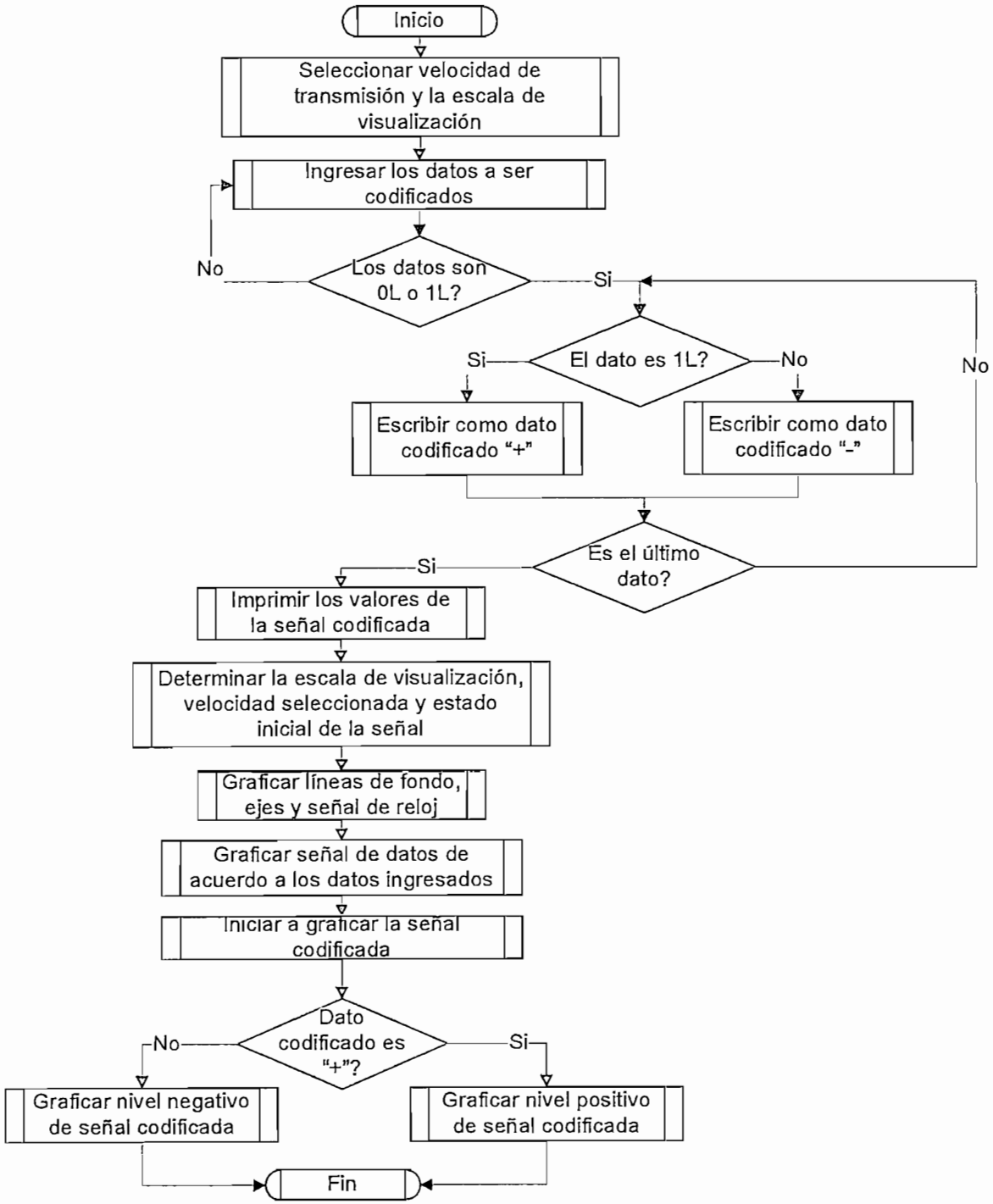


Figura 3.33 Diagrama de flujo de la rutina para la codificación NRZ Polar.

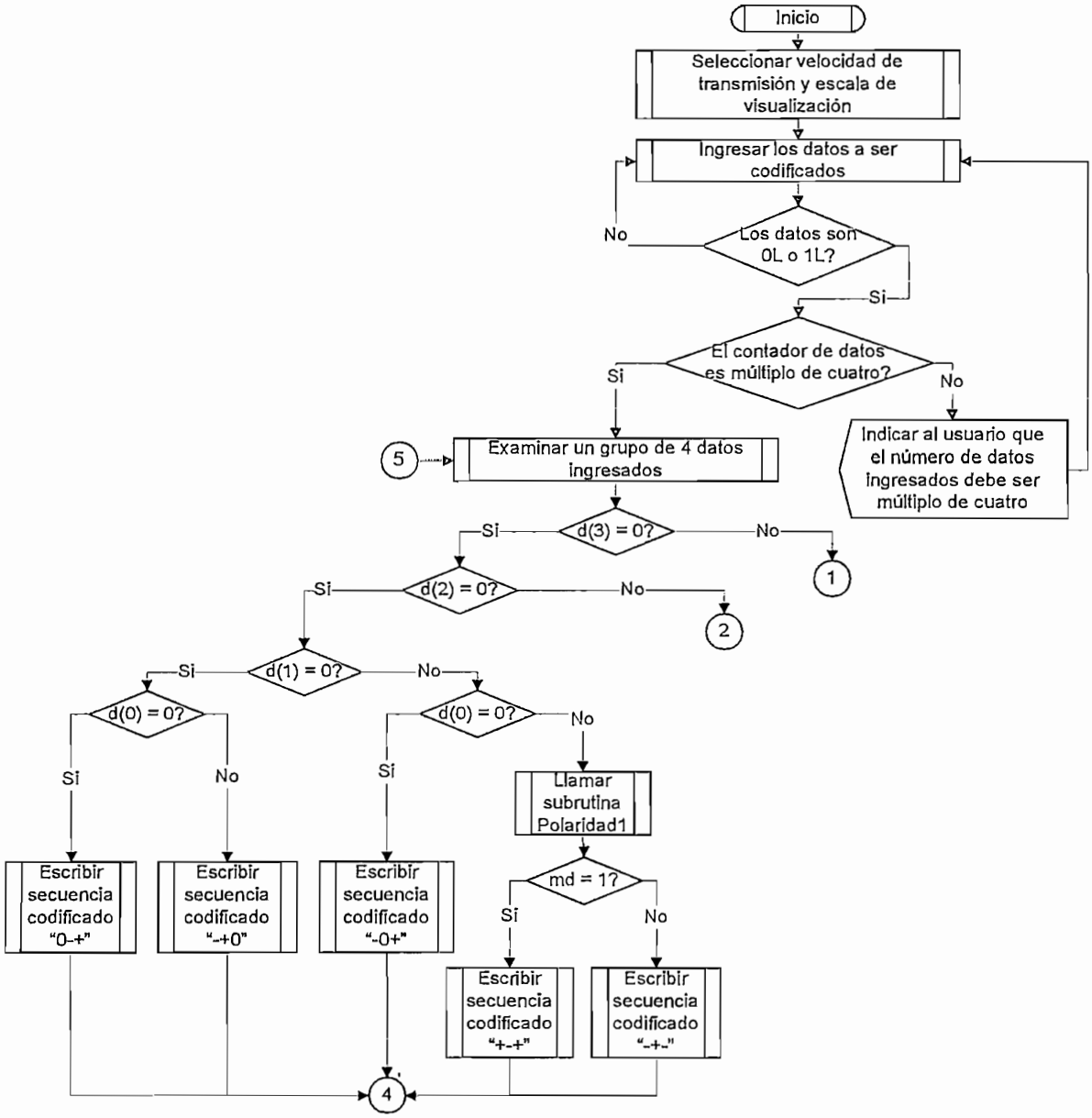
3.6.12 Rutina para el código 4B3T

a. Codificación 4B3T

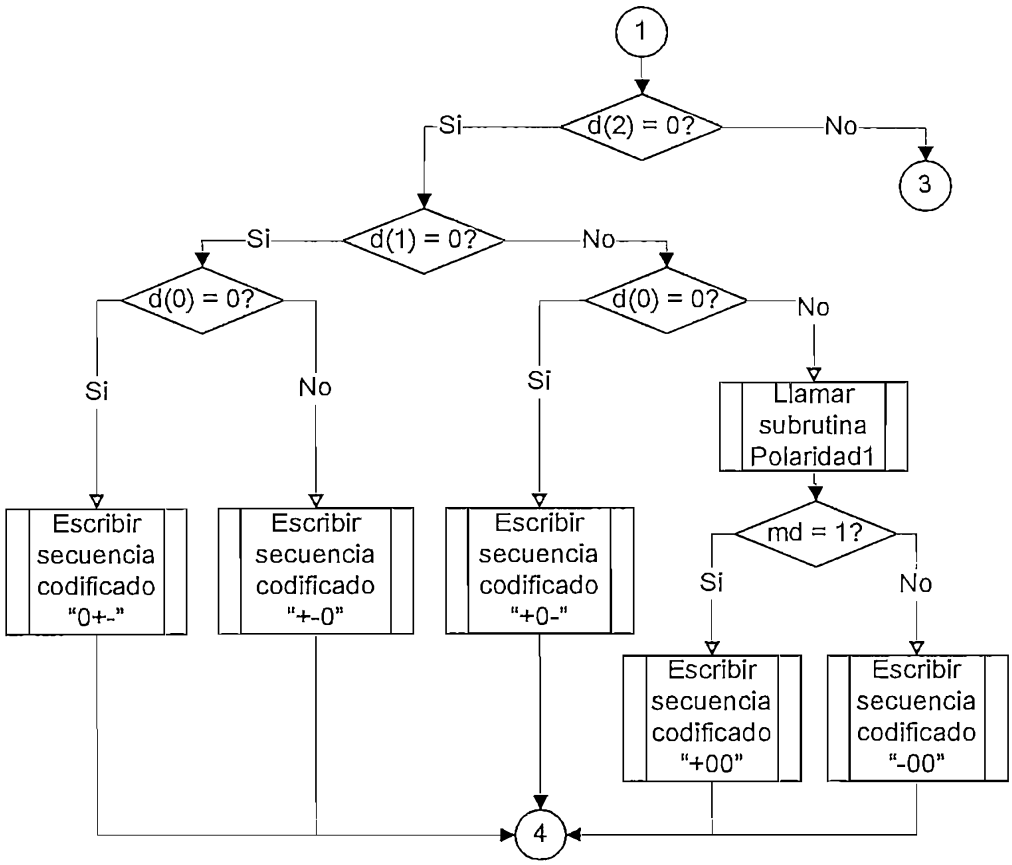
La codificación 4B3T inicia chequeando si el número de datos ingresados es múltiplo de 4, luego chequea cuatro datos ingresados, iniciando por el más significativo hasta el menos significativo para determinar la entrada binaria, luego llamamos a la subrutina de polaridad correspondiente, para determinar la polaridad acumulada y si la palabra ternaria tiene modo positivo o negativo. Una vez determinada la secuencia de niveles de señal, procederá a graficar las señales de reloj, datos y codificada. En la Figura 3.34 se presenta el diagrama de flujo de la rutina para la codificación 4B3T. La Figura 3.35 presenta el diagrama de flujo para determinar la polaridad acumulada y el modo al cual corresponde la palabra ternaria.

b. Decodificación 4B3T

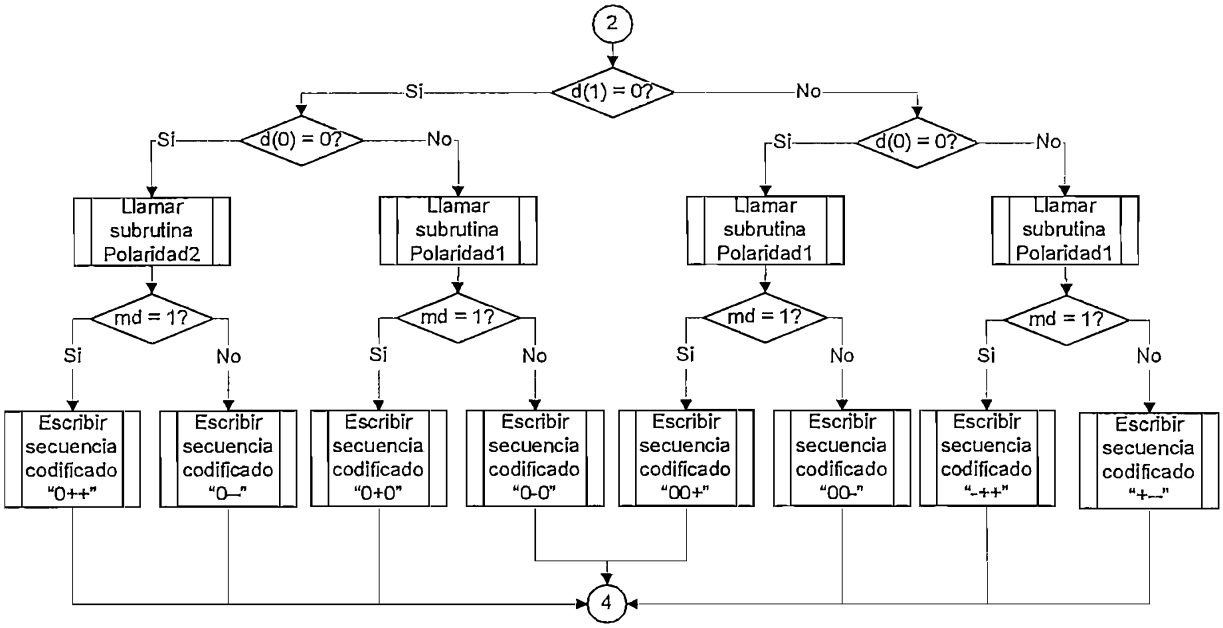
La decodificación de señales 4B3T se la realiza con las mismas consideraciones expuestas para la decodificación MS43.


Inicio
Seleccionar velocidad de transmisión y escala de visualización
Ingresar los datos a ser codificados
Los datos son 0L o 1L?
No
Si
El contador de datos es múltiplo de cuatro?
Si
No
Indicar al usuario que el número de datos ingresados debe ser múltiplo de cuatro
5
Examinar un grupo de 4 datos ingresados
d(3) = 0?
Si
No
1
d(2) = 0?
Si
No
2
d(1) = 0?
Si
No
d(0) = 0?
Si
No
d(0) = 0?
Si
No
Escribir secuencia codificado "0-+"
Escribir secuencia codificado "-+0"
Escribir secuencia codificado "-0+"
Llamar subrutina Polaridad1
md = 1?
Si
No
Escribir secuencia codificado "+-+"
Escribir secuencia codificado "-+-"
4

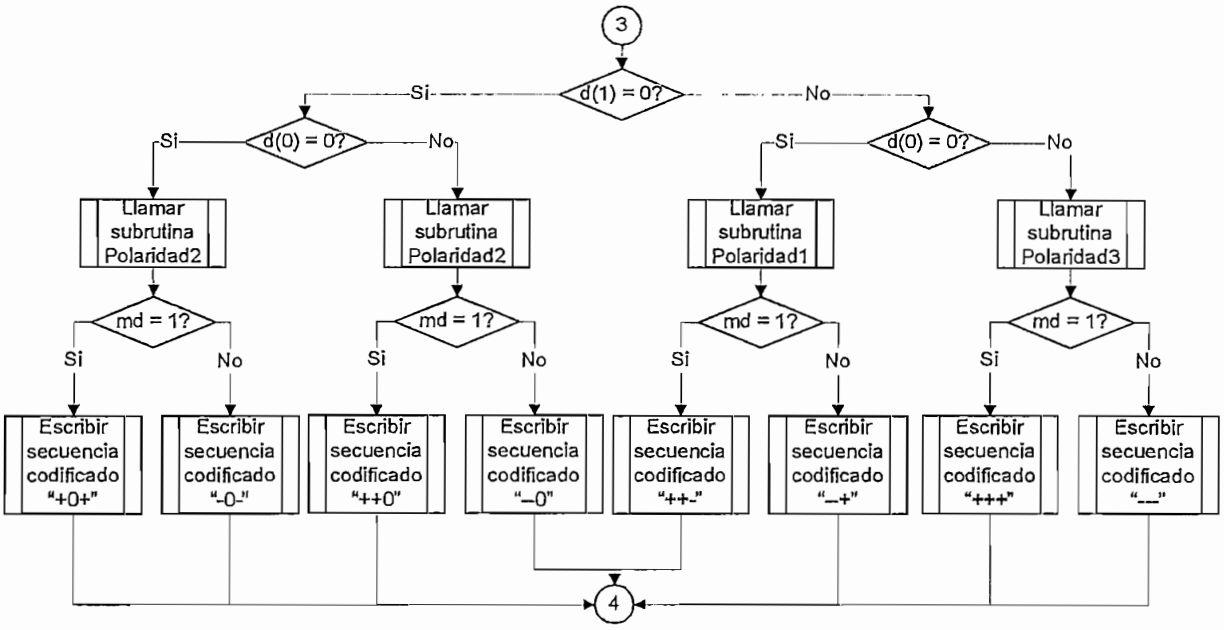
(a)


1
d(2) = 0?
Si
No
3
d(1) = 0?
Si
No
d(0) = 0?
Si
No
d(0) = 0?
Si
No
Llamar subrutina Polaridad1
Escribir secuencia codificado "0+-"
Escribir secuencia codificado "+-0"
Escribir secuencia codificado "+0-"
md = 1?
Si
No
Escribir secuencia codificado "+00"
Escribir secuencia codificado "-00"
4

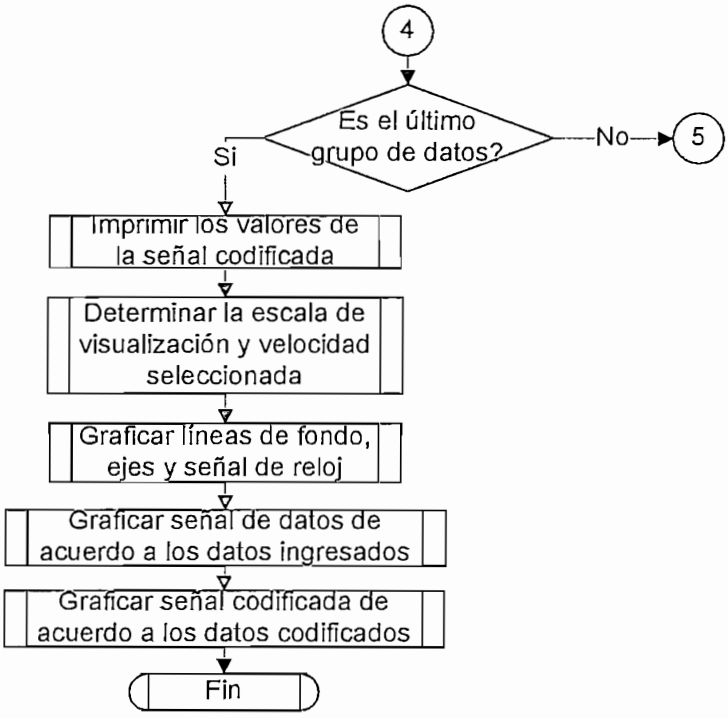
(b)


2
d(1) = 0?
Si
No
d(0) = 0?
Si
No
d(0) = 0?
Si
No
Llamar subrutina Polaridad2
Llamar subrutina Polaridad1
Llamar subrutina Polaridad1
Llamar subrutina Polaridad1
md = 1?
Si
No
Escribir secuencia codificado "0++"
Escribir secuencia codificado "0–"
Escribir secuencia codificado "0+0"
Escribir secuencia codificado "0-0"
Escribir secuencia codificado "00+"
Escribir secuencia codificado "00-"
Escribir secuencia codificado "-++"
Escribir secuencia codificado "+–"
4

(c)

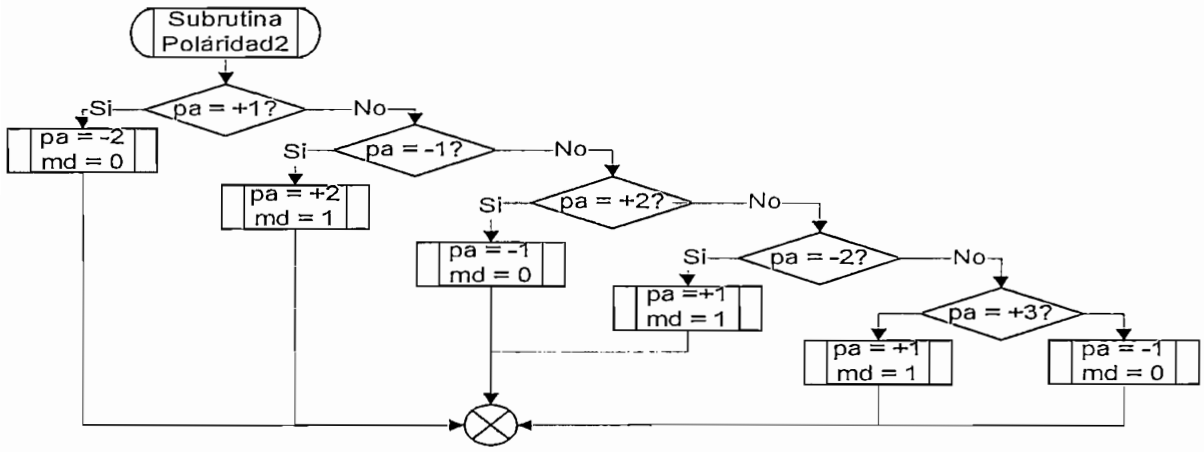


(d)

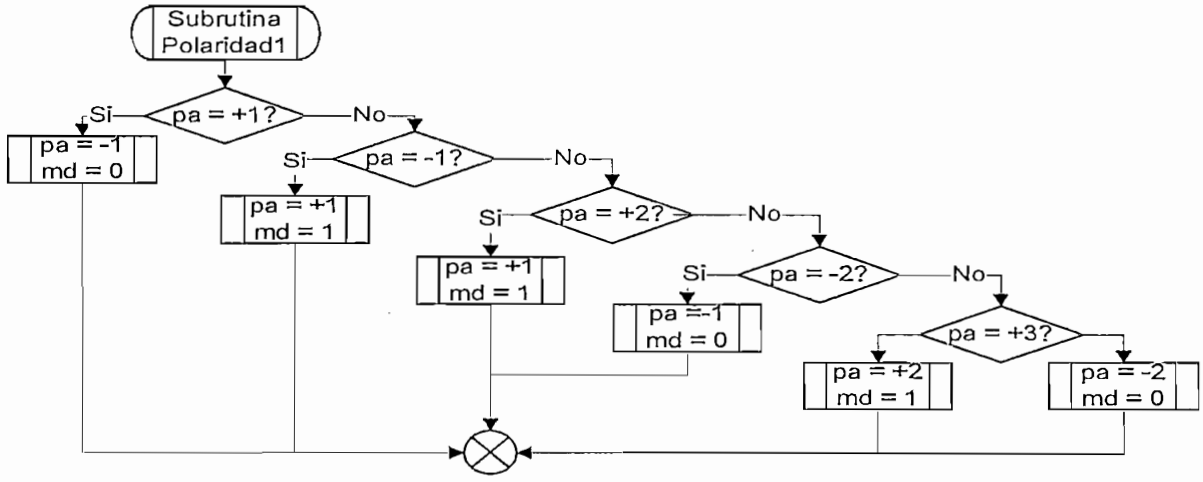


(e)

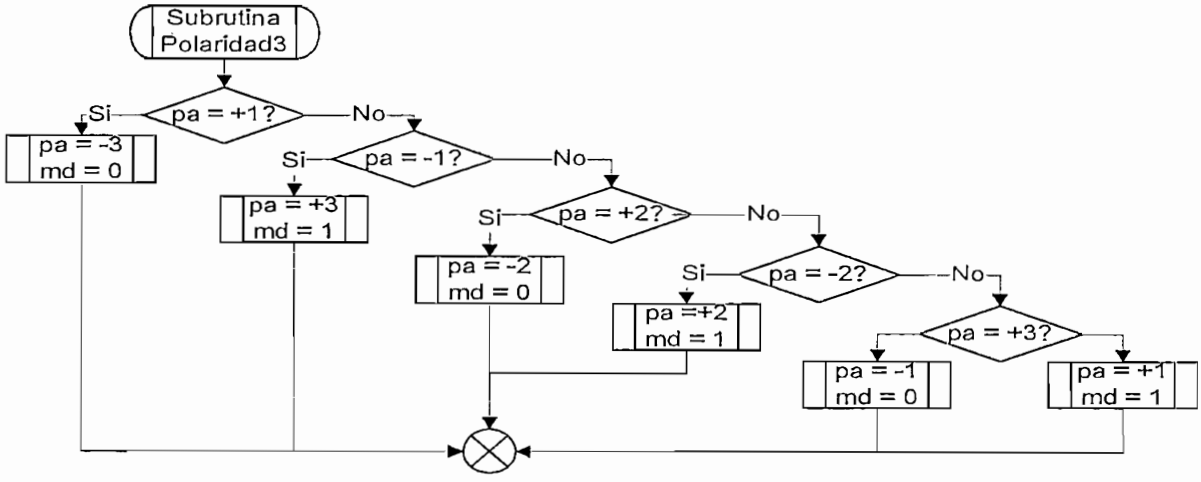
Figura 3.34 Diagrama de flujo de la rutina para la codificación 4B3T.



(a)



(b)



(c)

Figura 3.35 Diagrama de flujo de las subrutinas para la codificación 4B3T

CAPÍTULO IV

EVALUACIÓN DEL CODIFICADOR DECODIFICADOR DE CÓDIGOS DE LÍNEA

EVALUACIÓN DEL CODIFICADOR / DECODIFICADOR DE CÓDIGOS DE LÍNEA

Luego de haber descrito detalladamente cada uno de los elementos utilizados en el diseño del codificador / decodificador de códigos de línea, así como las herramientas visuales utilizadas en el desarrollo del software que permite la interacción del PC con el dispositivo; en este capítulo se muestra cada uno de los elementos de los que está constituido externamente el CODEC, así como las pruebas realizadas al dispositivo.

4.1 DESCRIPCIÓN DE LA PARTE EXTERNA DEL DISPOSITIVO

En la parte frontal se puede observar los indicadores implementados en el dispositivo y que se mencionan a continuación:

- Indicador de Encendido (*ON/OFF*)
- Indicador de nivel TTL
- Indicador de Nivel RS-232

El indicador de Encendido está constituido por un "Led Rojo", el cual se enciende siempre y cuando el dispositivo esté conectado al PC mediante el interfaz USB, el cual es el que provee la alimentación de energía al CODEC.

Los indicadores de niveles de señal binaria RS-232 o TTL se encenderán dependiendo de la selección del usuario, el "Led Verde" indica que se ha realizado la selección del nivel TTL y el "Led Naranja" indica que se ha elegido el nivel RS-232, dicho nivel ha sido elegido por defecto por los diseñadores. Cabe indicar que la selección del tipo de nivel de señal se la hace a través de la PC, previamente instalado el programa Codec Portátil. En la figura 4.1 se muestra la parte frontal del dispositivo.

Además de poseer en la parte frontal los indicadores; el dispositivo dispone de conectores, los cuales permitirán obtener las siguientes señales.

Etapa de Codificación:

- Señal de entrada de datos RS-232.
- Señal de entrada de datos TTL.
- Señal Codificada.
- Señal de reloj de Codificación.
- Señal de reloj de Decodificación de salida.
- Tierra (*GND*).

Etapa de Decodificación:

- Señal de datos a decodificar.
- Señal de datos decodificados RS – 232.
- Señal de datos decodificados TTL.
- Señal de reloj de Decodificación de entrada.
- Tierra (*GND*).

La Figura 4.1 muestra cada una de los terminales que permitirán la codificación y decodificación.

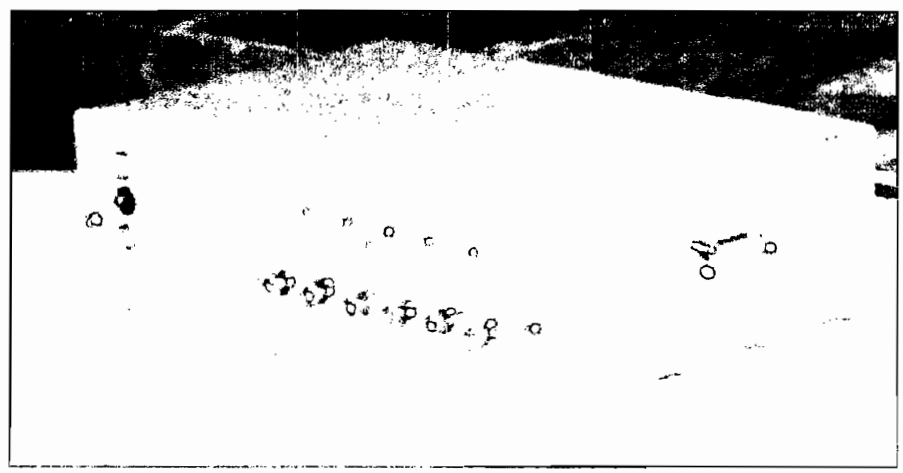


Figura 4.1 Vista frontal del dispositivo.

Como se puede ver se tiene un Terminal de señal de reloj de Decodificación de entrada y un Terminal de señal de reloj de Decodificación de salida, esto se debe

a que el CODEC es sincrónico y necesita de la señal de reloj para decodificar la señal en el destino.

Observando la parte trasera del dispositivo, se puede ver que posee dos conectores:

- Conector DB-9 hembra.
- Conector hembra USB tipo B.

El conector DB-9 hembra permite la transmisión y recepción de datos seriales que provengan del PC. Los datos pueden ser enviados utilizando la herramienta visual que se encuentra incorporada en el programa Codec Portátil o utilizando el *HyperTerminal*.

El conector hembra USB tipo B permite la transmisión de datos para la configuración del dispositivo, los mismos que son: el código de línea, velocidad de transmisión y nivel de señal de entrada de datos. Además este conector permite la transmisión de datos para mantener la conexión del CODEC con la PC En la Figura 4.2 se puede ver la disposición de los conectores.

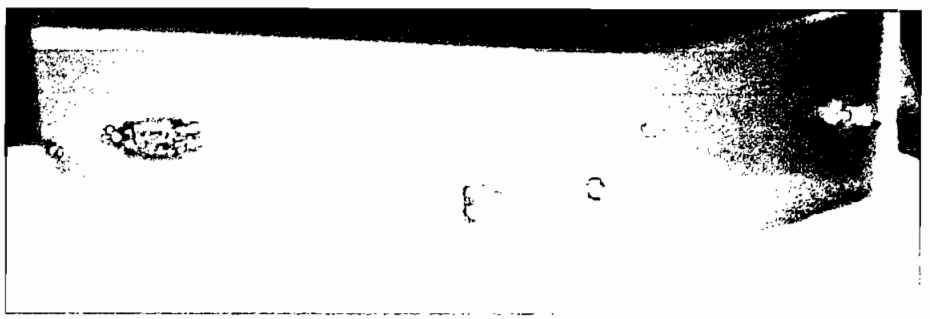


Figura 4.2 Vista Trasera del dispositivo.

Como accesorios, el CODEC dispone de un cable USB con los terminales Tipo A y Tipo B en cada uno de sus extremos, además de un cable serial con sus terminales DB – 9 macho.

4.2 HERRAMIENTAS UTILIZADAS EN LAS PRUEBAS DEL DISPOSITIVO.

Para verificar el funcionamiento del dispositivo se ha realizado las pruebas a dos velocidades diferentes, a las velocidades límites a las cuales puede trabajar el dispositivo; dependiendo del código, los límites de la velocidad de transmisión variarán.

Para tomar los valores de las señales de prueba, se utiliza el osciloscopio digital en tiempo real *Tektronix* TDS 220, éste posee dos canales con un ancho de banda de 100 MHz; posee un módulo de expansión para comunicaciones con puertos RS–232, GPIB y *Centronics* para realizar la adquisición de las señales capturadas en el osciloscopio.

La adquisición de las señales de prueba, capturadas en el osciloscopio, se ha realizado configurando el puerto serial a una velocidad de transmisión de 4800 bits/seg., 8 bits de datos, sin paridad y sin control de flujo, mientras que en la PC se configuran los mismos parámetros utilizando el programa *Tektronix Instrument Manager Service.* En la Figura 4.3 se muestra la pantalla de configuración de los parámetros de comunicación.

Una vez establecidos los parámetros de configuración de la comunicación se procede a realizar la prueba de establecimiento de la conexión haciendo click en el botón *Test New Settings* que se muestra en la Figura 4.3. Una vez que se ha conseguido la comunicación entre el computador y el osciloscopio aparecerá el mensaje que se muestra en la Figura 4.4.

Una vez establecida la comunicación con el osciloscopio, la adquisición de las imágenes capturadas en el osciloscopio se realiza utilizando el programa *WaveStar for oscilloscopes* versión 2.2. En la Figura 4.5 se muestra la pantalla principal del programa.

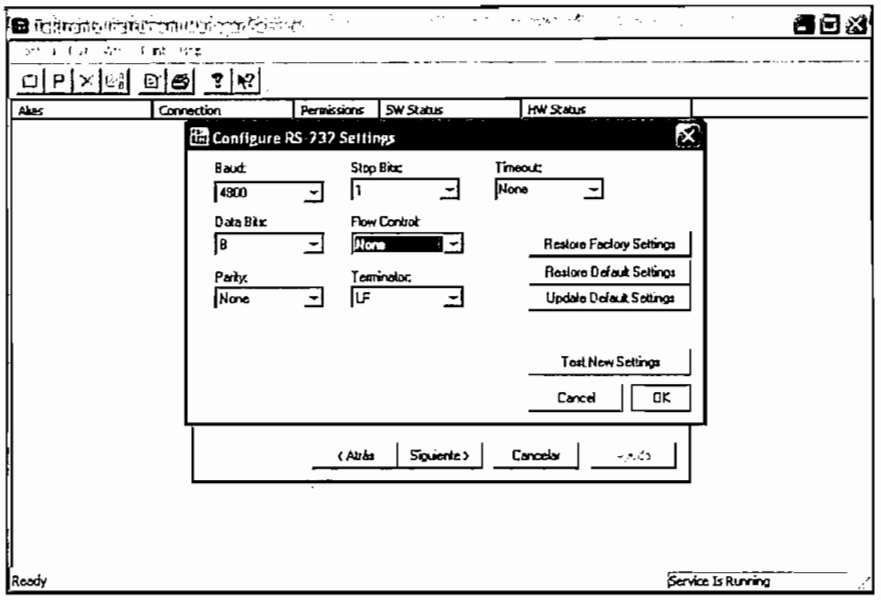


Figura 4.3 Configuración de los parámetros de comunicación del osciloscopio *Tektronix* TDS 220.

Figura 4.4 Comunicación entre la PC y el osciloscopio *Tektronix* TDS 220.

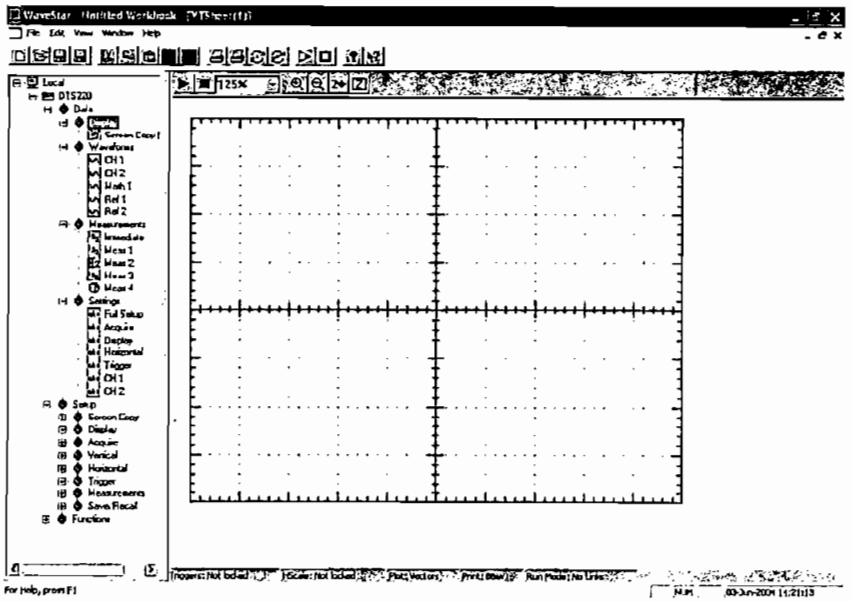


Figura 4.5 Pantalla principal del programa *WaveStar for oscilloscopes* versión 2.2.

Con la ayuda de este programa se ha capturado la señal de datos a codificar y la señal de datos codificados con la finalidad de verificar la correcta operación del CODEC a las velocidades límite. Se ha establecido con el color Azul (CANAL 1) a la señal de datos decodificada y con el color Rojo (CANAL 2) a la señal de datos a codificar.

4.3 PRUEBAS DEL FUNCIONAMIENTO DE LOS CÓDIGOS DE LÍNEA

Para la realización de las pruebas se ha utilizado uno de los dos CODEC, a éste CODEC se lo ha realimentado, es decir, recibe los datos a codificar de la PC, los codifica y a continuación nuevamente recibe la señal codificada para decodificarla y transmitirla hacia la PC.

Para realizar el análisis de las señales que se presentan en cada una de las pruebas, se ha designado los siguientes valores para representar los niveles de voltaje de las señales codificadas:

- +A representa el voltaje de +5V.
- - A representa el voltaje de -5V.
- 0 representa el voltaje de 0V.

Esta representación permitirá el mejor entendimiento de cada una de las codificaciones.

4.3.1 CÓDIGO AMI

El primer código en ser probado es el código AMI. En la Figura 4.6 se muestra la señal de codificación y la señal de datos a codificar a la velocidad de 300 bits/seg para la letra "u" cuyo valor en binario es 1110101. En la Figura 4.6 se ha codificado la secuencia de datos binaria 01011101 con la secuencia 0, +A, 0, -A, +A, -A, 0, +A.

En la Figura 4.7 la señal de datos a codificar y la señal codificada se encuentran a la velocidad de 19200 y la letra que se ha escogido para la prueba es la letra "u". Los valores que se encuentran codificados en la Figura 4.7 son 1011101 y los valores de la señal codificada que se muestran son +A, 0, -A, +A, -A, 0, +A.

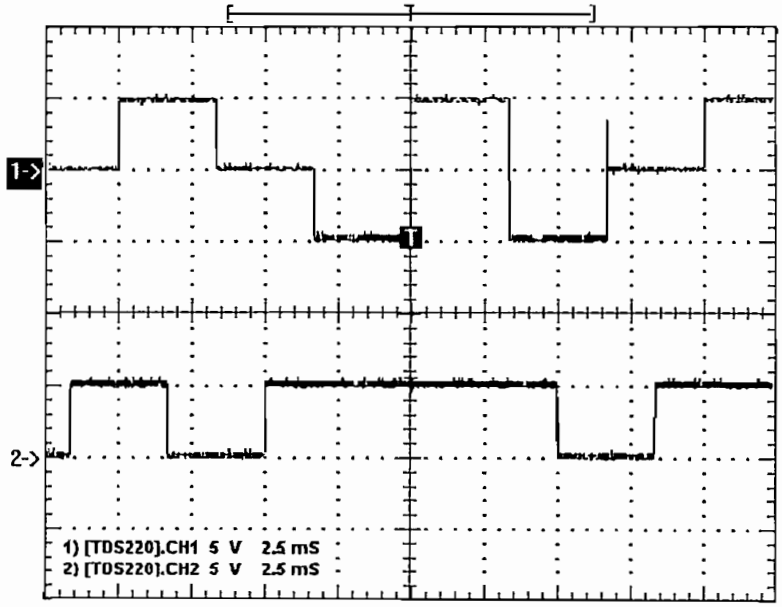


Figura 4.6 Codificación AMI de la letra "u" a 300 bits/seg.

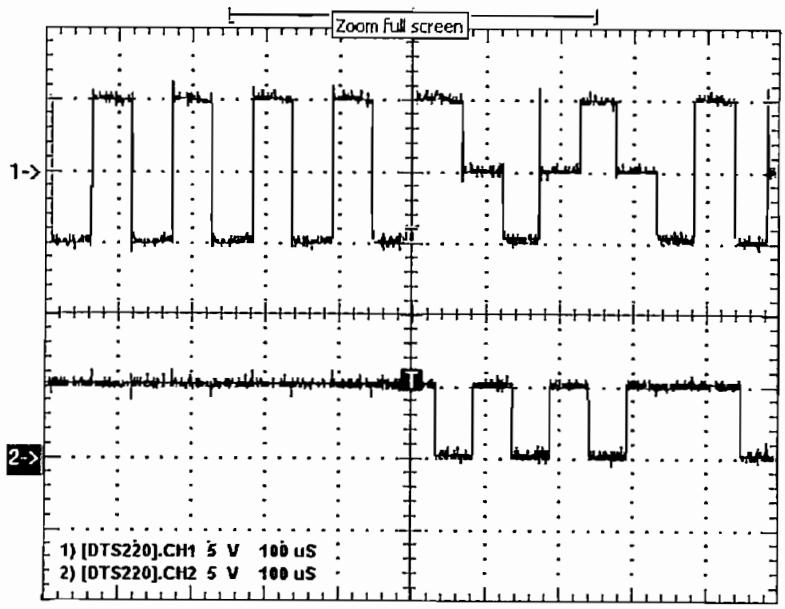


Figura 4.7 Codificación AMI de la letra "u" a 19200 bits/seg.

Cabe indicar que en las Figuras 4.6 y 4.7 se puede observar que existe un retardo de aproximadamente medio periodo de bit que presenta la señal codificada con

respecto a la señal de datos a codificar, él cual debe ser considerado para el análisis.

4.3.2 CÓDIGO NRZ POLAR

La señal de datos a codificar y la señal codificada para el código NRZ Polar a la velocidad de 300 bits/seg para la letra "U" (01010101 valor binario) se muestra en la Figura 4.8. La secuencia binaria que aparece codificada en la gráfica es 01010101010101 con la secuencia -A, +A, -A, +A, -A, +A, -A, +A, +A, -A, +A, -A, +A, -A.

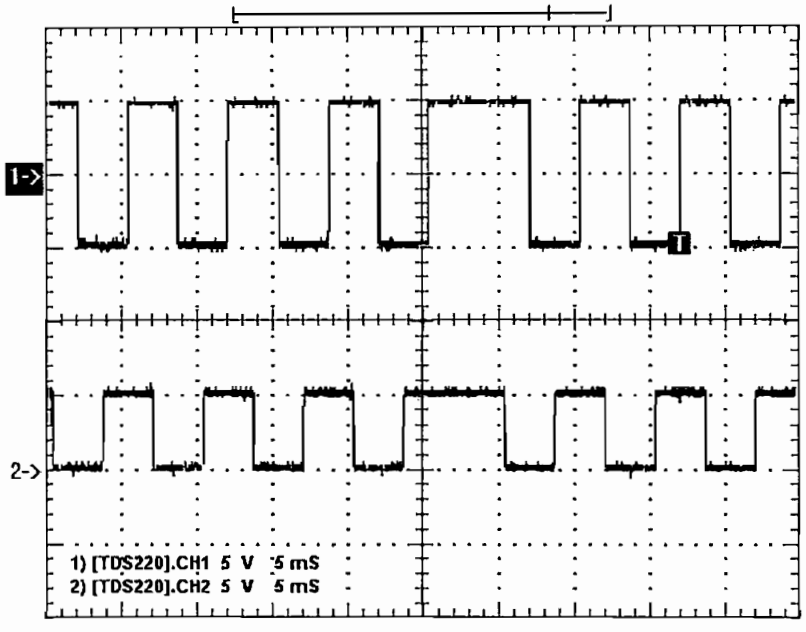


Figura 4.8 Codificación NRZ Polar de la letra "U" a 300 bits/seg.

La codificación NRZ Polar a 19200 bits/seg para la letra "U" se muestra en la Figura 4.9. La secuencia binaria a codificar es 10101010101 y su señal codificada presenta los siguientes valores +A, -A, +A, -A, +A, -A, +A, -A, N+A, -A, +A.

La razón de usar la letra "U" en esta codificación es porque esta nos permite observar los cambios de nivel que deben darse al momento de tener unos y ceros alternados y la letra "U" es perfecta para cumplir con este objetivo.

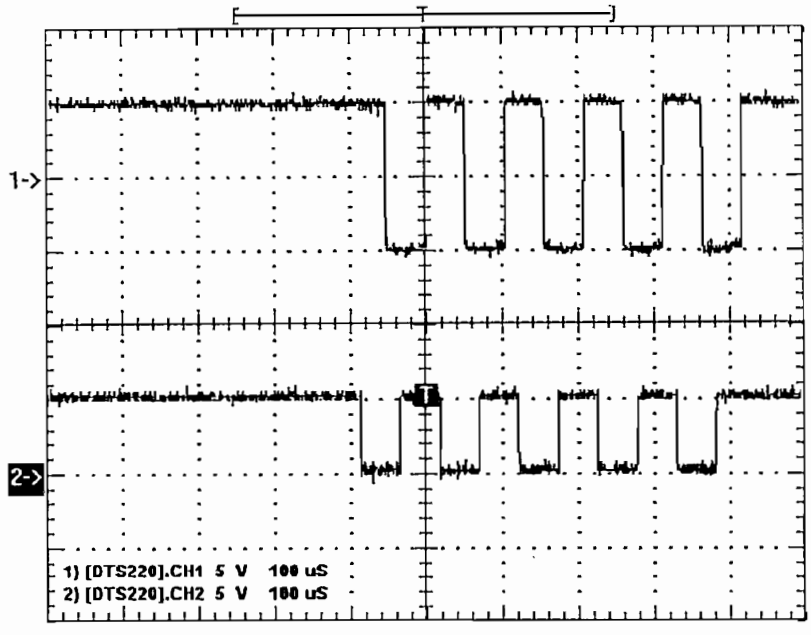


Figura 4.9 Codificación NRZ Polar de la letra "U" a 19200 bits/seg.

Al igual que en la codificación AMI, la codificación NRZ Polar presenta un retardo de aproximadamente medio período de bit como se puede observar en las Figuras 4.8 y 4.9.

Para los códigos HDB3 y B3ZS se tienen otros dos símbolos que son la letra V que indica la violación a la regla de AMI y la letra B que indica el bit de relleno.

4.3.3 CÓDIGO HDB3

Para el código HDB3 con la letra "A" a la velocidad de 300 bits/seg se tiene la Figura 4.10. La letra "A" esta representada por el valor binario 01000001. La señal de datos binaria capturada en la figura 4.10 muestra la siguiente secuencia binaria 0100000101 con sus respectivos niveles codificados que son 0, +A, 0, 0, 0, V (+A), 0, -A, 0, +A. Analizando la secuencia de datos codificados se puede observar que la sustitución que se ha realizado es 000+, la cual es una de las posibilidades de sustitución que presenta este tipo de codificación.

En la figura 4.10 se puede observar el retardo de aproximadamente cuatro y medio periodos de bit que son necesarios para empezar la codificación utilizando

el código HDB3, de ahí que los valores codificados e indicados anteriormente se los realizaron considerando este retardo.

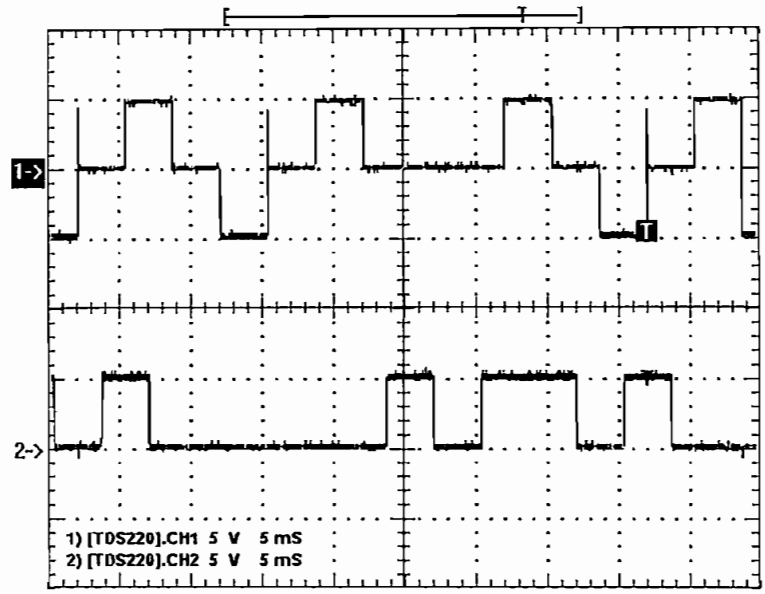


Figura 4.10 Codificación HDB3 de la letra "A" a 300 bits/seg.

La Figura 4.11 muestra la codificación HDB3 a la velocidad límite de 4800 bits/seg, utilizando la letra "A" para la realización de la prueba. La secuencia de datos a codificar que aparece en la Figura 4.11 es 111101000001011, en tanto que la secuencia codificada presenta los niveles +A, -A, +A, -A, 0, +A, -A, 0, 0, -A, 0, +A, 0, -A, +A.

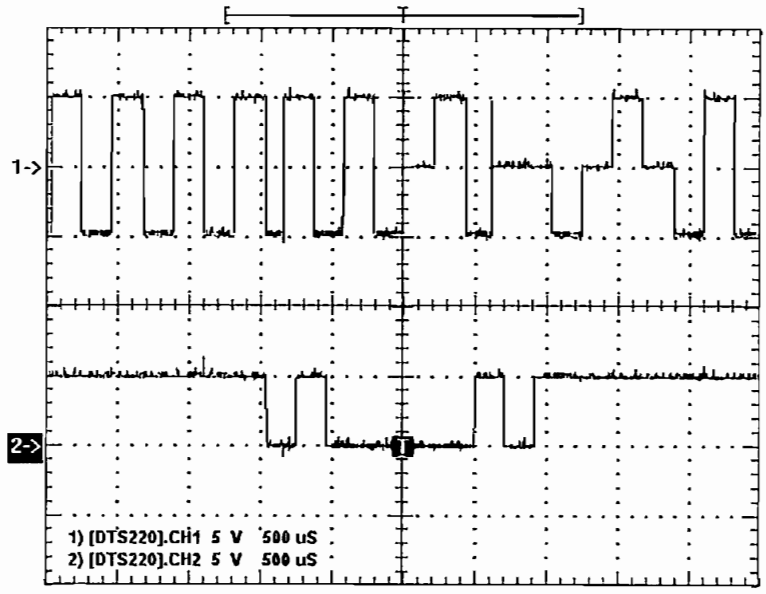


Figura 4.11 Codificación HDB3 de la letra "A" a 4800 bits/seg.

Al igual que para la velocidad de 300 bits/seg se debe tomar muy en cuenta el retardo de aproximadamente cuatro y medio períodos de bit para el análisis.

4.3.4 CÓDIGO B3ZS

En la codificación B3ZS se tiene un retardo de 3 períodos de bit obligatorios que se deben tomar en cuenta al momento de codificar una secuencia binaria. Es por esto que al momento de realizar la prueba de codificación en la Figura 4.12 se puede observar dicho retardo en la señal codificada. La letra que se ha tomado en cuenta para la realización de la prueba al igual que para HDB3, es la letra "A".

La secuencia de datos a codificar que aparece en esta Figura 4.12 es 0101000001011010000010110 1 mientras que la secuencia de datos codificados es 0, +A, 0, -A, 0, 0, V (-A), 0, 0, +A, 0, -A, +A, 0, -A, B (+A), 0, V (+A), 0, 0, -A, 0, +A, -A, 0, +A.

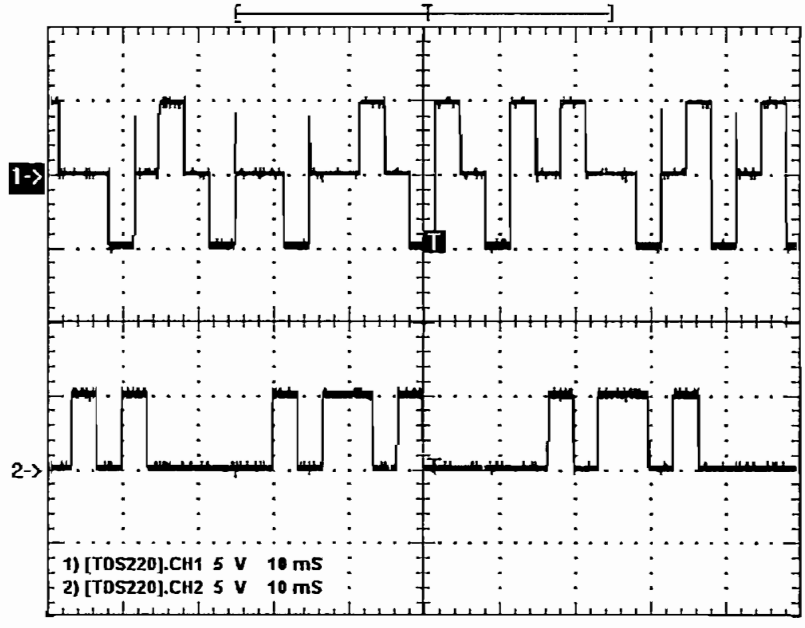


Figura 4.12 Codificación B3ZS de la letra "A" a 300 bits/seg.

En la Figura 4.12 se puede observar la sustitución 00- y +0+ que son dos de las cuatro posibles sustituciones que presenta esta regla de codificación.

La Figura 4.13 muestra otro ejemplo de la codificación B3ZS a la velocidad límite de 4800 bits/seg de la letra "A", con la secuencia de datos binaria a codificar de 11111010000010111111111, la misma al ser codificada se obtiene la secuencia –A, +A, -A, +A, -A, 0, +A, 0, 0, V (+A), 0, 0, -A, 0, +A, -A, +A, -A, +A, -A, +A.

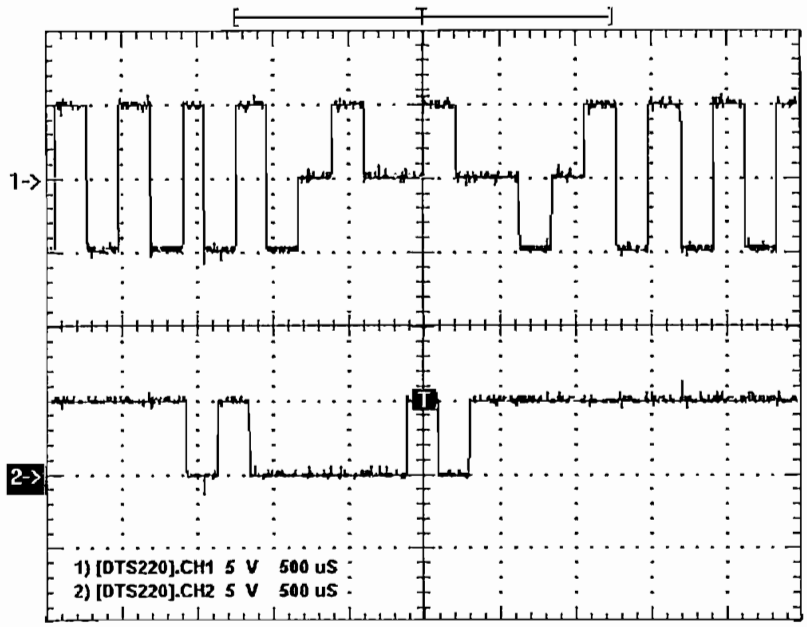


Figura 4.13 Codificación B3ZS de la letra "A" a 4800 bits/seg.

4.3.5 CÓDIGO MLT3

En la codificación MLT3, las pruebas se las ha realizado para la letra "U" debido a que es el carácter que presenta unos y ceros alternados. La secuencia de datos binarios a codificar según la Figura 4.14 son 101010101010101, en tanto que la secuencia de datos codificados es 0, 0, -A, -A, 0, 0, +A, +A, 0, 0, -A, -A, 0, 0, +A. Analizando la secuencia se puede ver, que si en la secuencia de datos está presente un cero lógico, la señal codificada mantendrá el estado anterior.

En la Figura 4.14 se puede observar que el retardo que presenta la señal de datos codificados con respecto a la señal de datos a codificar es de aproximadamente medio período de bit para esta codificación. La Figura 4.15 muestra otro ejemplo de codificación MLT3 a la velocidad de 4800 bits/seg para la misma letra del anterior ejemplo.

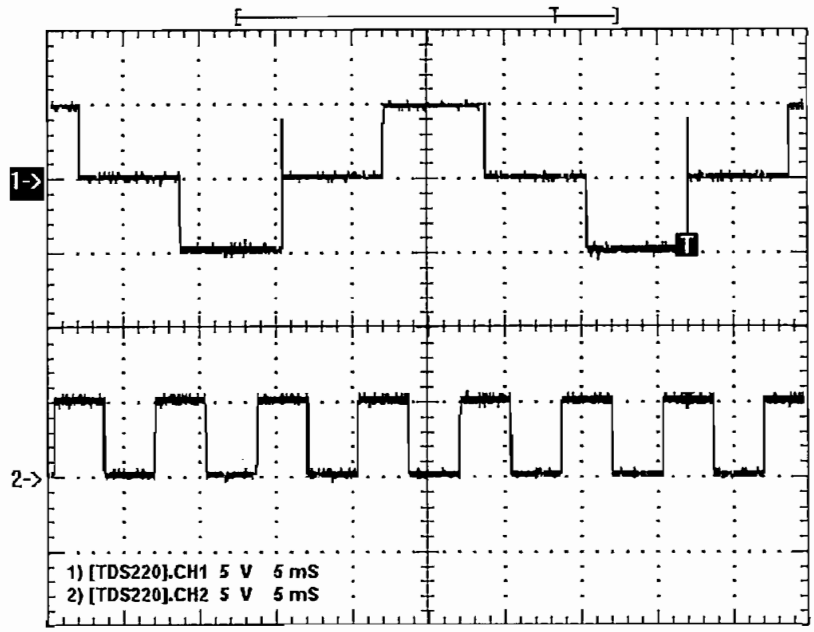


Figura 4.14 Codificación MLT3 de la letra "U" a 300 bits/seg.

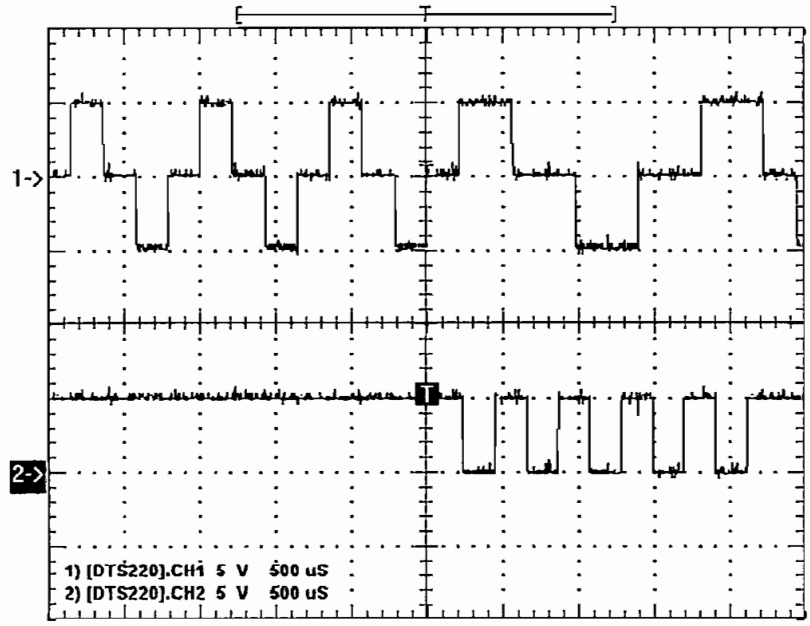


Figura 4.15 Codificación MLT3 de la letra "U" a 4800 bits/seg.

4.3.6 CÓDIGO RZ POLAR

Las pruebas del código RZ Polar se muestran en la Figura 4.16, en ella se muestra la señal de datos a codificar que representa la letra "U". Como se puede ver en la Figura 4.16, los retornos a cero se producen aproximadamente a medio período de bit tanto para los unos lógicos al igual que los ceros lógicos.

La Figura 4.16 presenta la codificación de la secuencia binaria 0101010101010.

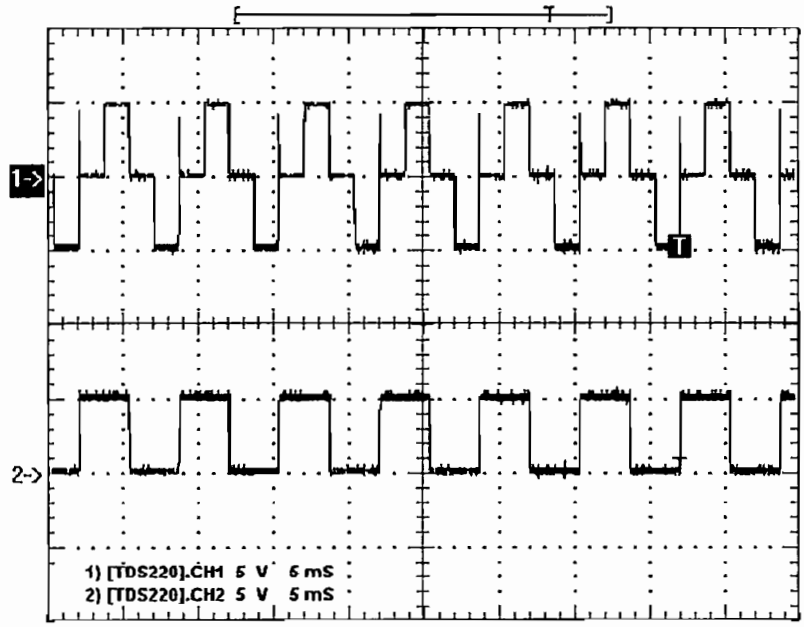


Figura 4.16 Codificación RZ de la letra "U" a 300 bits/seg.

El límite superior de velocidad para el código RZ es 4800 bits/seg y la prueba realizada a esta velocidad se muestra en la Figura 4.17. En la figura, la secuencia de datos a codificar es 11111110101010101011111111.

Se debe tomar en cuenta para el análisis de las Figuras 4.16 y 4.17, que se tiene un retardo de aproximadamente medio período de bit, debido al procesamiento de la señal de datos.

4.3.7 CÓDIGO BIFASE-M

Un ejemplo de la codificación Bifase-M a la velocidad de 300 bits/seg se presenta en la Figura 4.18, donde se puede observar las transiciones al inicio y a medio período de bit cuando se presenta un uno lógico en tanto que se tienen únicamente transiciones al inicio cuando se tienen presentes ceros lógicos. La secuencia de datos a codificar es 1110110101011110.

Para el análisis de la codificación Bifase-M se debe tomar en cuenta, que se tiene un retardo de aproximadamente medio período de bit.

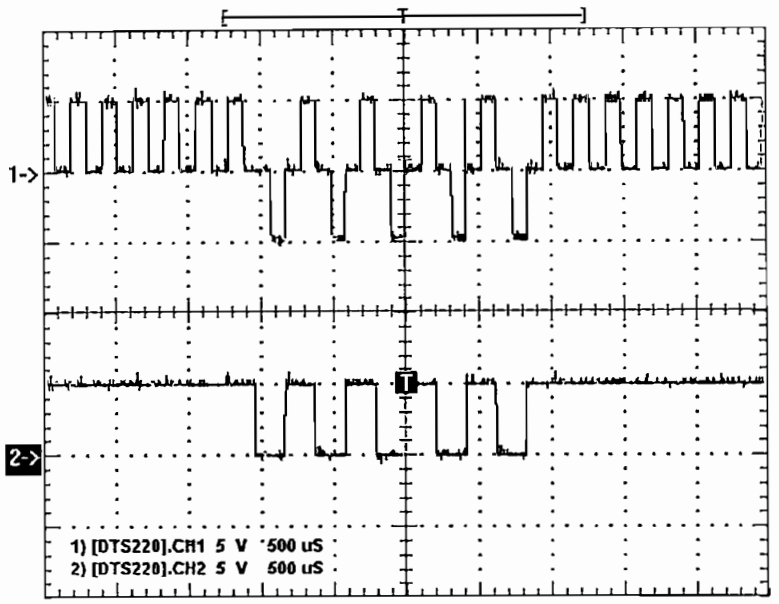


Figura 4.17 Codificación RZ de la letra "U" a 4800 bits/seg.

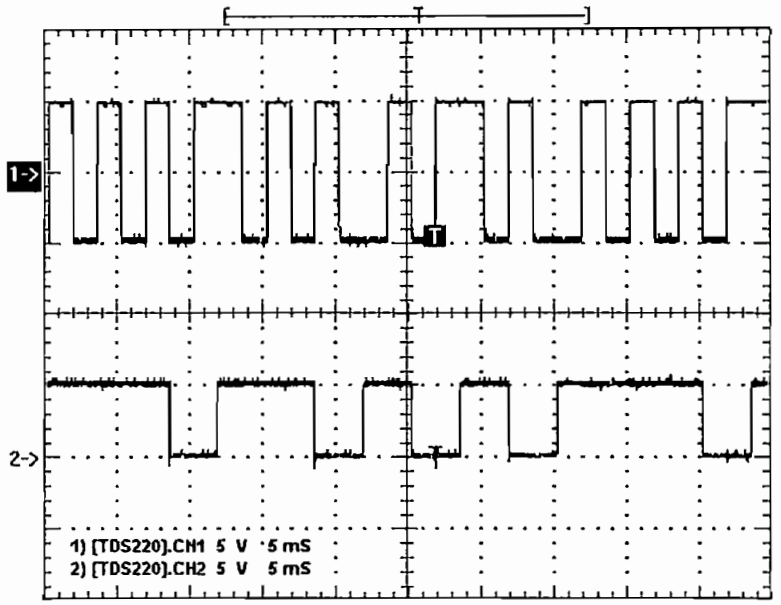


Figura 4.18 Codificación Bifase-M de la letra "u" a 300 bits/seg.

En la Figura 4.19 se muestra otro ejemplo de codificación Bifase-M a la velocidad de 4800 bits/seg, tomando como carácter de prueba la letra "u".

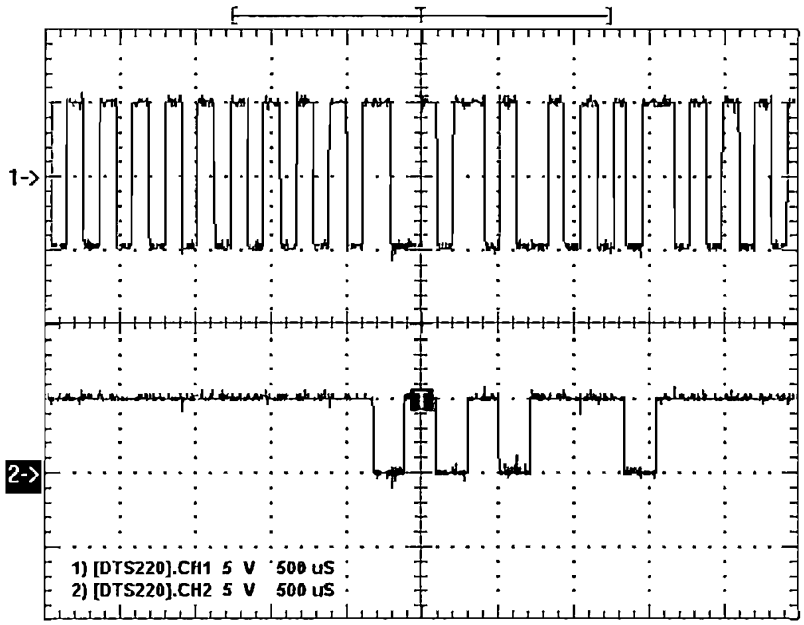


Figura 4.19 Codificación Bifase-M de la letra "u" a 4800 bits/seg.

4.3.8 CÓDIGO MANCHESTER DIFERENCIAL

Las pruebas que se ha realizado para la codificación Manchester Diferencial a las velocidades de 300 bits/seg y 4800 bits/seg se muestran en las Figuras 4.20 y 4.21 respectivamente.

Allí se puede observar que se producen transiciones al inicio y a la mitad del período de bit para un cero lógico que ingrese al codificador, mientras que se tiene una transición a la mitad del período de bit para un uno lógico.

Para el análisis de codificación de las dos gráficas mencionadas, se debe considerar el retardo de aproximadamente medio período de bit que se tiene entre la señal de datos codificados y la señal de datos a codificar.

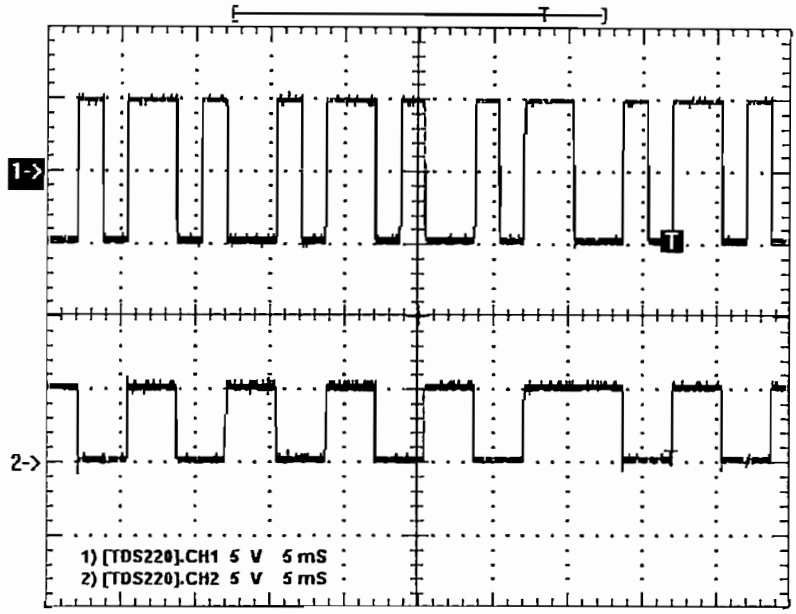


Figura 4.20 Codificación Manchester Diferencial de la letra "U" a 300 bits/seg.

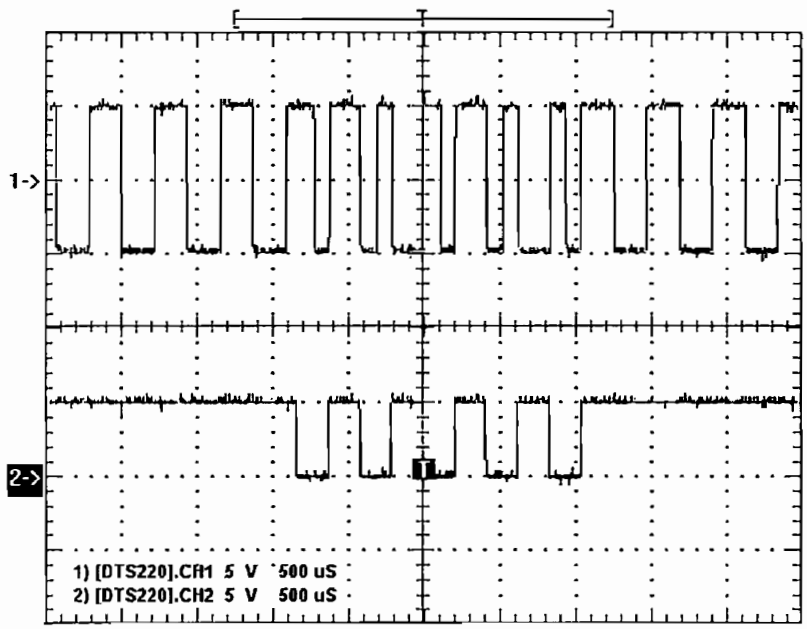


Figura 4.21 Codificación Manchester Diferencial de la letra "U" a 4800 bits/seg.

4.3.9 CÓDIGO CMI

En la codificación CMI, las pruebas realizadas se muestran en las Figuras 4.22 y 4.23 a las velocidades de 300 bits/seg y 4800 bits/seg respectivamente. El carácter utilizado en las pruebas es la letra "U".

En el análisis de la señal codificada se debe tomar en cuenta el retardo de medio periodo de bit que existe entre la señal de datos codificada y la señal de datos a codificar, con esta consideración se puede observar que los unos lógicos siguen la regla de AMI, en tanto que en los ceros lógicos se tiene transición ascendente a medio período de bit.

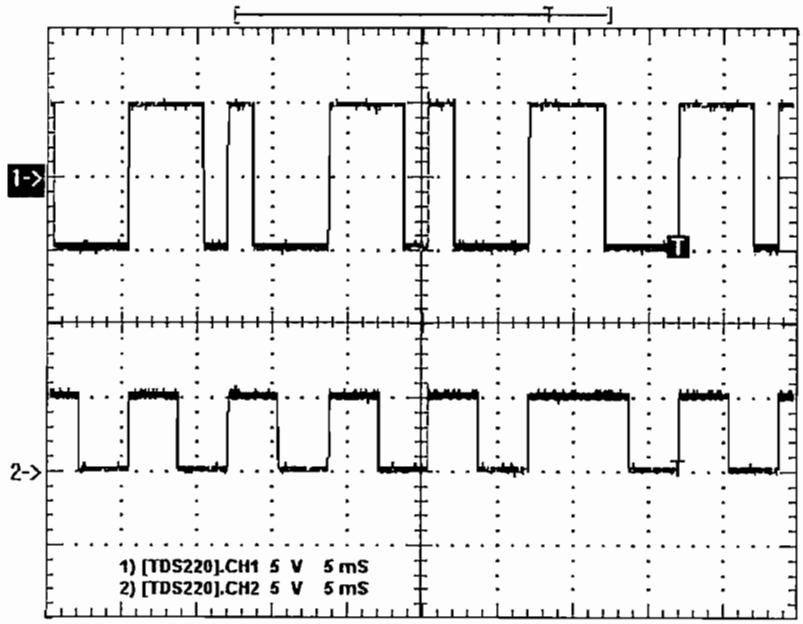


Figura 4.22 Codificación CMI de la letra "U" a 300 bits/seg.

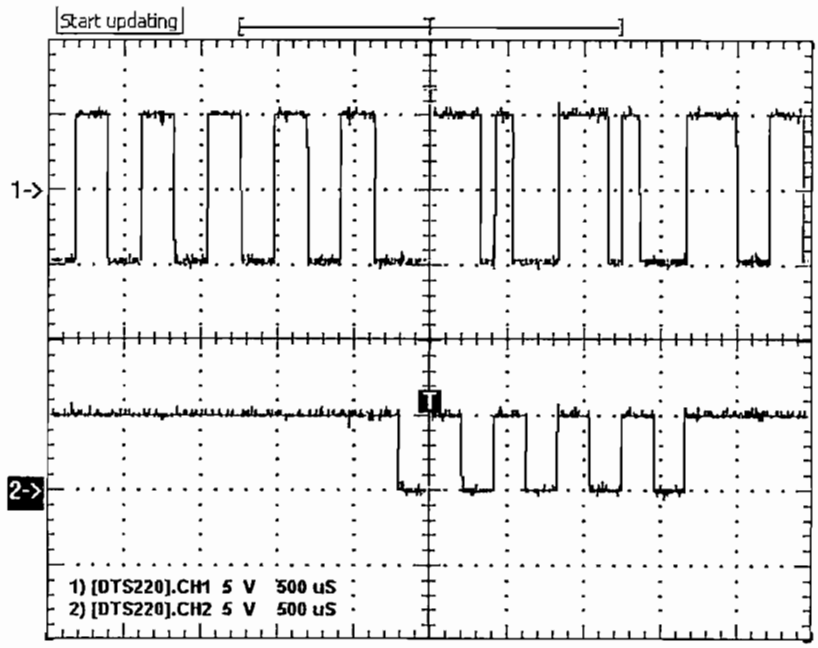


Figura 4.23 Codificación CMI de la letra "U" a 4800 bits/seg.

4.3.10 CÓDIGO MILLER

En las Figuras 4.24 y 4.25 se ilustran la codificación Miller a las velocidades de 300 bits/seg y 4800 bits/seg respectivamente, para la letra "A".

En esta codificación, si se tiene dos ceros consecutivos se debe producir una transición al inicio, de no presentarse éste caso, el cero lógico debe mantener el estado anterior. En el caso de tener un uno lógico se debe producir una transición a medio periodo de bit.

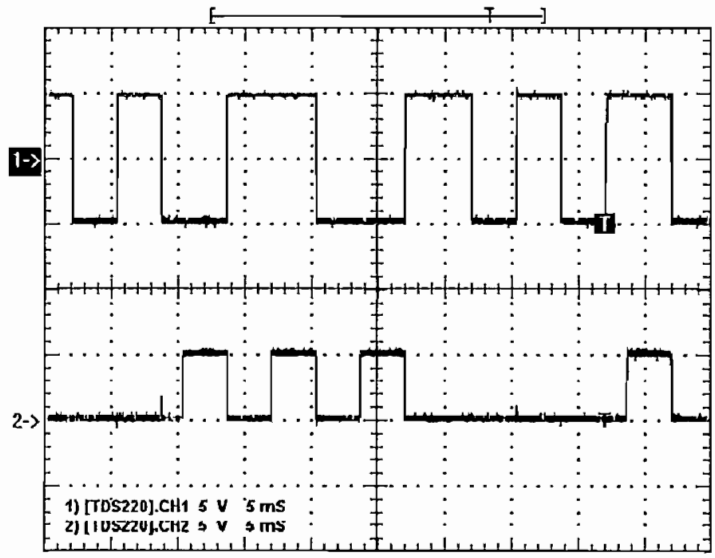


Figura 4.24 Codificación Miller para letra "A" a 300 bits/seg.

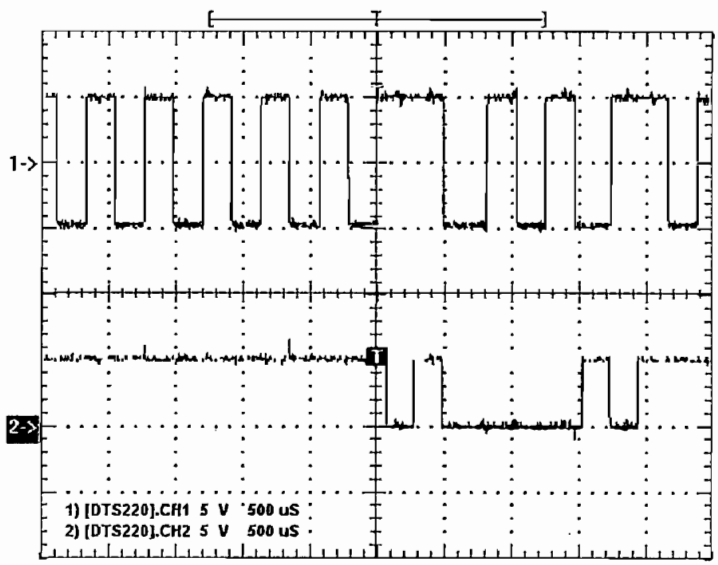


Figura 4.25 Codificación Miller para letra "A" a 4800 bits/seg.

Al igual que en la mayoría de los códigos, para realizar el análisis de la codificación en las Figuras 4.24 y 4.25 se debe considerar el retardo existente de medio período de bit.

4.3.11 CÓDIGO MS43

En la Figura 4.26 se muestra la codificación MS43 para la velocidad de 300 bits/seg. La letra utilizada en la prueba de codificación es la letra "A", la secuencia codificada es 01000001. La secuencia de datos codificados es 0, -A, 0, +A, -A, +A.

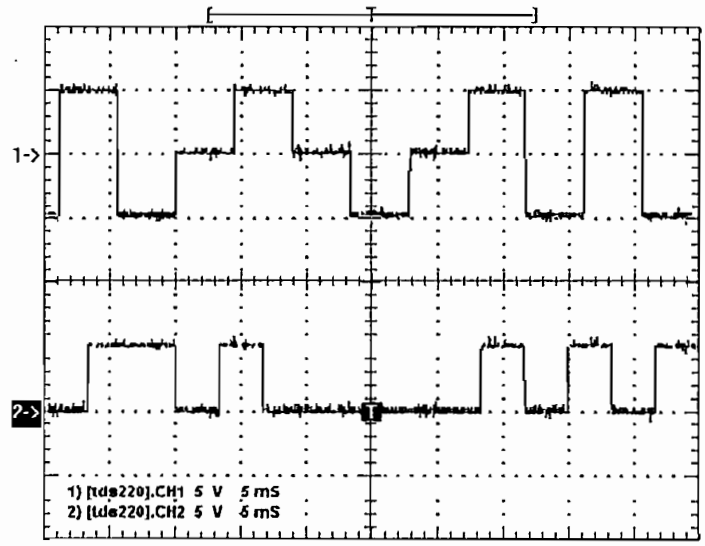


Figura 4.26 Codificación MS43 de la letra "A" a 300 bits/seg.

En este código se tiene un retardo de aproximadamente cuatro períodos y medio de bit, por lo que para el análisis se debe tomar muy en cuenta este aspecto, así como la polaridad acumulada, para este código la polaridad inicial acumulada es de -1.

4.3.12 CÓDIGO 4B3T

La Figura 4.27 muestra la codificación 4B3T para la velocidad de 300 bits/seg que es la única velocidad a la cual trabaja esta codificación. La letra utilizada en la prueba de codificación es la letra "A", la secuencia codificada es 01101000. La secuencia de datos codificados es 0, 0, -A, -A, +A, 0.

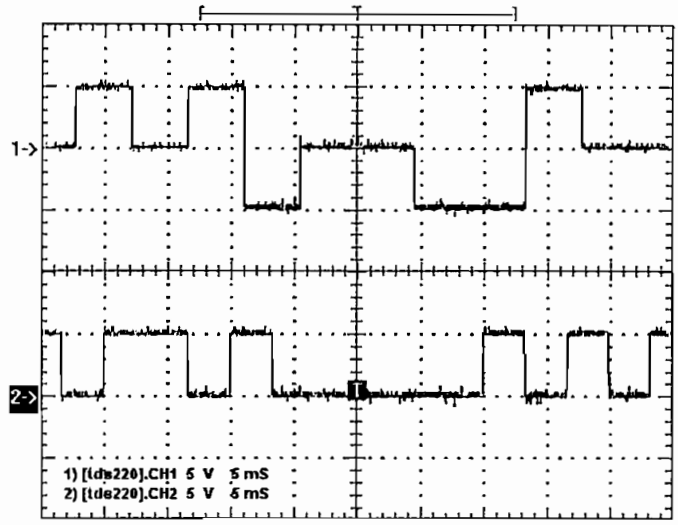


Figura 4.27 Codificación 4B3T de la letra "A" a 300 bits/seg.

Este código al igual que el anterior tiene un retardo de aproximadamente cuatro períodos y medio de bit, por lo que para el análisis se debe tomar muy en cuenta este aspecto, así como la polaridad acumulada, para este código la polaridad inicial acumulada es de +1.

4.3.13 SEÑALES DE RELOJ

Una vez hecha las pruebas para cada uno de los códigos, se presentan las señales de reloj que se utilizan para codificación y decodificación de los diferentes códigos, estas señales llevan los colores Azul para la señal de decodificación y el Rojo para la señal de reloj de codificación.

En la Figura 4.28 se muestra las señales de reloj de codificación y decodificación de los códigos AMI, NRZ Polar, MLT3, HDB3, B3ZS.

En la Figura 4.29 se muestra las señales de reloj de codificación y decodificación de los códigos RZ Polar, Manchester Diferencial, CMI, Miller, Bifase M.

En la Figura 4.30 se muestra las señales de reloj de codificación y decodificación de los códigos MS43 y 4B3T.

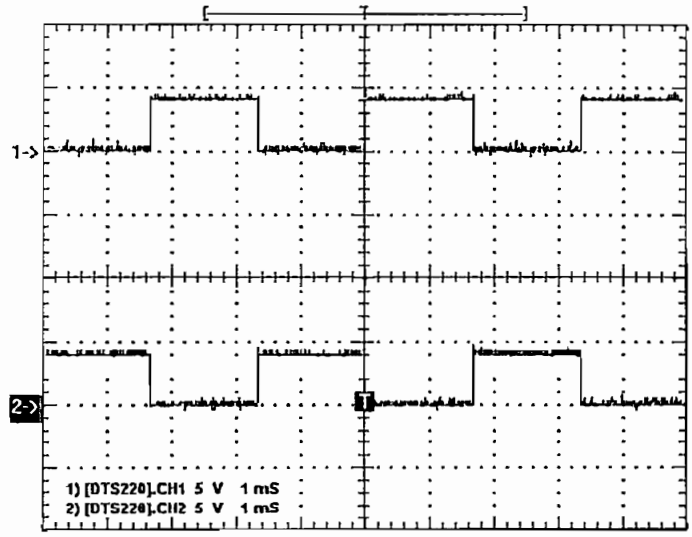


Figura 4.28 Reloj de codificación y decodificación AMI, NRZ Polar, MLT3, B3ZS, HDB3.

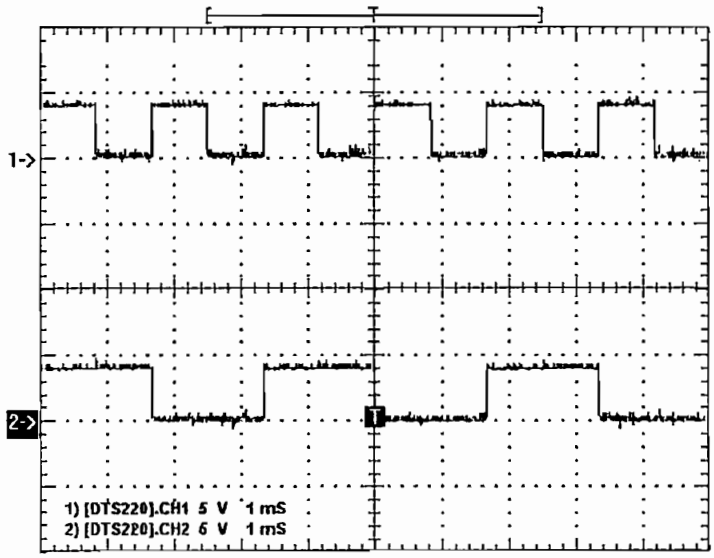


Figura 4.29 Reloj de codificación y decodificación RZ Polar, Manchester Diferencial, CMI, Miller, Bifase M.

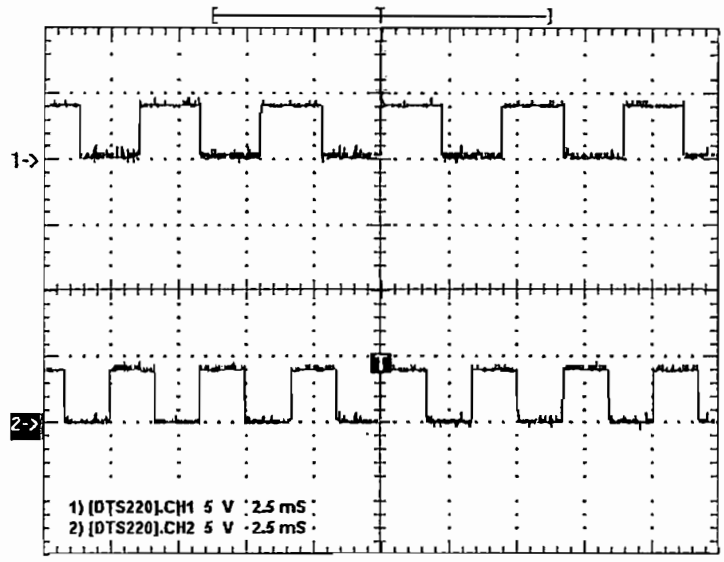


Figura 4.30 Reloj de codificación y decodificación MS43 y 4B3T.

4.4 PRUEBAS DE VELOCIDADES DE TRANSMISIÓN

De las pruebas de funcionamiento de cada uno de los códigos se obtuvieron las velocidades máximas a las cuales puede trabajar el codificador / decodificador de códigos de línea modo full duplex, estos valores se muestran en la Tabla 4.1.

CÓDIGO DE LÍNEA	VELOCIDAD MÁXIMA (MODO FULL DUPLEX)
Código AMI	19200 bits/seg.
Código NRZ	19200 bits/seg.
Código HDB3	2400 bits/seg.
Código B3ZS	2400 bits/seg.
Código MLT3	4800 bits/seg.
Código RZ	4800 bits/seg.
Código Bifase M	4800 bits/seg.
Código Manchester Diferencial	4800 bits/seg.
Código CMI	4800 bits/seg.
Código Miller	4800 bits/seg.
Código MS43	300 bits/seg.
Código 4B3T	300 bits/seg.

Tabla 4.1 Velocidades Máximas de Transmisión.

4.5 TIEMPOS PROMEDIOS DE PROCESAMIENTO

La variación del tiempo de procesamiento que se le da a la señal a codificar, hace que se produzcan la diversidad de velocidades en los diferentes códigos, es por esto que se ha realizado un promedio de los tiempos de procesamiento para cada uno de los códigos de línea presentes en el codificador/decodificador. Tabla 4.2.

Los tiempos de procesamiento permiten darse cuenta de cómo afectan estos en la codificación, pues de producirse la interrupción para decodificar, en un punto intermedio de la codificación se puede producir errores y esto es cada vez más notorio al momento de ir aumentando la velocidad de transmisión.

CÓDIGO DE LÍNEA	TIEMPOS PROMEDIO DE PROCESAMIENTO
Código AMI	3.3 us
Código NRZ	2.4 us
Código HDB3	9.0 us
Código B3ZS	10.2 us
Código MLT3	4.2 us
Código RZ	6.0 us
Código Bifase M	6.3 us
Código Manchester Diferencial	6.9 us
Código CMI	5.4 us
Código Miller	7.5 us
Código MS43	24.6 us
Código 4B3T	24.6 us

Tabla 4.2 Tiempos promedios de procesamiento.

4.6 LIMITACIONES DEL SISTEMA

4.6.1 LIMITACIONES DEL SOFTWARE PARA CONFIGURAR EL CODEC PORTÁTIL V1.0

Las limitaciones que presenta el *software* son:

Las limitaciones para configurar el CODEC Didáctico radican en que se debe ejecutar el programa Codec Portátil antes de conectar el CODEC, puesto que el programa contiene los eventos *SysInfo* que permiten detectar la conexión del CODEC. Si se hace lo contrario, el programa no reconocerá que el CODEC ya se ha conectado.

Para retirar y luego volver a conectar el CODEC, se debe tener cuidado de dar *click* en el botón aceptar a todo mensaje enviado por el programa, para evitar complicaciones con la comunicación entre la PC y el CODEC, pues en ocasiones puede producirse la no detección del CODEC. Para solucionar este problema se debe reiniciar el computador.

En cuanto a las aplicaciones que presenta el programa para codificar y decodificar, se encuentran limitadas por el procesamiento de datos, pues para los códigos MS43 y 4B3T se procesan un máximo de 39 datos, y 40 datos para los códigos restantes, mientras que la aplicación para visualizar la densidad espectral de potencia se encuentra limitada para trabajar con 6 códigos de línea.

Para visualizar la gráfica de la codificación de la información en la PC, se debe de utilizar la opción de enviar datos a ser codificados que presenta el programa, caso contrario no se podrá visualizar dicha gráfica.

4.6.2 LIMITACIONES DEL *HARDWARE* DEL CODEC PORTÁTIL

Las limitaciones que presenta el *hardware* son:

Debido a que el dispositivo no posee fuente de alimentación externa, éste se encuentra limitado por la utilización del voltaje que entrega el interfaz USB, por lo tanto éste no puede funcionar sino se encuentra conectado al PC utilizando el interfaz USB o un concentrador que tenga la capacidad de entregar la corriente suficiente para que el dispositivo funcione correctamente.

Puesto que el dispositivo es sincrónico, es necesario transmitir a más de la señal codificada la señal de reloj, la cual permitirá la recuperación de la señal. Esto es una limitante, pues el medio de transmisión puede presentar ruido y hacer que la señal de reloj de decodificación se corrompa, produciendo que la recuperación de los datos sea errónea. La señal de reloj se puede recuperar de la propia señal de datos, pero al tratarse de un dispositivo de uso didáctico es preferible tratar la señal de datos y la señal de reloj por separado.

Otra limitante, son los tiempos de procesamiento que son los que impiden que varios códigos puedan alcanzar la velocidad de 19200 bits/seg., el tiempo de procesamiento promedio como se puede ver en la Tabla 4.2 varía dependiendo del código, es decir del grado de complejidad que presente cada uno.

Puesto que no se trata de un dispositivo comercial, éste no cumple con los niveles de señal que se requiere según el estándar que se vaya a aplicar a determinada tecnología, pues los niveles de voltaje que maneja el CODEC son de +5V, 0V, y –5V.

La señal de reloj de decodificación es otra limitante, pues dependiendo del código, éste tiene que variar de frecuencia con respecto a la frecuencia del reloj de codificación, es decir, para el caso del código RZ Polar y códigos Bifases se debe tener un reloj de decodificación del doble de la frecuencia de la señal de reloj de codificación, no así, para el código MS43 y 4B3T que necesitan de una señal de reloj de decodificación con una frecuencia igual a los ¾ de la frecuencia de la señal de reloj de codificación.

Debido a que la señal de reloj de decodificación debe sincronizarse con la señal de datos a codificar, se suelen producir errores en la recepción de los datos decodificados. Este error aleatorio es notorio cuando se digitan distintos caracteres, en tanto que cuando se tiene un cadena de datos, el error que se produce es menor, es decir que se puede producir un dato erróneo en el momento de la sincronización, luego de esto la ocurrencia del error disminuye considerablemente con respecto al caso anterior.

Otra limitante del CODEC, consiste en que a la entrada de la señal de codificación solo se puede ingresar una de las dos posibilidades de niveles de señal, sea ésta RS-232 o TTL, no las dos al mismo tiempo. A su vez, la señal RS-232 puede ser ingresada por una de las dos entradas que posee, es decir, que se lo puede hacer a través del conector DB-9 que se encuentra en la parte trasera del dispositivo, o por el conector que se encuentra en la parte frontal del dispositivo, no por las dos entradas al mismo tiempo.

4.7 COSTOS DE CONSTRUCCIÓN

En la Tabla 4.3 se presenta una lista detallada de precios de cada uno de los elementos utilizados en el diseño, indicando la cantidad, el precio unitario, el precio total y el monto total del proyecto.

CANTIDAD	COMPONENTE	PRECIO UNITARIO	PRECIO TOTAL
1	74HC4052	1,50	1,50
1	LM319N	3,85	3,85
4	LM310N	0,45	1,80
1	74LS32	0,44	0,44
2	74LS244	0,60	1,20
1	74LS14	0,44	0,44
1	PIC 16F870	7,85	7,85
1	At89C52	8,00	8,00
2	CRISTAL 20 MHz	1,20	2,40
1	74LS157	0,50	0,50
1	MAX232	3,30	3,30
1	LM318N	1,50	1,50
1	MODULO USB MOD2	60,00	60,00
15	CAP. 0.1 uF cerámicos	0,10	1,50
1	NTE 585	1,41	1,41
1	MAX765	13,80	13,80
1	LM339N	0,37	0,37
10	CAP. 100 uF cerámicos	0,10	1,00
4	CAP. 33 pF cerámicos	0,10	0,40
9	Resistencias de 1/4 W	0,02	0,18
1	Caja Metálica	2,80	2,80
2	Zener de 5.1V de 1/4 W	0,15	0,30

50cm	Cable Multipar	0,30 c/m	0,15
2	Conector DB-9 Macho	0,28	0,56
2	Conector dDB-9 Hembra	0,70	1,40
1	Placa	25,00	25,00
3	Zócalo de 16 pines	0,07	0,21
6	Zócalo de 8 pines	0,04	0,24
6	Zócalo de 14 pines	0,06	0,36
2	Zócalo de 20 pines	0,09	0,18
2	Zócalo de 24 pines	0,10	0,20
3	Jacks	0,24	0,72
1	Cable USB	1,50	1,50
3	Cable UTP	0,28	0,84
TOTAL			**145,90**

Tabla 4.3 Detalle de precios de los elementos utilizados en el diseño.

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

Con el avance tecnológico cada día aparecen más y más dispositivos con interfaz USB, ya que estos son los preferidos por los usuarios, por su facilidad de conexión y demás ventajas que se han mencionado en el Capítulo I.

El dispositivo de codificación digital utiliza la interfaz USB, que le permite alimentar los circuitos integrados utilizados, así como establecer una comunicación con la PC para el control del mismo y la verificación de su estado de conexión.

Además el dispositivo al utilizar la interfaz USB tiene la capacidad de ser conectado al PC en pleno funcionamiento, sin causar daños al PC o al dispositivo, y sin necesidad de reiniciar o abrir el PC, ya que un PC típico tiene por lo menos dos puertos USB.

Si el dispositivo se conecta por primera vez, Windows solicitará al usuario la ubicación del controlador, el cual se encuentra en C:\Windows\System32, dicha instalación se la detalla detenidamente en el Anexo C.

La interfaz USB incluye una señal de alimentación, proveniente de las fuentes de la PC o del concentrador, lo cual permite eliminar una parte del circuito que ocupa un espacio considerable en todos los diseños, que es la fuente de alimentación, disminuyendo a la vez los costos de construcción del dispositivo.

Los circuitos y el código de bajo consumo de energía se accionan automáticamente en los periféricos USB cuando no están en uso, con lo cual, los circuitos integrados utilizados quedan sin alimentación y por ende se suspende el

funcionamiento del dispositivo. Por esta razón el dispositivo debe estar siempre enviando datos al PC y así evitar la suspensión del mismo, imposibilitando así la utilización del control *SysInfo* para detectar la desconexión inmediata del CODEC y se utiliza el método de chequear la llegada de datos desde el dispositivo para determinar la desconexión del CODEC.

En la realización del proyecto, en la parte del *hardware* la mayor dificultad que se presentó es en la decodificación, la cual requiere mayor circuitería para el reconocimiento de la señal en forma digital de los niveles presentes en un determinado código; se hace necesario aplicar los distintos métodos para la sincronización de la señal de decodificación, ya sea esto por recuperación de la señal del reloj de la propia señal codificada o a través de la transmisión de la señal de reloj de decodificación.

Aplicado el método de transmisión de reloj de decodificación junto con la señal codificada, se ha podido constatar que con algunos códigos se pueden alcanzar mayores velocidades que si se realizara la recuperación de la señal de reloj, esto se lo ha hecho comparando con los datos de velocidad obtenida con el codificador diseñado por el Ing. Nelson Ávila, dichos valores se muestran en la Tabla 5.1.

En el desarrollo del software para configurar el dispositivo a través de la PC utilizamos las herramientas proporcionadas por Visual Basic 6.0, las mismas que nos ayudan a detectar la conexión del CODEC, la configuración del mismo vía USB y a desarrollar las demás aplicaciones presentadas por el programa Codec Portátil para ser utilizadas por el usuario, mejorando la comprensión de la codificación de línea.

La comunicación entre la PC y el dispositivo se realizó a través del VCP (*Virtual Com Port*), el cual es configurado fácilmente en Visual Basic 6.0 con el control *MSComm.*

CÓDIGO DE LÍNEA	VELOCIDAD MÁXIMA (MODO FULL DUPLEX)	
	CODEC SÍNCRONO	CODEC ASÍNCRONO
Código AMI	19200 bits/seg.	4800 bits/seg.
Código NRZ	19200 bits/seg.	4800 bits/seg.
Código HDB3	2400 bits/seg.	1200 bits/seg.
Código B3ZS	2400 bits/seg.	1200 bits/seg.
Código RZ	4800 bits/seg.	1200 bits/seg.
Código Bifase M	4800 bits/seg.	2400 bits/seg.
Código Manchester Diferencial	4800 bits/seg.	1200 bits/seg.
Código Millar	4800 bits/seg.	1200 bits/seg.
Código MS43	300 bits/seg.	300 bits/seg.
Código 4B3T	300 bits/seg.	300 bits/seg.

Tabla 5.1 Comparación de valores de velocidad máximas de transmisión en modo Full Duplex

La verificación del estado de conexión se la realiza con la ayuda de los controles *SysInfo, MSComm* y *Ttimer.* El control *Sysinfo* permite detectar cuando el CODEC ha sido conectado por el usuario. El control *MSComm* permite chequear la presencia de un dato en el VCP y con la ayuda del control *Timer*, permiten determinar si el CODEC ha sido desconectado, al no recibir datos desde el dispositivo dentro de un intervalo de tiempo de 1 segundo.

5.2 RECOMENDACIONES

Sería importante que se pueda realizar controladores *plug and play* USB, es por esto que se recomienda realizar un estudio que se dedique exclusivamente a la generación de controladores USB.

Las velocidades de transmisión máximas en modo *full duplex* para cada uno de los códigos de línea aumentaría si utilizamos un microcontrolador de más alta velocidad, como lo es el microcontrolador ATMEL AT89C51RD2 que llega a velocidades de 60 *MHz* con 12 ciclos de reloj por ciclo de máquina o el microcontrolador PHILIPS P89C51RD2 que llega a velocidades de 66 *MHz* con 12

ciclos de reloj por ciclo de máquina. Al utilizar cualquiera de estos microcontroladores los tiempos de procesamiento se reducirían notablemente.

Los tiempos promedios de procesamiento y las velocidades de transmisión máximas teóricas alcanzadas al utilizar el microcontrolador PHILIPS P89C51RD2 se muestran en la Tabla 5.2.

CÓDIGO DE LÍNEA	TIEMPOS PROMEDIO DE PROCESAMIENTO TEÓRICO	VELOCIDAD MÁXIMA (MODO FULL DUPLEX) TEÓRICA
Código AMI	0.99 us	38400 bits/seg.
Código NRZ	0.72 us	38400 bits/seg.
Código HDB3	2.70 us	19200 bits/seg.
Código B3ZS	3.06 us	19200 bits/seg.
Código MLT3	1.26 us	19200 bits/seg.
Código RZ	1.80 us	19200 bits/seg.
Código Bifase M	1.89 us	19200 bits/seg.
Código Manchester Diferencial	2.07 us	19200 bits/seg.
Código CMI	1.62 us	19200 bits/seg.
Código Miller	2.25 us	19200 bits/seg.
Código MS43	7.38 us	4800 bits/seg.
Código 4B3T	7.38 us	4800 bits/seg.

Tabla 5.2 Tiempos promedios de procesamiento y velocidades de transmisión máximas teóricas alcanzadas por el microcontrolador P89C51RD2.

BIBLIOGRAFÍA

BIBLIOGRAFÍA

[1] AXELSON, Jan. USB Complet. Segunda Edición. Editorial Lakeview Research. Estados Unidos de Norteamérica. 2001.

[2] ÁVILA, Nelson. Diseño y construcción de un CODEC didáctico para la transmisión en banda base. Ecuador. 1994.

[3] ANGULO, José; ROMERO, Susana; ANGULO, Ignacio. Microcontroladores PIC. Diseño Práctico de Aplicaciones. Segunda Parte: PIC16F87X. Segunda Edición; Editorial Mc Graw Hill; España; 2000.

[4] PHILIPS. ECG. Master Replacement, Guide. 18 TH Edición. Editorial Philips. Estados Unidos de Norteamérica; 1998.

[5] HIDALGO, Pablo. Folleto de Comunicación Digital. Ecuador. 2002.

[6] COUCH II, León W. Sistemas de comunicación digital y analógica. Quinta Edición. Prentice Hall. México. 1998.

[7] LATHI, B. Introducción a la teoría y sistemas de comunicación. Editorial Limusa. México. 1995.

[8] TANENBAUM, Andrew. Redes de ordenadores. Tercera edición. Prentice Hall. México, 1997.

[9] SWARTZFAGER, Gene; CHANDAK, Armes; CHANDAK, Purshottam; ALVAREZ, Steve. Visual basic 6, Programación orientada a objetos. Paraninfo. España. 1999.

[10] MCKINNEY, Bruce. Programación avanzada con Microsoft Visual Basic 5.0. McGraw Hill, España, 1997.

[11] Cevallos Francisco J, Curso de programación de Visual Basic 6, Alfaomega, España, 1999.

ARTÍCULOS

[1] SOPORTE TÉCNICO OEM. Un Paseo por USB-1. Editorial FUJITSU. España. 2000.

[2] SOPORTE TÉCNICO OEM. Un Paseo por USB 2.0. Editorial FUJITSU. España. 2003.

DIRECCIONES ELECTRÓNICAS

[1] FORO PC *WORLD*. http://www .idg.es/pcworld/foro.

[2] INTEL. http://www.intel.com/technology/usb.

[3] *APPLE Corporation.* http://www.apple.com.

[4] Microsoft. http://www.microsoft.com/whdc/hwdev/bus/USB/default.mspx.

[5] Microchip *Technology* Inc. http://www.microchip.com.

[6] http://www.usb.org/developers/onthego.

[7] http://www.lvr.com/usb.htm.

[8] http://www.support@ftdichip.com.

[9] http://datasheets@dialelec.com.

[10] http://www.ftdi.co.uk.

[11] http://www.ravar.net.

[12] http://www.monografias.com/trabajos13/fire/fire.shtml#intro.

[13] http://www.diotronic.com.

[14] http://www.inictel.gob.pe/internet/publica.htm.

[15] http://www.hispavila.

ANEXOS

ANEXO A

ESTÁNDAR USB 1.1 Y REGLAS DE CODIFICACIÓN

CONTENIDO:

Tabla de Peticiones Estándar USB 1.1.

Reglas de Codificación.

# Petición	Petición	Fuente de Datos (etapa de datos	Receptor	Valor	Índice	Longitud de datos (*bytes*) (en el estado de datos)	Datos (en la etapa de datos)
00h	*Get_Status*	Dispositivo	dispositivo, interfaz, *endpoint*	0	Dispositivo, interfaz, *endpoint*	2	Estado
01h	*Clear_Feature*	Ninguno	dispositivo, interfaz, *endpoint*	característico	Dispositivo, interfaz, *endpoint*	0	Ninguno
03h	*Set_Feature*	Ninguno	dispositivo, interfaz, *endpoint*	característico	Dispositivo, interfaz, *endpoint*	0	Ninguno
05h	*Set_Address*	Ninguno	Dispositivo	Direcciones de dispositivo	0	0	Ninguno
06h	*Get_ Descriptor*	Dispositivo	Dispositivo	Tipos de descriptores e índices	Dispositivo o ID de lenguaje	Longitud del descriptor	Descriptor
07h	*Set_ Descriptor*	Host	Dispositivo	Tipos de descriptores e índices	Dispositivo o ID de lenguaje	Longitud del descriptor	Descriptor
08h	*Get_ Configuration*	Dispositivo	Dispositivo	0	Dispositivo	1	Configuración
09h	*Set_ Configuration*	Ninguno	Dispositivo	Configuraciones	Dispositivo	0	Ninguno
0Ah	*Get_Interface*	Dispositivo	Interfaz	0	Interfaz	1	Ajuste alterno
0Bh	*Set_Interface*	Ninguno	Interfaz	Interfaz	Interfaz	0	Ninguno
0Ch	*Synch_Frame*	Dispositivo	*endpoint*	0	*endpoint*	2	Número de trama

Tabla A1 Peticiones estándares para transferencias de control[1].

[1] Tabla tomada del libro "*USB COMPLETE Second Edition*" de JAN AXELSON

REGLAS DE CODIFICACIÓN

CÓDIGO NRZ POLAR

La codificación NRZ Polar permite la transmisión de un 1L mediante un pulso positivo de amplitud +A y un 0L mediante un pulso negativo de amplitud –A.

CÓDIGO AMI (*Alternate Mark Inversion*)

La codificación utilizando el código AMI permite reducir la componente continua, esto se logra alternando la polaridad de los 1L con una marca positiva o negativa, en tanto que la condición de 0L se transmite como valor cero.

CÓDIGO RZ Polar

La metodología de codificación consiste en que el 1L tiene una amplitud de +A y el cero lógico tiene una amplitud de –A, pero ambas señales retornan a cero, durante un porcentaje del tiempo de bit T_b.

CÓDIGO B3ZS (Bipolar with 3 Zeros Substitution)

El código B3ZS (código bipolar con substitución de 3 ceros) permite que se codifiquen los 1L utilizando pulsos positivos y negativos de forma alternada. Este tipo de codificación no permite más de dos ceros consecutivos, si se tiene más de dos ceros consecutivos se reemplaza por la secuencia B0V o 00V, donde B es un pulso de relleno y V es el pulso de violación de la regla de AMI. En la Tabla A2 se muestra la regla de codificación para el código B3ZS.

CÓDIGO HDB3 (*High Density Bipolar orden* 3)

El código HDB3 (código bipolar de alta densidad de orden 3) no admite más de tres ceros consecutivos. De tener más de tres ceros consecutivos, el cuarto cero se lo sustituye por un pulso de violación. Si se tienen 1L, estos siguen la regla de codificación de AMI. La regla para la codificación se puede observar en la tabla A3.

CÓDIGO MANCHESTER DIFERENCIAL

La manera de codificar la información en Manchester Diferencial es tener transiciones a medio período de bit tanto para el 1L como para el 0L. Además cuando se tiene presente un 0L también se debe realizar una transición al inicio del período de bit. De lo dicho anteriormente se puede decir que cuando se tiene presente un 1L cambia su estado anterior y si se tiene un 0L mantiene su estado anterior.

Polaridad del pulso precedente	**Número de 1L desde la última sustitución**	
	IMPAR	PAR
-	00-	+0+
+	00+	-0-

Tabla A2 Regla de codificación[1]

REGLA DE CODIFICACIÓN HDB3		
Polaridad del pulso precedente	**Números de unos lógicos desde la última sustitución**	
	IMPAR	PAR
+	000-	+00+
-	000+	-00-

Tabla A3 Regla de codificación HDB3[2].

CÓDIGO BIFASE M

Es un código de dos niveles +A y –A. Una transición aparece siempre al principio del intervalo. El símbolo 1L produce otra transición a mitad del intervalo Tb/2, en tanto que el símbolo 0L no produce transición.

CÓDIGO DE MODULACIÓN POR RETARDO (DM) o CÓDIGO MILLER

Este código presenta una transición entre dos niveles de la misma amplitud y con polaridad contraria (+A y -A). El símbolo uno lógico produce una transición en el punto medio del periodo, mientras que el cero lógico no produce ninguna transición, a no ser que vaya seguido por otro cero lógico en cuyo caso se produce una transición entre los dos ceros al final del primer periodo. Es eficiente

[1 y 2] Tabla tomada del Folleto de Comunicación Digital realizado por el Ing. Pablo Hidalgo

en términos de ancho de banda cuando no se requiere de una buena respuesta en bajas frecuencia.

CÓDIGO 4B3T (4 *BINARY* – 3 *TERNARY*)

En este código se hace corresponder cuatro señales binarias a tres señales ternarias, así pues el codificador 4B3T convierte grupos de 4 dígitos binarios en grupos de tres dígitos ternarios. Dado que las palabras binarias de 4 bits solo requieren 16 de las 27 posibles palabras del código ternario de 3 dígitos, existe una cierta flexibilidad en la selección de los códigos ternarios. Esta redundancia permite detectar errores de transmisión.

En la Tabla A4 se presentan la regla de codificación. Las palabras del código se eligen para mantener el balance de DC. Si se transmiten mas pulsos positivos que negativos, se debe seleccionar el modo negativo. Cuando la polaridad entre pulsos positivos y negativos cambia (mas pulsos negativos) se debe elegir el modo positivo.

Este código determina la disminución de la velocidad de la señal codificada en un factor de 3/4. En la Figura A1 se observa el diagrama de estados del codificador 4B3T.

Entrada Binaria (4B)	Palabra Ternaria 3T		Disparidad de la palabra código
	Modo Positivo	Modo Negativo	
0000	0 - +	0 - +	0
0001	- + 0	- + 0	0
0010	- 0 +	- 0 +	0
0011	+ - +	- + -	+1,-1
0100	0 + +	0 - -	+2,-2
0101	0 + 0	0 - 0	+1,-1
0110	0 0 +	0 0 -	+1,-1
0111	- + +	+ - -	+1,-1
1000	0 + -	0 + -	0
1001	+ - 0	+ - 0	0
1010	+ 0 -	+ 0 -	0
1011	+ 0 0	- 0 0	+1,-1
1100	+ 0 +	- 0 -	+2,-2
1101	+ + 0	- - 0	+2,-2
1110	+ + -	- - +	+1,-1
1111	+ + +	- - -	+3,-3

Tabla A4 Regla de codificación 4B3T[1].

[1 y 2] Tomadas del Folleto de Comunicación Digital realizado por el Ing. Pablo Hidalgo

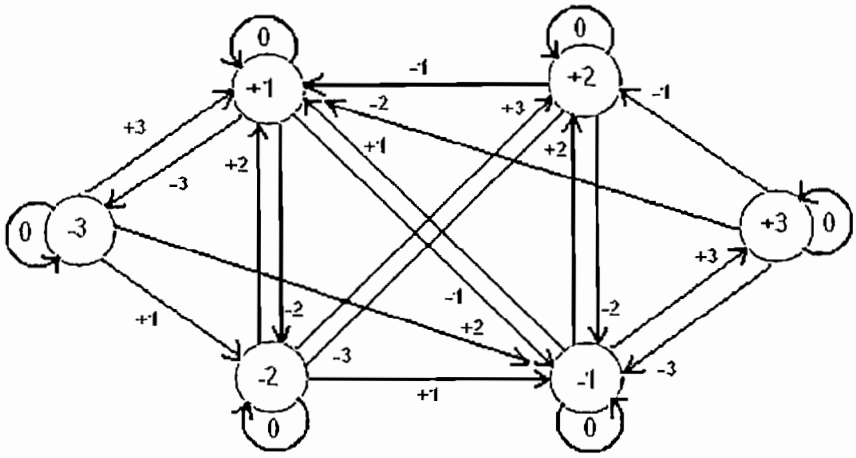


Figura A1 Diagrama de estados de codificación $4B3T^2$.

Es importante indicar que el diagrama de estado no contiene ningún estado de polaridad 0; de igual manera se ve que la polaridad acumulada siempre permanecerá en el rango -3 a +3, esto en la practica no siempre ocurre por errores ocasionales de transmisión.

CÓDIGO MS43 (*MONITORING STATE 43*)

El código MS43 (*Monitoring State 43*) fue adoptado por la Administración Alemana de Telecomunicaciones para las redes digitales de servicios integrados (ISDN-RDSI).

El MS43 tiene un concepto similar al 4B3T descrito anteriormente. En este caso 4 dígitos binarios se codifican para dar una secuencia ternaria de 3 dígitos, con la cual se reduce el ancho de banda ocupado por la señal transmitida.

Existen más de dos modos de codificación que pueden ser seleccionados. En la Tabla A5 se encuentra la regla de codificación del código MS43 donde se tiene las 16 combinaciones de cuatro bits y los cuatro modos posibles de combinaciones ternarias.

La selección entre los diferentes modos se combina de tal forma que el numero de estados +, 0, - sea equiprobable. El diagrama de transición o de estados para el código MS43 esta dado en la Figura A2.

CÓDIGO CMI (*CODED MARK INVERSION*)

Basados en el código Manchester se han desarrollado una gran variedad de códigos de línea; de estos, el código CMI (de inversión de marcas codificadas) es el mas representativo. Este código bipolar de dos niveles o binario responde a la siguiente regla de codificación: el 0L en vez de ser representado por una ausencia de señal, es emitido con un cambio de polaridad negativo a positivo, que se produce a la mitad del periodo Tb/2 del *bit*. El 1L es codificado con pulsos alternados de duración Tb.

Palabras Binarias	Palabras Ternarias				Disparidad de la palabra código
	Modo 1	Modo 2	Modo 3	Modo 4	
0011	0 - +	0 - +	0 - +	0 - +	0
0101	- 0 +	- 0 +	- 0 +	- 0 +	0
0110	- + 0	- + 3	- + 0	- + 0	0
1110	+ - 0	+ - 0	+ - 0	+ - 0	0
1101	+ 0 -	+ 0 -	+ 0 -	+ 0 -	0
1011	0 + -	0 + -	0 + -	0 + -	0
1000	+ - +	+ - +	+ - +	- - -	+1,+1,+1,-3
1001	0 0 +	0 0 +	0 0 +	- - 0	+1,+1,+1,-2
1010	0 + 0	0 + 0	0 + 0	- 0 -	+1,+1,+1,-2
1100	+ 0 0	+ 0 0	+ 0 0	0 - -	+1,+1,+1,-2
0111	- + +	- + +	- - +	- - +	+1,+1,-1,-1
1111	+ + -	+ + -	+ - -	+ - -	+1,+1,-1,-1
0001	+ + 0	0 0 -	0 0 -	0 0 -	+2,-1,-1,-1
0010	+ 0 +	0 - 0	0 - 0	0 - 0	+2,-1,-1,-1
0100	0 + +	- 0 0	- 0 0	- 0 0	+2,-1,-1,-1
0000	+ + +	- + -	- + -	- + -	+3,-1,-1,-1

Tabla A5: Regla de codificación MS43[1].

Este código es equivalente a uno de velocidad doble donde el símbolo 1L se halla codificado alternativamente con 11 y 00, y el símbolo 0L con 01. Aquí se repite la equiprobabilidad de polaridad y la consiguiente minimización de componente continua; al igual que en los códigos HDBn, también se repite la alta densidad bipolar para mantener presente la señal de reloj y la posibilidad de transmitir datos y temporización por el mismo par.

[1 y 2] Tomadas del Folleto de Comunicación Digital realizado por el Ing. Pablo Hidalgo

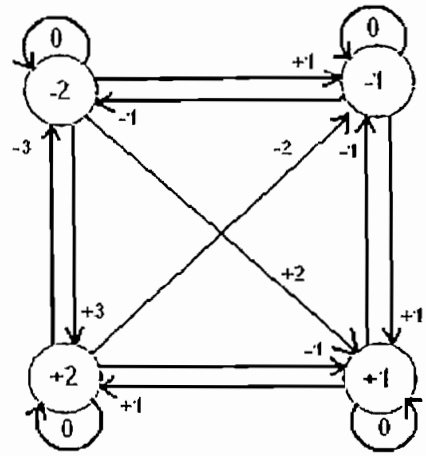


Figura A2: Diagrama de transición del código MS43[2].

CÓDIGO MLT-3 (*Multi – Level Transmit*)

El código de línea es utilizado en algunas redes de área local de alta velocidad como es el caso de las redes 100 Base –TX. El efecto del esquema de codificación MLT-3 es concentrar la mayor parte de la energía en la señal transmitida bajo los 30 MHz, lo cual reduce las emisiones radiadas; esto a su vez reduce los problemas debido a la interferencia.

La codificación MLT-3 produce una salida que tiene una transición para cada 1L, el cual usa tres niveles: un voltaje positivo (+A) y un nivel de voltaje nulo. Los niveles de voltaje para cada 1L son seleccionados de manera secuencial (+A, 0, -A, 0, +A, ...). El *bit* 0L mantiene el nivel del voltaje del estado anterior codificado.

ANEXO B

HOJA DE DATOS DE LOS CIRCUITOS INTEGRADOS UTILIZADOS

CONTENIDO:

Hoja de datos del CI LM318N

Hoja de datos del CI LM319N

Hoja de datos del CI LM310N

Hoja de datos del Módulo USB MOD2

Hoja de datos del CI FT8U245AM

Hoja de datos del PIC 16F870

Hoja de datos del CI 74HC4052

Hoja de datos del CI Microcontrolador AT89C52

Hoja de datos del CI MAX765

Circuito Impreso del Prototipo

Diagrama de conexiones del Prototipo

August 2000

National Semiconductor

LM118/LM218/LM318 Operational Amplifiers

General Description

The LM118 series are precision high speed operational amplifiers designed for applications requiring wide bandwidth and high slew rate. They feature a factor of ten increase in speed over general purpose devices without sacrificing DC performance.

The LM118 series has internal unity gain frequency compensation. This considerably simplifies its application since no external components are necessary for operation. However, unlike most internally compensated amplifiers, external frequency compensation may be added for optimum performance. For inverting applications, feedforward compensation will boost the slew rate to over 150V/µs and almost double the bandwidth. Overcompensation can be used with the amplifier for greater stability when maximum bandwidth is not needed. Further, a single capacitor can be added to reduce the 0.1% settling time to under 1 µs.

The high speed and fast settling time of these op amps make them useful in A/D converters, oscillators, active filters, sample and hold circuits, or general purpose amplifiers. These devices are easy to apply and offer an order of magnitude better AC performance than industry standards such as the LM709.

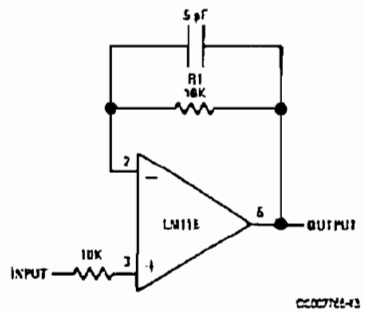
The LM218 is identical to the LM118 except that the LM218 has its performance specified over a −25°C to +85°C temperature range. The LM318 is specified from 0°C to +70°C.

Features

- 15 MHz small signal bandwidth
- Guaranteed 50V/µs slew rate
- Maximum bias current of 250 nA
- Operates from supplies of ±5V to ±20V
- Internal frequency compensation
- Input and output overload protected
- Pin compatible with general purpose op amps

Fast Voltage Follower

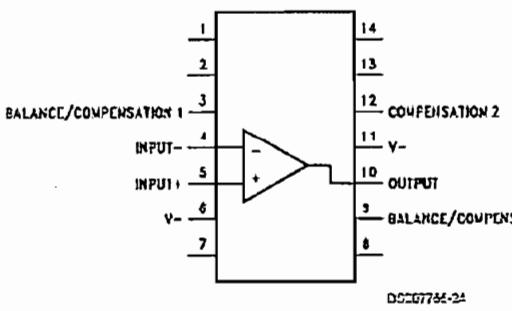
(Note 1)



Note 1: Do not hardwire as voltage follower ($R1 \geq 5\ k\Omega$)

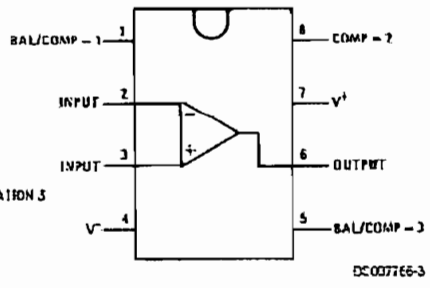
Connection Diagram

Dual-In-Line Package



Top View
Order Number LM118J/883 (Note 13)
See NS Package Number J14A

Dual-In-Line Package



Top View
Order Number LM118J-8/883 (Note 13), LM318M or LM318N
See NS Package Number J08A, M08A or N08B

© 2000 National Semiconductor Corporation

LM119
LM219 - LM319

HIGH SPEED DUAL COMPARATORS

- TWO INDEPENDENT COMPARATORS
- SUPPLY VOLTAGE : +5V TO ±15V
- TYPICALLY 80ns RESPONSE TIME AT ±15V
- MINIMUM FAN-OUT OF 2 EACH SIDE
- MAXIMUM INPUT CURRENT OF 1µA OVER OPERATING TEMPERATURE RANGE
- INPUTS AND OUTPUTS CAN BE ISOLATED FROM SYSTEM GROUND
- HIGH COMMON-MODE SLEW RATE

N
DIP14
(Plastic Package)

D
SO14
(Plastic Micropackage)

DESCRIPTION

These products are precision high speed dual comparators designed to operate over a wide range of supply voltages down to a single 5V logic supply and ground and have low input currents and high gains.

The open collector of the output stage makes compatible with TTL as well as capable of driving lamps and relays at currents up to 25mA.

Although designed primarily for applications requiring operation from digital logic supplies, the comparators are fully specified for power supplies up to ±15V.

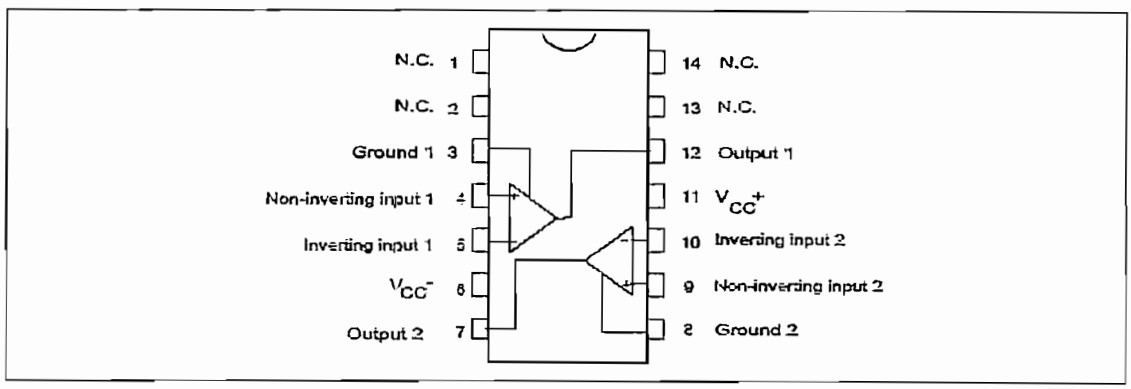
They feature faster response than LM111 at the expense of higher current consumption. However, the high speed, wide operating voltage range and low package count make the much more versatile.

ORDER CODE

Part Number	Temperature Range	Package	
		N	D
LM119	-55°C, +125°C	•	•
LM219	-40°C, +105°C	•	•
LM319	0°C, +70°C	•	•
Example : LM219N			

N = Dual in Line Package (DIP)
D = Small Outline Package (SO) - also available in Tape & Reel (DT)

PIN CONNECTIONS (top view)



National Semiconductor

November 1994

LM110/LM210/LM310 Voltage Follower

General Description

The LM110 series are monolithic operational amplifiers internally connected as unity-gain non-inverting amplifiers. They use super-gain transistors in the input stage to get low bias current without sacrificing speed. Directly interchangeable with 101, 741 and 709 in voltage follower applications, these devices have internal frequency compensation and provision for offset balancing.

The LM110 series are useful in fast sample and hold circuits, active filters, or as general-purpose buffers. Further, the frequency response is sufficiently better than standard IC amplifiers that the followers can be included in the feedback loop without introducing instability. They are plug-in replacements for the LM102 series voltage followers, offering lower offset voltage, drift, bias current and noise in addition to higher speed and wider operating voltage range.

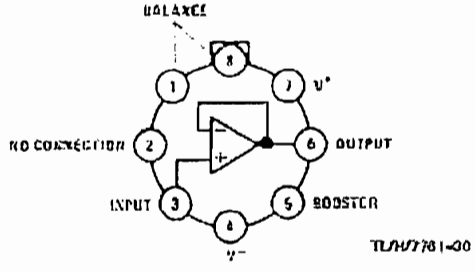
The LM110 is specified over a temperature range $-55°C \leq T_A \leq +125°C$, the LM210 from $-25°C \leq T_A \leq +85°C$ and the LM310 from $0°C \leq T_A \leq +70°C$.

Features

■ Input current	10 nA max over temperature
■ Small signal bandwidth	20 MHz
■ Slew rate	30 V/μs
■ Supply voltage range	±5V to ±18V

Connection Diagrams

Metal Can Package

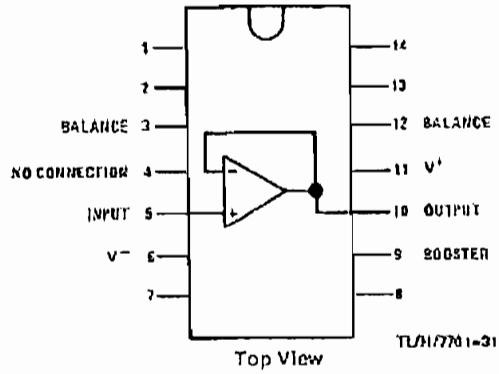


Package is connected to Pin 4 (V⁻)

Top View

Order Number LM110H, LM210H or LM310H
LM110H/883*
See NS Package Number H08C

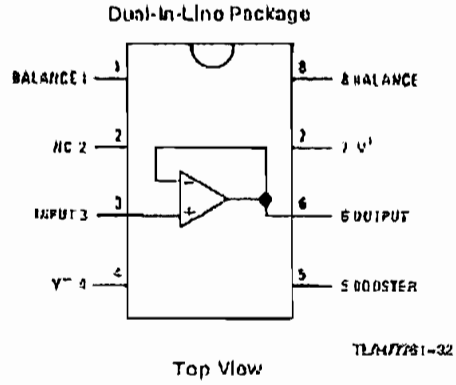
Dual-In-Line Package



Top View

Order Number LM110J, LM210J,
LM310J or LM110J/883*
See NS Package Number J14A

Dual-In-Line Package



Top View

Order Number LM310M, LM310N or LM110J-8/883*
See NS Package Number J06A, M08A or N08E

©1995 National Semiconductor Corporation TL/H/7761 RRD-B30M115/Printed in U. S. A.

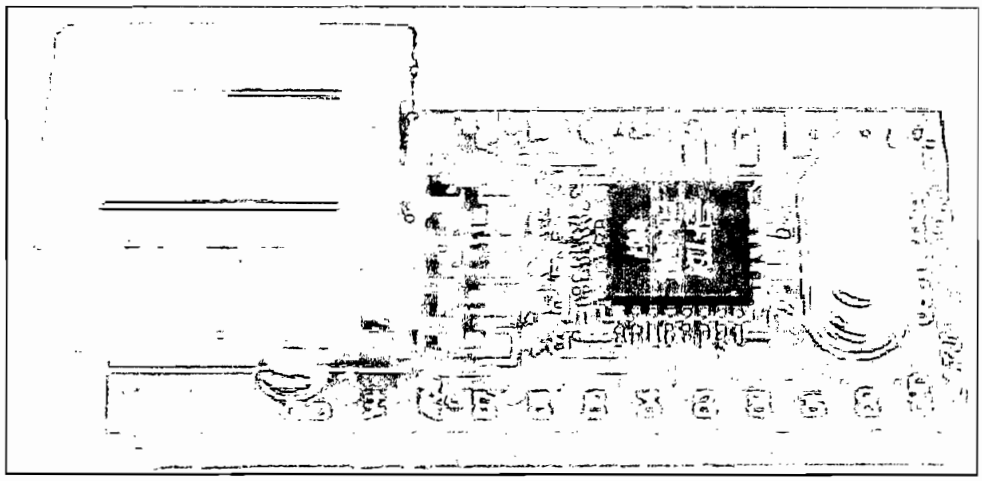
RAVAR

USB MOD2 User's Manual

USB MOD2 - USB Plug and Play Parallel 8-Bit FIFO Development Module

The USBMOD2 is a low-cost integrated module for transferring data to / from a peripheral and a host P.C at up to 8 Million bit (1 Megabyte) per second. Based on the FTDI FT8U245 USB FIFO – Fast Parallel Data Transfer IC, it's simple FIFO-like design makes it easy to interface to an CPU (MCU) either by mapping the device into the memory / I/O map of the PCU, using DMA or controlling the device via IO ports.

The USBMOD2 is ideal for rapid prototyping and development by offering a complete plug and play solution.



MODULE FEATURES

- Single module High-Speed USB UART solution
- Based on FTDI FT8U245 USB FIFO – Fast Parallel Data Transfer IC
- Integrated Type-B USB Connector
- On-board 6MHz Crystal
- Provision for external EEPROM for USB enumeration data
- No external passive components required
- Module powered from USB bus (up to 60mA from USB for user application)
- 32-pin Dual In-Line Package (Ideal for prototyping)
- Fits into a standard 32-pin 600mil IC Socket

USB MOD2 User's Manual

FT8U245 IC FEATURES

- Single Chip Multi-Function Data Transfer Solution
- Send / Receive Data over USB at up to 1 Mb / Sec
- 384 byte receive buffer / 128 byte transmit buffer for high data throughput
- Simple interface to CPU or MCU bus
- No in-depth knowledge of USB required as all USB Protocol is handled automatically within the I.C
- FTDI's Virtual COM port drivers eliminate the need for USB driver development in most cases.
- Compact 32 pin (7mm x 7mm) MQFP package
- Integrated 6Mhz – 48Mhz Clock Multiplier aids FCC and CE compliance
- Integrated 3.3v Regulator – No External Regulator Required
- UHCI / OHCI Compliant
- USB 1.1 Specification Compliant
- USB VID, PID, Serial Number and Product Description Strings in external E2PROM.

VIRTUAL COM PORT (VCP)
DRIVERS for

- Windows 98 and Windows 98 SE
- Windows 2000 / ME / XP
- Windows CE **
- MAC OS-8 and OS9
- MAC OS-X
- Linux 2.40 and greater

[** = In the planning or under development]

D2XX
(USB Direct Drivers + DLL S/W Interface)

- Windows 98 and Windows 98 SE
- Windows 2000 / ME / XP

For further information regarding the FTDI FT8U245AM USB FIFO – Fast Parallel Data Transfer IC please refer to the FT8U245AM Datasheet. This datasheet can be found on the Ravar website at **http://www.ravar.net**

USB MOD2 User's Manual

As mentioned above in module features, the USB MOD2 is in a 32-pin Dual In-Line Package. This allows the module to fit into a standard 32-pin 600mil IC Socket which makes the module ideal for prototyping and development work. Shown in diagram 2 below is the pin out for the USB MOD2.

USB MOD2 PINOUT

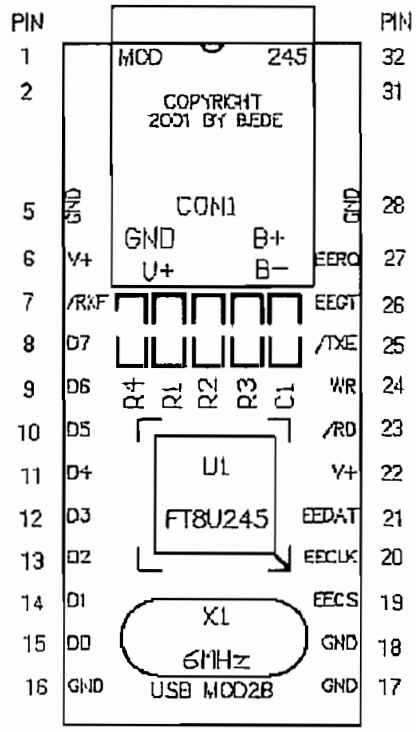


DIAGRAM 2

On the following page is the pin out table showing what the various pins are on the module.

RAVAR

USB MOD2 User's Manual

USBMOD2 PINOUT

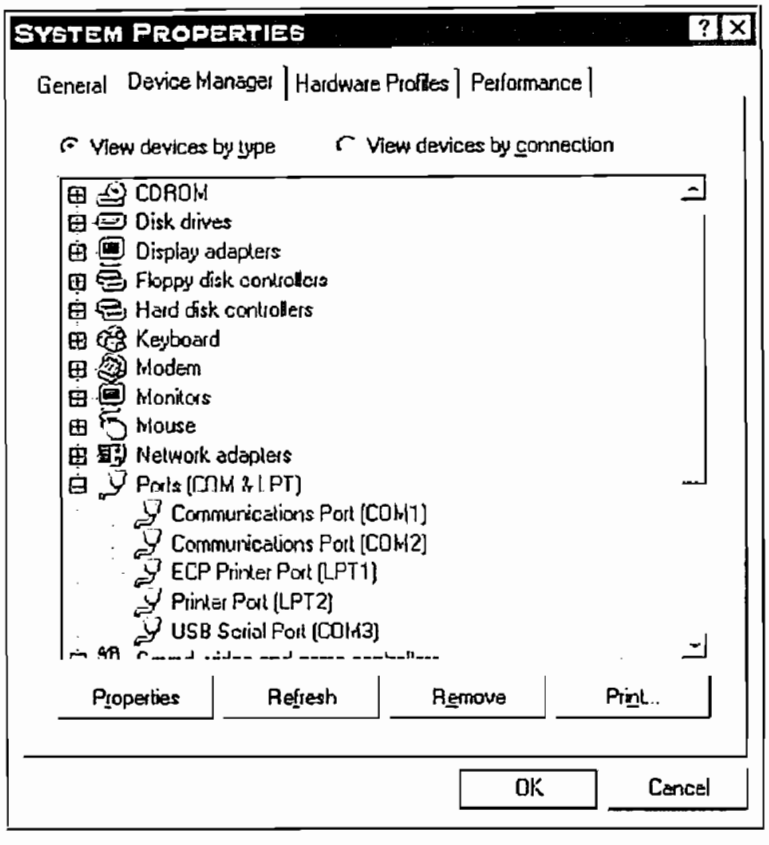
PIN #	SIGNAL	TYPE	DESCRIPTION
1	N/C	MOUNT	Mounting Pin for module USB connector support
2	N/C	MOUNT	Mounting Pin for module USB connector support
3	NO PIN	NO PIN	
4	NO PIN	NO PIN	
5	GND	PWR	Device – Ground Supply Pin
6	V÷	PWR	Device - ÷4.4 volt to ÷5.25 volt Power Supply Pin
7	/RXF	OUT	When high, do not read data from FIFO. When low, there is data available in the FIFO which can be read by strobing RD# low the high again.
8	D7	I/O	Bi-Directional Data Bus Bit #7
9	D6	I/O	Bi-Directional Data Bus Bit #6
10	D5	I/O	Bi-Directional Data Bus Bit #5
11	D4	I/O	Bi-Directional Data Bus Bit #4
12	D3	I/O	Bi-Directional Data Bus Bit #3
13	D2	I/O	Bi-Directional Data Bus Bit #2
14	D1	I/O	Bi-Directional Data Bus Bit #1
15	D0	I/O	Bi-Directional Data Bus Bit #0
16	GND	PWR	Device – Ground Supply Pin
17	GND	PWR	Device – Ground Supply Pin
18	GND	PWR	Device – Ground Supply Pin
19	EECS	I/O	Optional EEPROM – Chip Select
20	EECLK	I/O	Optional EEPROM – Clock
21	EEDAT	I/O	Optional EEPROM – Data I/O
22	V÷	PWR	Device - ÷4.4 volt to ÷5.25 volt Power Supply Pin
23	/RD	IN	Enables Current FIFO Data Byte on D0..D7 when low. Fetches the next FIFO Data Byte (if available) from the Receive FIFO Buffer when /RD goes from low to high.
24	WR	IN	Writes the Data Byte on D0..D7 into the Transmit FIFO Buffer when WR goes from high to low.
25	/TXE	OUT	When high, do not write data into the FIFO. When low, data can be written into the FIFO by strobing WR high then low.
26	EEGT	OUT	When low, allows the EEPROM contents to be accessed via the Data Bus
27	EERQ	IN	Requests the EEPROM contents to be accessed via the Data Bus.
28	GND	PWR	Device – Ground Supply Pin
29	NO PIN	NO PIN	
30	NO PIN	NO PIN	
31	N/C	MOUNT	Mounting Pin for module USB connector support
32	N/C	MOUNT	Mounting Pin for module USB connector support

USB MOD2 User's Manual

Driver Installation.

Your first choice when using the USBMOD is whether you want to use the Virtual Com Port driver or the Direct DLL driver.

For programming simplicity the best driver is the Virtual Com Port and when installed the USBMOD will appear in the System Properties / Device Manager as an USB Serial Port (COMn) as follows.



The Com port number will vary depending on the number of existing com ports on your computer and the number of USBMODs connected to your system.

To install the Virtual Com Port drivers, download the driver from our website or the ftdichip.com website and unzip the files to a local directory. Then connect the USBMOD and windows will automatically ask for the driver, select to specify a location and browse to the directory where you have unzipped the files. (Use the Non Plug & Play driver for the USBMOD to avoid a delay on connecting the USBMOD)

Once the Virtual Com Port is installed it can be programmed exactly as a regular serial com port using the MSComm control from Visual Basic or API calls from C or other languages. Set the com port to the same number as appears in the Device Manager, the baud rate, stop bits, parity etc are not used as the device always runs at full speed.

The Direct DLL driver is installed in a similar manner but using the alternative download from the website.

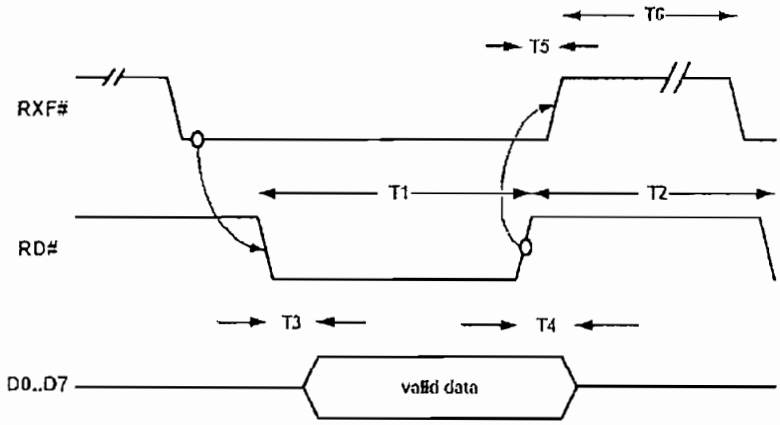
Programming the Direct DLL driver is by call to the DLL Library functions.
Please download the Direct DLL programmers guide from the Ravar website.

Application Notes

On the following pages there are schematic drawings showing a sample applications for the USB MOD2.

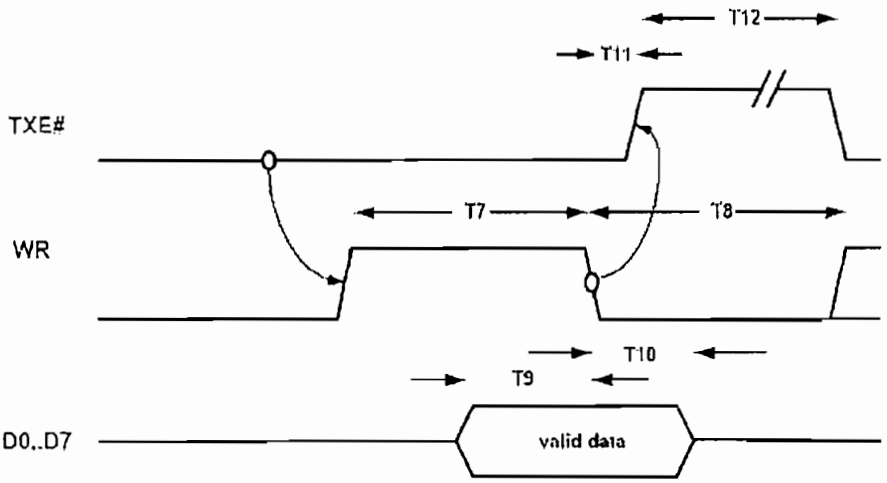
The application uses a micro controller, in this case we are using a PIC 16C84 as well as a 93C46 EEPROM chip.

FT8U245AM TIMING DIAGRAM – FIFO READ CYCLE



Time	Description	Min	Max	Unit
T1	RD Active Pulse Width	50		ns
T2	RD to RD Pre-Charge Time	50		ns
T3	RD Active to Valid Data		30	ns
T4	Valid Data Hold Time from RD inactive	10		ns
T5	RD Inactive to RXF#	5	25	ns
T6	RXF inactive after RD cycle	80		ns

FT8U245AM TIMING DIAGRAM – FIFO WRITE CYCLE



Time	Description	Min	Max	Unit
T7	WR Active Pulse Width	50		ns
T8	WR to WR Pre-Charge Time	50		ns
T9	Data Setup Time before WR inactive		20	ns
T10	Data Hold Time from WR inactive	10		ns
T11	WR Inactive to TXE#	5	25	ns
T12	TXE Inactive after RD cycle	80		ns

MICROCHIP

PIC16F870/871

28/40-Pin, 8-Bit CMOS FLASH Microcontrollers

Devices Included in this Data Sheet:

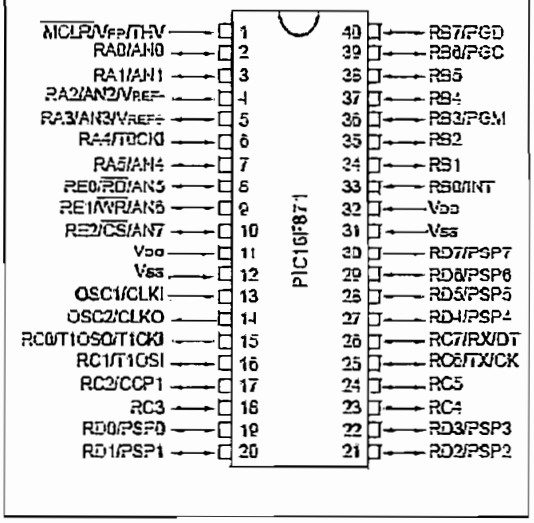
- PIC16F870
- PIC16F871

Microcontroller Core Features:

- High performance RISC CPU
- Only 35 single word instructions to learn
- All single cycle instructions except for program branches which are two-cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200 ns instruction cycle
- 2K x 14 words of FLASH Program Memory
  128 x 8 bytes of Data Memory (RAM)
  64 x 8 bytes of EEPROM Data Memory
- Pinout compatible to the PIC16CXXX 28 and 40-pin devices
- Interrupt capability (up to 11 sources)
- Eight level deep hardware stack
- Direct, Indirect and Relative Addressing modes
- Power-on Reset (POR)
- Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code protection
- Power saving SLEEP mode
- Selectable oscillator options
- Low power, high speed CMOS FLASH/EEPROM technology
- Fully static design
- In-Circuit Serial Programming™ (ICSP™) via two pins
- Single 5V In-Circuit Serial Programming capability
- In-Circuit Debugging via two pins
- Processor read/write access to program memory
- Wide operating voltage range: 2.0V to 5.5V
- High Sink/Source Current: 25 mA
- Commercial and Industrial temperature ranges
- Low power consumption:
  - < 1.6 mA typical @ 5V, 4 MHz
  - 20 µA typical @ 3V, 32 kHz
  - < 1 µA typical standby current

Pin Diagram

PDIP

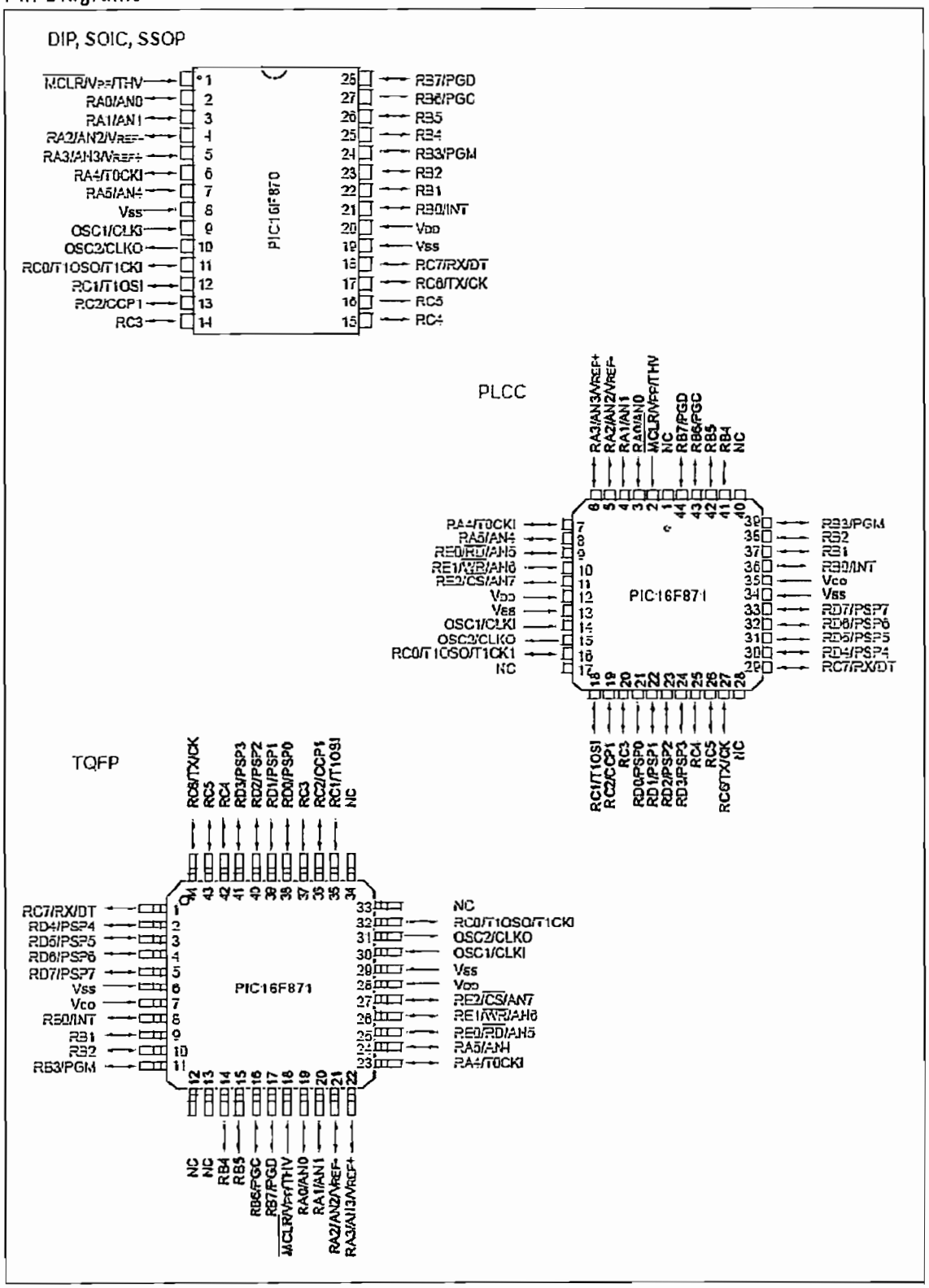


Peripheral Features:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during SLEEP via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- One Capture, Compare, PWM module
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- 10-bit multi-channel Analog-to-Digital converter
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) 8-bits wide, with external RD, WR and CS controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)

PIC16F870/871

Pin Diagrams



12.0 INSTRUCTION SET SUMMARY

Each PIC16F870/871 instruction is a 14-bit word, divided into an OPCODE, which specifies the instruction type, and one or more operands, which further specify the operation of the instruction. The PIC16F870/871 instruction set summary in Table 12-2 lists byte-oriented, bit-oriented, and literal and control operations. Table 12-1 shows the opcode field descriptions.

For byte-oriented instructions, 'f' represents a file register designator and 'd' represents a destination designator. The file register designator specifies which file register is to be used by the instruction.

The destination designator specifies where the result of the operation is to be placed. If 'd' is zero, the result is placed in the W register. If 'd' is one, the result is placed in the file register specified in the instruction.

For bit-oriented instructions, 'b' represents a bit field designator which selects the number of the bit affected by the operation, while 'f' represents the address of the file in which the bit is located.

For literal and control operations, 'k' represents an eight or eleven bit constant or literal value.

TABLE 12-1: OPCODE FIELD DESCRIPTIONS

Field	Description
f	Register file address (0x00 to 0x7F)
W	Working register (accumulator)
b	Bit address within an 8-bit file register
k	Literal field, constant data or label
x	Don't care location (= 0 or 1). The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.
d	Destination select; d = 0: store result in W, d = 1: store result in file register f. Default is d = 1.
PC	Program Counter
TO	Time-out bit
PD	Power-down bit

The instruction set is highly orthogonal and is grouped into three basic categories:

- Byte-oriented operations
- Bit-oriented operations
- Literal and control operations

All instructions are executed within one single instruction cycle, unless a conditional test is true, or the program counter is changed as a result of an instruction. In this case, the execution takes two instruction cycles, with the second cycle executed as a NOP. One instruction cycle consists of four oscillator periods. Thus, for an oscillator frequency of 4 MHz, the normal instruction execution time is 1 µs. If a conditional test is true, or the program counter is changed as a result of an instruction, the instruction execution time is 2 µs.

Table 12-2 lists the instructions recognized by the MPASM™ assembler.

Figure 12-1 shows the general formats that the instructions can have.

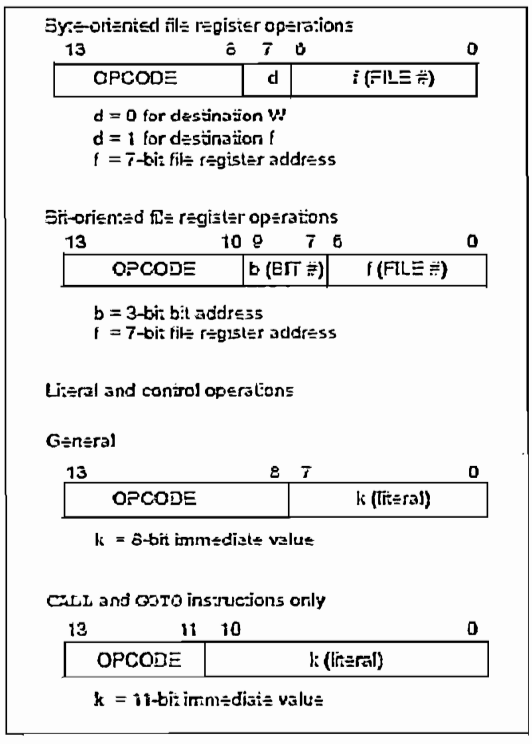
Note:	To maintain upward compatibility with future PIC16F870/871 products, do not use the OPTION and TRIS instructions.

All examples use the following format to represent a hexadecimal number:

0xhh

where h signifies a hexadecimal digit.

FIGURE 12-1: GENERAL FORMAT FOR INSTRUCTIONS



A description of each instruction is available in the PICmicro™ Mid-Range MCU Family Reference Manual (DS33023).

PIC16F870/871

TABLE 12-2: PIC16F870/871 INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb			LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS									
ADDWF	f, d	Add W and f	1	00	0111	dfff	ffff	C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00	0101	dfff	ffff	Z	1,2
CLRF	f	Clear f	1	00	0001	lfff	ffff	Z	2
CLRW	-	Clear W	1	00	0001	0xxx	xxxx	Z	
COMF	f, d	Complement f	1	00	1001	dfff	ffff	Z	1,2
DECF	f, d	Decrement f	1	00	0011	dfff	ffff	Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00	1011	dfff	ffff		1,2,3
INCF	f, d	Increment f	1	00	1010	dfff	ffff	Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00	1111	dfff	ffff		1,2,3
IORWF	f, d	Inclusive OR W with f	1	00	0100	dfff	ffff	Z	1,2
MOVF	f, d	Move f	1	00	1000	dfff	ffff	Z	1,2
MOVWF	f	Move W to f	1	00	0000	lfff	ffff		
NOP	-	No Operation	1	00	0000	0xx0	0000		
RLF	f, d	Rotate Left f through Carry	1	00	1101	dfff	ffff	C	1,2
RRF	f, d	Rotate Right f through Carry	1	00	1100	dfff	ffff	C	1,2
SUBWF	f, d	Subtract W from f	1	00	0010	dfff	ffff	C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00	1110	dfff	ffff		1,2
XORWF	f, d	Exclusive OR W with f	1	00	0110	dfff	ffff	Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS									
BCF	f, b	Bit Clear f	1	01	00bb	bfff	ffff		1,2
BSF	f, b	Bit Set f	1	01	01bb	bfff	ffff		1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01	10bb	bfff	ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01	11bb	bfff	ffff		3
LITERAL AND CONTROL OPERATIONS									
ADDLW	k	Add literal and W	1	11	111x	kkkk	kkkk	C,DC,Z	
ANDLW	k	AND literal with W	1	11	1001	kkkk	kkkk	Z	
CALL	k	Call subroutine	2	10	0kkk	kkkk	kkkk		
CLRWDT	-	Clear Watchdog Timer	1	00	0000	0110	0100	$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to address	2	10	1kkk	kkkk	kkkk		
IORLW	k	Inclusive OR literal with W	1	11	1000	kkkk	kkkk	Z	
MOVLW	k	Move Literal to W	1	11	00xx	kkkk	kkkk		
RETFIE	-	Return from interrupt	2	00	0000	0000	1001		
RETLW	k	Return with literal in W	2	11	01xx	kkkk	kkkk		
RETURN	-	Return from Subroutine	2	00	0000	0000	1000		
SLEEP	-	Go into Standby mode	1	00	0000	0110	0011	$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from literal	1	11	110x	kkkk	kkkk	C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11	1010	kkkk	kkkk	Z	

Note 1: When an I/O register is modified as a function of itself (e.g., MOVF PORTB, 1), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared if assigned to the Timer0 module.

3: If Program Counter (PC) is modified, or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.

Note:	Additional information on the mid-range instruction set is available in the PICmicro™ Mid-Range MCU Family Reference Manual (DS33023).

Dual 4-channel analog multiplexer, demultiplexer

74HC4052; 74HCT4052

FEATURES

- Wide analog input voltage range from −5 V to +5 V
- Low ON-resistance:
  - 80 Ω (typical) at $V_{CC} - V_{EE} = 4.5$ V
  - 70 Ω (typical) at $V_{CC} - V_{EE} = 6.0$ V
  - 60 Ω (typical) at $V_{CC} - V_{EE} = 9.0$ V
- Logic level translation: to enable 5 V logic to communicate with ±5 V analog signals
- Typical "break before make" built in
- Complies with JEDEC standard no. 8-1 A
- ESD protection:
  - HBM EIA/JESD22-A114-A exceeds 2000 V
  - MM EIA/JESD22-A115-A exceeds 200 V.
- Specified from −40 to +85 °C and −40 to +125 °C.

APPLICATIONS

- Analog multiplexing and demultiplexing
- Digital multiplexing and demultiplexing
- Signal gating.

DESCRIPTION

The 74HC4052/74HCT4052 are high-speed Si-gate CMOS devices and are pin compatible with the HEF4052B. They are specified in compliance with JEDEC standard no. 7A.

The 74HC4052/74HCT4052 are dual 4-channel analog multiplexers or demultiplexers with common select logic. Each multiplexer has four independent inputs/outputs (pins nY0 to nY3) and a common input/output (pin nZ). The common channel select logics include two digital select inputs (pins S0 and S1) and an active LOW enable input (pin $\overline{E}$). When pin $\overline{E}$ = LOW, one of the four switches is selected (low-impedance ON-state) with pins S0 and S1. When pin $\overline{E}$ = HIGH, all switches are in the high-impedance OFF-state, independent of pins S0 and S1.

V_{CC} and GND are the supply voltage pins for the digital control inputs (pins S0, S1, and $\overline{E}$). The V_{CC} to GND ranges are 2.0 to 10.0 V for 74HC4052 and 4.5 to 5.5 V for 74HCT4052. The analog inputs/outputs (pins nY0 to nY3 and nZ) can swing between V_{CC} as a positive limit and V_{EE} as a negative limit. $V_{CC} - V_{EE}$ may not exceed 10.0 V.

For operation as a digital multiplexer/demultiplexer, V_{EE} is connected to GND (typically ground).

FUNCTION TABLE

INPUT(1)			CHANNEL BETWEEN
$\overline{E}$	S1	S0	
L	L	L	nY0 and nZ
L	L	H	nY1 and nZ
L	H	L	nY2 and nZ
L	H	H	nY3 and nZ
H	X	X	none

Note

1. H = HIGH voltage level
   L = LOW voltage level
   X = don't care.

PINNING

PIN	SYMBOL	DESCRIPTION
1	2Y0	independent input or output
2	2Y2	independent input or output
3	2Z	common input or output
4	2Y3	independent input or output
5	2Y1	independent input or output
6	$\overline{E}$	enable input (active LOW)
7	V_{EE}	negative supply voltage
8	GND	ground (0 V)
9	S1	select logic input
10	S0	select logic input
11	1Y3	independent input or output
12	1Y0	independent input or output
13	1Z	common input or output
14	1Y1	independent input or output
15	1Y2	independent input or output
16	V_{CC}	positive supply voltage

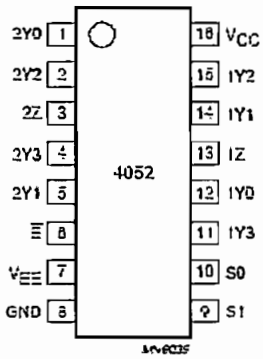
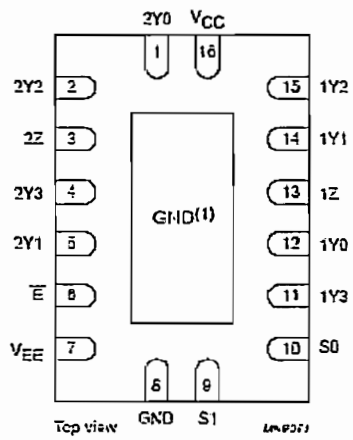


Fig.1 Pin configuration DIP16, SO16 and (T)SSOP16.



(1) The die substrate is attached to this pad using conductive die attach material. It can not be used as a supply pin or input.

Fig.2 Pin configuration DHVQFN16.

Features

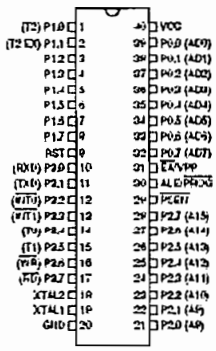
- Compatible with MCS-51™ Products
- 8K Bytes of In-System Reprogrammable Flash Memory
- Endurance: 1,000 Write/Erase Cycles
- Fully Static Operation: 0 Hz to 24 MHz
- Three-level Program Memory Lock
- 256 x 8-bit Internal RAM
- 32 Programmable I/O Lines
- Three 16-bit Timer/Counters
- Eight Interrupt Sources
- Programmable Serial Channel
- Low-power Idle and Power-down Modes

Description

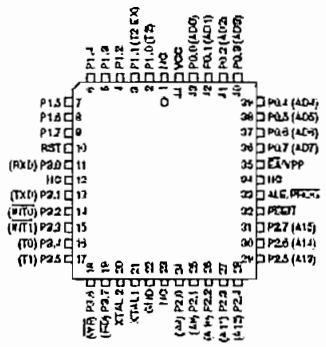
The AT89C52 is a low-power, high-performance CMOS 8-bit microcomputer with 8K bytes of Flash programmable and erasable read only memory (PEROM). The device is manufactured using Atmel's high-density nonvolatile memory technology and is compatible with the industry-standard 80C51 and 80C52 instruction set and pinout. The on-chip Flash allows the program memory to be reprogrammed in-system or by a conventional nonvolatile memory programmer. By combining a versatile 8-bit CPU with Flash on a monolithic chip, the Atmel AT89C52 is a powerful microcomputer which provides a highly-flexible and cost-effective solution to many embedded control applications.

Pin Configurations

PDIP



PLCC



8-bit Microcontroller with 8K Bytes Flash

AT89C52

Rev. 0313H-0200

19-0175 Rev 0; 6/94

Evaluation Kit Available

MAXIM

-5V/-12V/-15V or Adjustable, High-Efficiency, Low IQ DC-DC Inverters

General Description

The MAX764/MAX765/MAX766 inverting switching regulators are highly efficient over a wide range of load currents, delivering up to 1.5W. A unique, current-limited, pulse-frequency-modulated (PFM) control scheme combines the benefits of traditional PFM converters with the benefits of pulse-width-modulated (PWM) converters. Like PWM converters, the MAX764/MAX765/MAX766 are highly efficient at heavy loads. Yet because they are PFM devices, they use less than 120µA of supply current (vs. 2mA to 10mA for a PWM device).

The input voltage range is 3V to 16V. The output voltage is preset at -5V (MAX764), -12V (MAX765), or -15V (MAX766); it can also be adjusted from -1V to -16V using two external resistors (Dual Mode™). The maximum operating VIN - VOUT differential is 20V.

These devices use miniature external components; their high switching frequencies (up to 300kHz) allow for less than 5mm diameter surface-mount magnetics. A standard 47µH inductor is ideal for most applications, so no magnetics design is necessary.

An internal power MOSFET makes the MAX764/MAX765/MAX766 ideal for minimum component count, low- and medium-power applications. For increased output drive capability or higher output voltages, use the MAX774/MAX775/MAX776 or MAX1774, which drive an external power P-channel MOSFET for loads up to 5W.

Applications

LCD-Bias Generators
Portable Instruments
LAN Adapters
Remote Data-Acquisition Systems
Battery-Powered Applications

Features

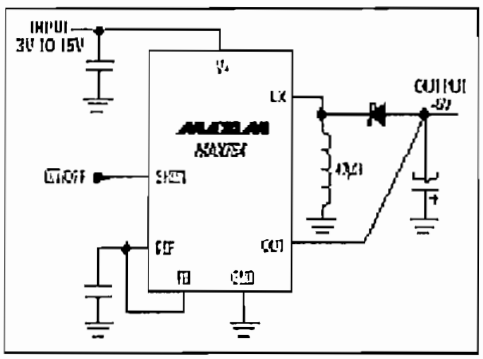
- High Efficiency for a Wide Range of Load Currents
- 250mA Output Current
- 120µA Max Supply Current
- 5µA Max Shutdown Current
- 3V to 16V Input Voltage Range
- -5V (MAX764), -12V (MAX765), -15V (MAX766), or Adjustable Output from -1V to -16V
- Current-Limited PFM Control Scheme
- 300kHz Switching Frequency
- Internal, P-Channel Power MOSFET

Ordering Information

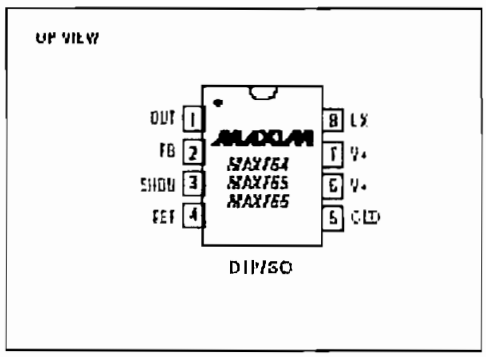
PART	TEMP. RANGE	PIN-PACKAGE
MAX764CPA	0°C to +70°C	8 Plastic DIP
MAX764CSA	0°C to +70°C	8 SO
MAX764C/D	0°C to +70°C	Dice*
MAX764EPA	-40°C to +85°C	8 Plastic DIP
MAX764ESA	-40°C to +85°C	8 SO
MAX764MJA	-55°C to +125°C	8 CERDIP**
MAX765CPA	0°C to +70°C	8 Plastic DIP
MAX765CSA	0°C to +70°C	8 SO
MAX765C/D	0°C to +70°C	Dice*
MAX765EPA	-40°C to +85°C	8 Plastic DIP
MAX765ESA	-40°C to +85°C	8 SO
MAX765MJA	-55°C to +125°C	8 CERDIP**

Ordering information continued on last page.
* Dice are tested at T_A = +25°C. DC parameters only.
**Contact factory for availability and processing to MIL-STD-883.

Typical Operating Circuit



Pin Configuration



Call toll free 1-800-998-8800 for free samples or literature.

-5V/-12V/-15V or Adjustable, High-Efficiency, Low IQ DC-DC Inverters

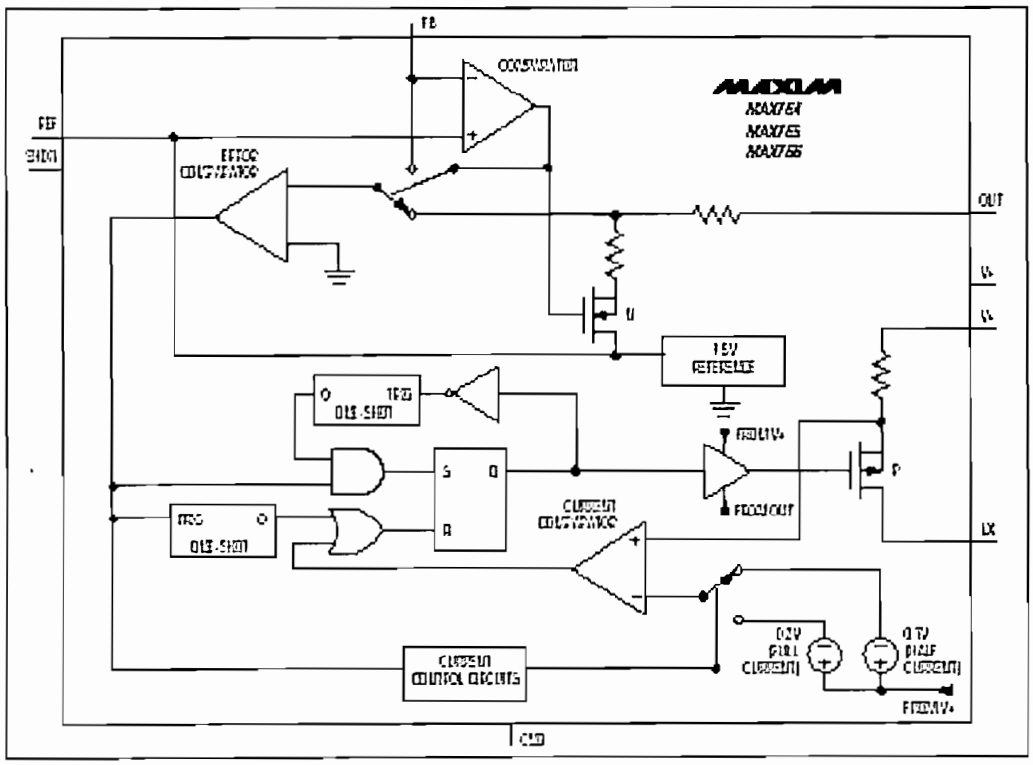


Figure 1. Block Diagram

Detailed Description

Operating Principle

The MAX764/MAX765/MAX766 are BiCMOS, inverting, switch-mode power supplies that provide fixed outputs of -5V, -12V, and -15V, respectively; they can also be set to any desired output voltage using an external resistor divider. Their unique control scheme combines the advantages of pulse-frequency modulation (pulse skipping) and pulse-width modulation (continuous pulsing). The internal P-channel power MOSFET allows peak currents of 0.75A, increasing the output current capability over previous pulse-frequency-modulation (PFM) devices. Figure 1 shows the MAX764/MAX765/MAX766 block diagram.

The MAX764/MAX765/MAX766 offer three main improvements over prior solutions:

1) They can operate with miniature (less than 5mm diameter) surface-mount inductors, because of their 300kHz switching frequency.
2) The current-limited PFM control scheme allows efficiencies exceeding 80% over a wide range of load currents.
3) Maximum quiescent supply current is only 120µA.

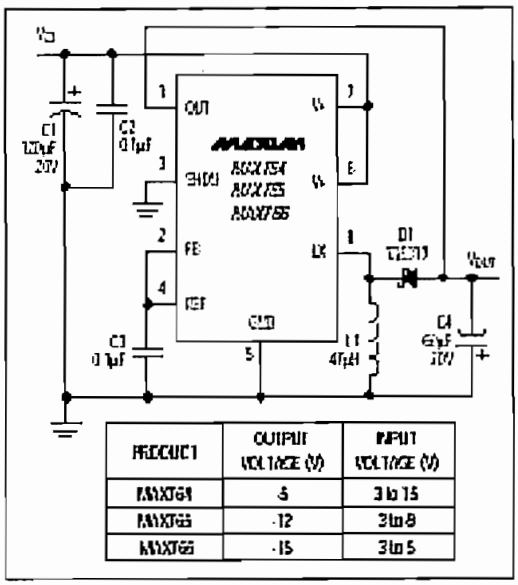
Figures 2 and 3 show the standard application circuits for these devices. In these configurations, the IC is powered from the total differential voltage between the input (V+) and output (V_{OUT}). The principal benefit of this arrangement is that it applies the largest available signal to the gate of the internal P-channel power MOSFET. This increased gate drive lowers switch on-resistance and increases DC-DC converter efficiency.

Since the voltage on the LX pin swings from V+ (when the switch is ON) to |V_{OUT}| plus a diode drop (when the

-5V/-12V/-15V or Adjustable, High-Efficiency, Low IQ DC-DC Inverters

switch is OFF), the range of input and output voltages is limited to a 21V absolute maximum differential voltage.

When output voltages more negative than -16V are required, substitute the MAX764/MAX765/MAX766 with Maxim's MAX774/MAX775/MAX776 or MAX1774, which use an external switch.



PRODUCT	OUTPUT VOLTAGE (V)	INPUT VOLTAGE (V)
MAX764	-5	3 to 15
MAX765	-12	3 to 8
MAX766	-15	3 to 5

Figure 2. Fixed Output Voltage Operation

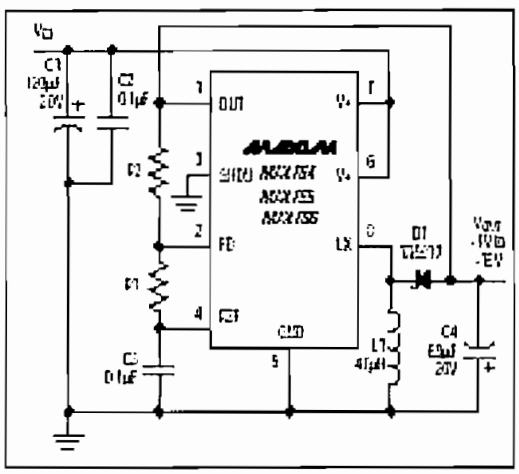


Figure 3. Adjustable Output Voltage Operation

PFM Control Scheme

The MAX764/MAX765/MAX766 use a proprietary, current-limited PFM control scheme that blends the best features of PFM and PWM devices. It combines the ultra-low supply currents of traditional pulse-skipping PFM converters with the high full-load efficiencies of current-mode pulse-width modulation (PWM) converters. This control scheme allows the devices to achieve high efficiencies over a wide range of loads, while the current-sense function and high operating frequency allow the use of miniature external components.

As with traditional PFM converters, the internal power MOSFET is turned on when the voltage comparator senses that the output is out of regulation (Figure 1). However, unlike traditional PFM converters, switching is accomplished through the combination of a peak current limit and a pair of one-shots that set the maximum on-time (16µs) and minimum off-time (2.3µs) for the switch. Once off, the minimum off-time one-shot holds the switch off for 2.3µs. After this minimum time, the switch either 1) stays off if the output is in regulation, or 2) turns on again if the output is out of regulation.

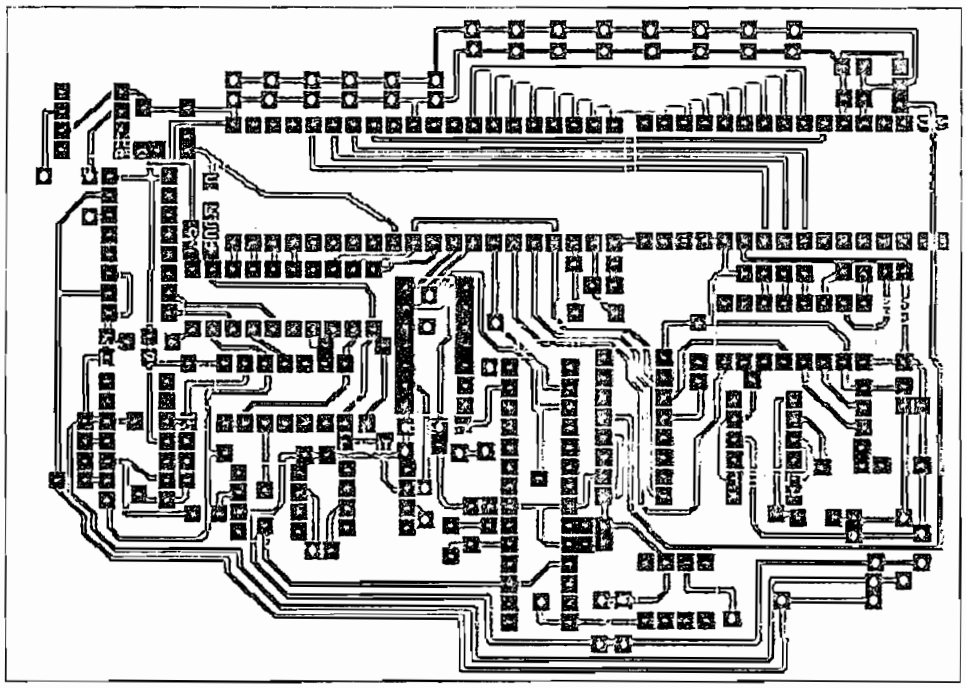
The MAX764/MAX765/MAX766 limit the peak inductor current, which allows them to run in continuous-conduction mode and maintain high efficiency with heavy loads. (See the photo Continuous Conduction at Full Current Limit in the *Typical Operating Characteristics*.) This current-limiting feature is a key component of the control circuitry. Once turned on, the switch stays on until either 1) the maximum on-time one shot turns it off (16µs later), or 2) the current limit is reached.

To increase light-load efficiency, the current limit is set to half the peak current limit for the first two pulses. If those pulses bring the output voltage into regulation, the voltage comparator holds the MOSFET off and the current limit remains at half the peak current limit. If the output voltage is still out of regulation after two pulses, the current limit is raised to its 0.75A peak for the next pulse. (See the photo Discontinuous Conduction at Half and Full Current Limit in the *Typical Operating Characteristics*.)

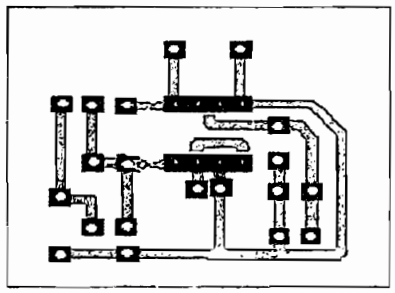
Shutdown Mode

When SHDN is high, the MAX764/MAX765/MAX766 enter a shutdown mode in which the supply current drops to less than 5µA. In this mode, the internal biasing circuitry (including the reference) is turned off and OUT discharges to ground. SHDN is a TTL/CMOS-logic level input. Connect SHDN to GND for normal operation. With a current-limited supply, power-up the device while unloaded or in shutdown mode (hold SHDN high until V+ exceeds 3.0V) to save power and reduce power-up current surges. (See the Supply Current vs. Supply Voltage graph in the *Typical Operating Characteristics*.)

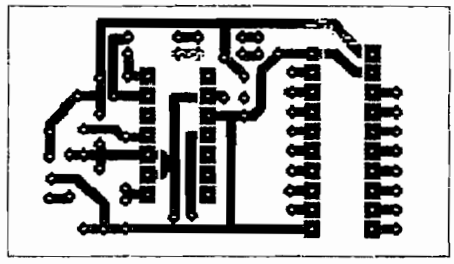
CIRCUITO IMPRESO DEL PROTOTIPO

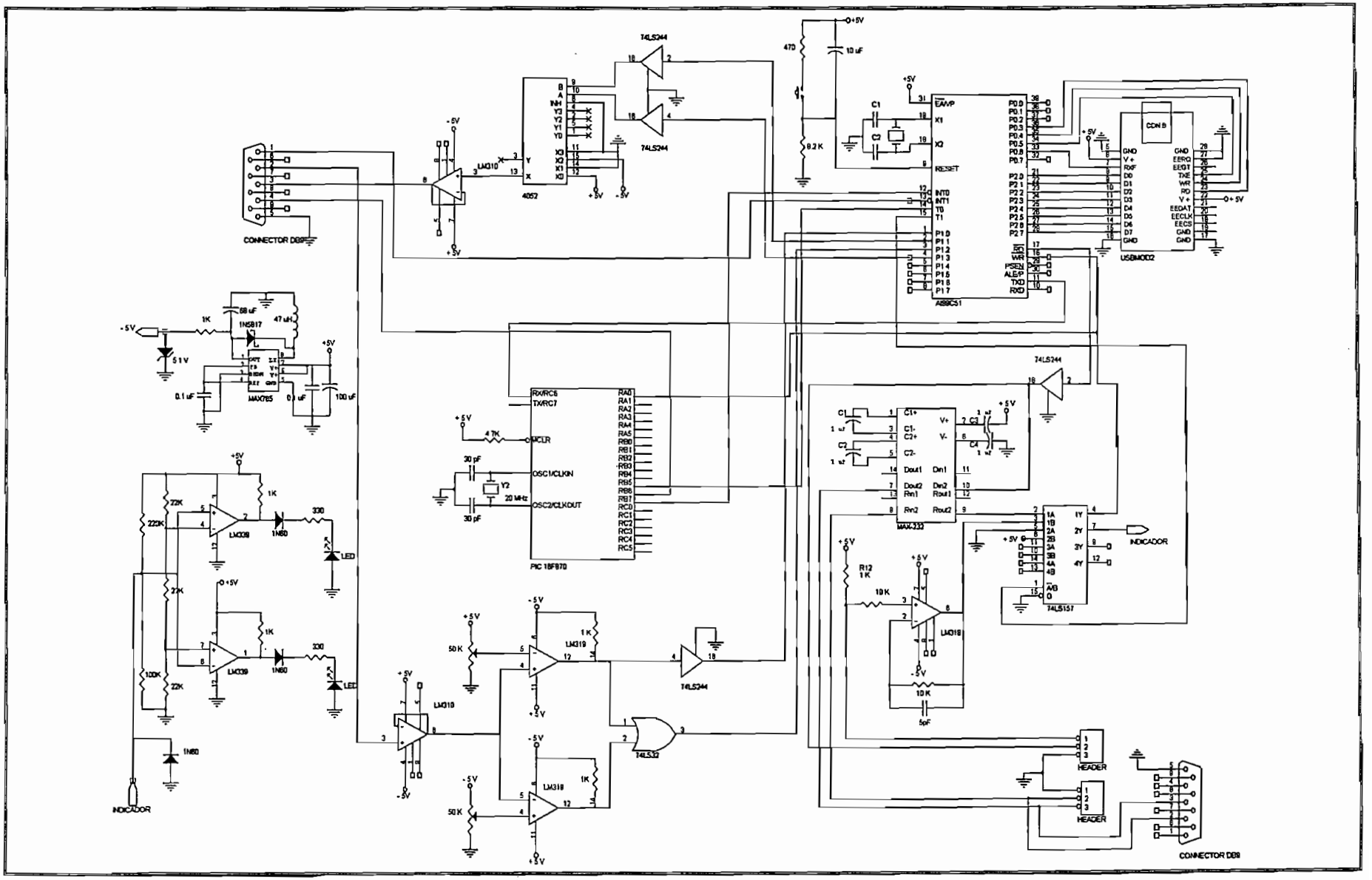


CIRCUITO IMPRESO DE LA FUENTE DE – 5V (MAX765)



CIRCUITO IMPRESO DEL INDICADOR DE NIVELES DE VOLTAJE




74LS244
4052
LM310
CONNECTOR DB9
AS9C51
USBMOD2
CON B
MAX765
1N5817
PIC 18F870
MCLR
OSC1/CLKIN
OSC2/CLKOUT
MAX-232
74LS157
INDICADOR
LM339
LM319
LM318
74LS32
LED
HEADER
CONNECTOR DB9

ANEXO C

MANUAL DE USUARIO DEL PROGRAMA PARA CONFIGURAR EL CODEC PORTÁTIL V 1.0

CONTENIDO:

INSTALACIÓN DEL SOFTWARE CODEC PORTÁTIL.
INSTALACIÓN DEL HARDWARE CODEC PORTÁTIL.
MANUAL DE USO DEL PROGRAMA CODEC PORTÁTIL.

MANUAL DE USUARIO DEL PROGRAMA PARA CONFIGURAR EL CODEC PORTATIL V1.0

Requerimientos del sistema

Para ejecutar el programa el PC debe tener las siguientes características:

- Sistema Operativo Windows 98, Windows 2000 o Windows XP.
- Un Puerto USB.
- Un Puerto serial DB9.
- Memoria RAM de 128Mbyte como mínimo.
- Procesador Pentium Celeron o superior.

Instalación del software

Para instalar CODEC PORTATIL en el PC, deberá tener el archivo ejecutable *Setup.exe* generado a través de Visual Studio *Installer* 1.1. Este archivo lo encuentra en la carpeta Instalador.
Los pasos a seguir en la instalación son los siguientes:

1. Inicie la instalación haciendo doble click en el archivo *Setup.exe*. La ventana que aparece es la siguiente:

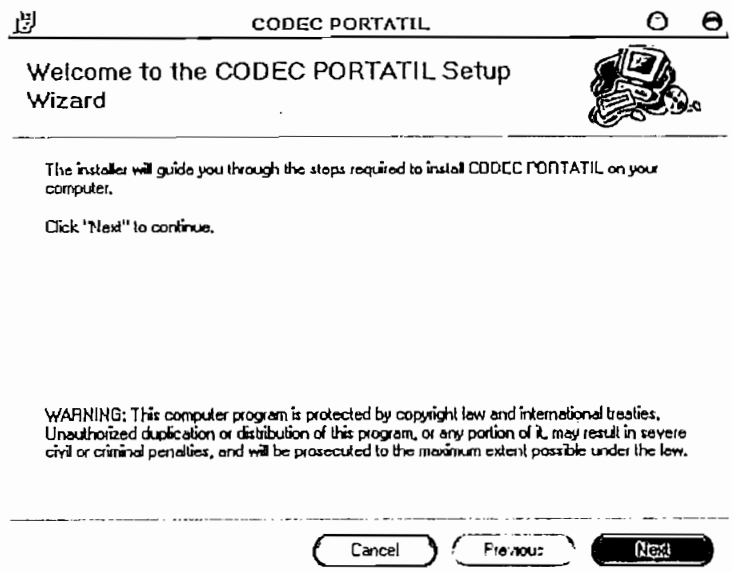


Figura C1 Inicio de la instalación.

2. Hacemos click en *Next:*

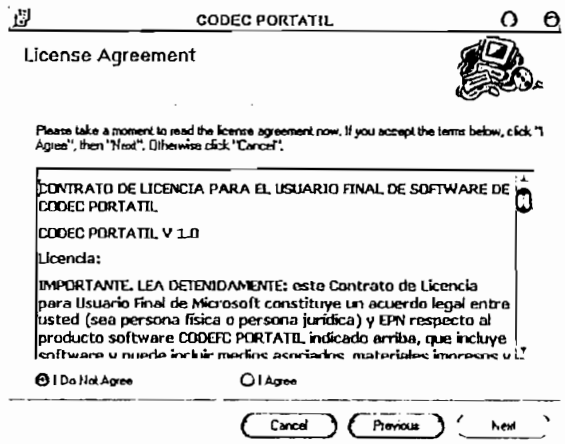


Figura C2 Aceptar Licencia

3. Escoger la opción *I Agree* (aceptar contrato) y hacer un click en *Next.*

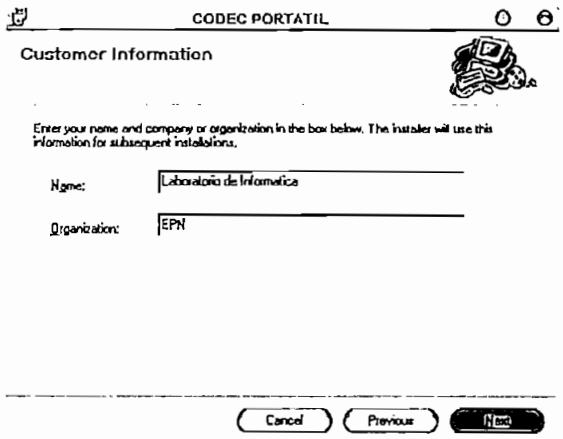


Figura C3 Completar información personal.

4. Completamos la información pedida y hacer click en *Next*

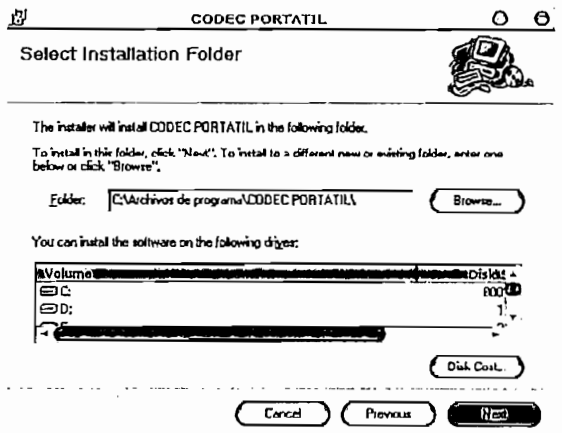


Figura C4 Selección de la carpeta de instalación.

5. Escogemos la ubicación donde se va ha instalar el *software* y hacer click en *Next.*

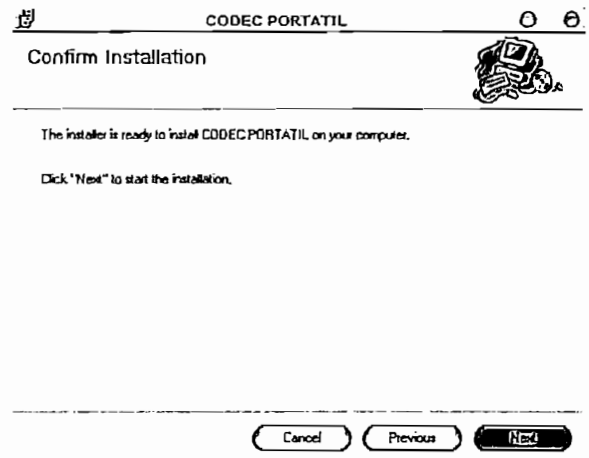


Figura C5 Confirmar instalación

6. Como siguiente paso hacer un click en *Next* para iniciar la instalación.

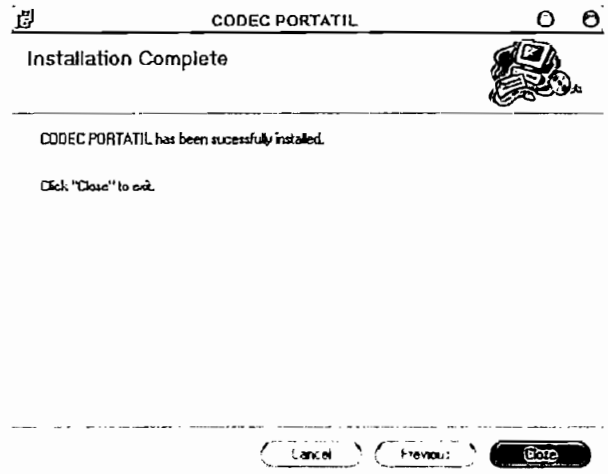


Figura C6 Finalización de la instalación

7. Como ultimo paso hacer un click en *Close* para finalizar la instalación del *software.*

Instalación del Hardware

Los siguientes pasos son necesarios para conectar el CODEC al PC por primera vez.

1. Conecte el *plug* serie A del cable USB al puerto USB de la PC y el *plug* serie B al puerto USB del CODEC.

2. Cuando el CODEC se conecta por primera vez, Windows solicitará al usuario el *software* para USB *High Speed Serial Converter.*

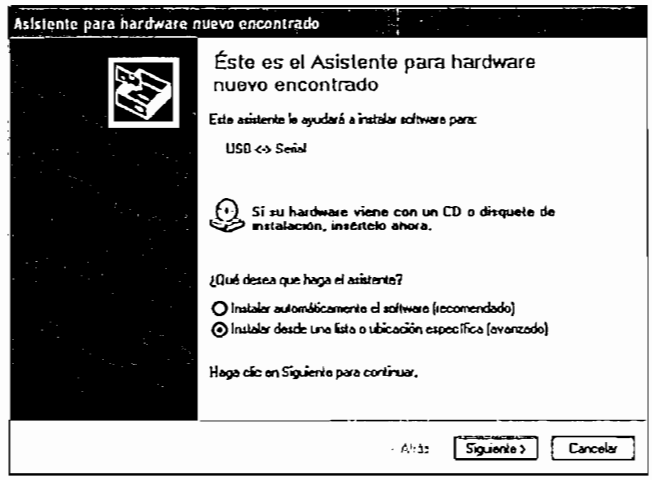


Figura C7 Asistente para instalar USB *High Speed Serial Converter.*

3. Seleccionamos la opción de instalar desde una ubicación específica y hacer click en Siguiente.

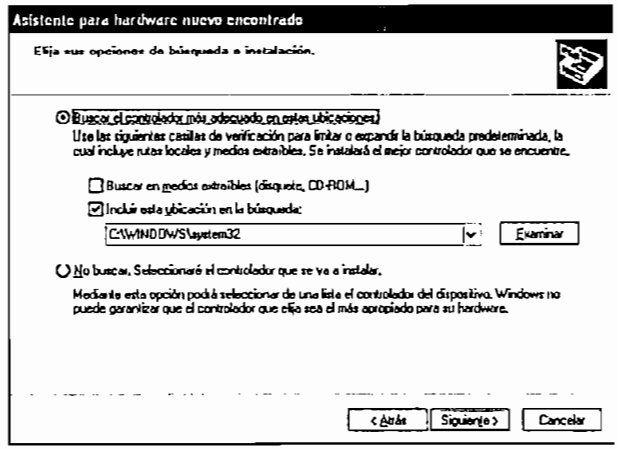


Figura C8 Ubicación de los controladores del *hardware*.

4. Seleccionamos la opción de incluir una ubicación de búsqueda y escribimos la siguiente dirección: C:\WINDOWS\System32 y hacer click en Siguiente para iniciar la instalación de *software* USB *High Speed Serial Converter.*

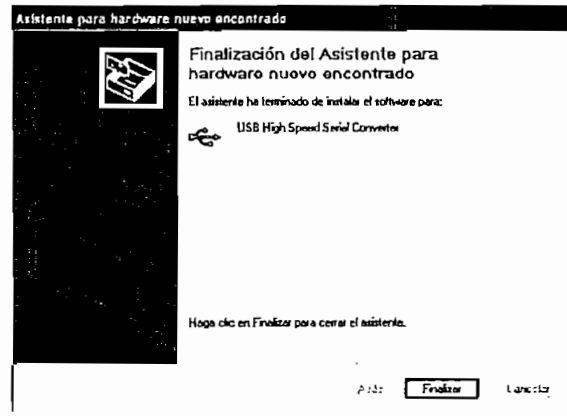


Figura C9 Finalización de la instalacion del USB *High Spedd Serial Converter.*

5. Hacemos click en Finalizar para terminar la instalación del software USB *High Speed Serial Converter.*

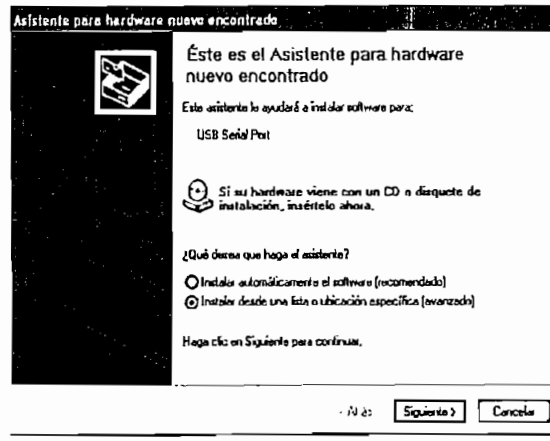


Figura C10 Asistente para instalar USB *Serial Port.*

6. Ahora Windows solicitará al usuario el software para USB *Serial Port.*
7. Repetimos los pasos 3 y 4.

Figura C11 Finalización de la instalación de USB *Serial Port.*

8. Hacemos click en Finalizar para terminar la instalación del software USB *Serial Port* y del *Hardware*.

Iniciando CODEC PORTÁTIL.

Luego de instalar el hardware y previo a la ejecución del programa por primera vez, debemos saber que USB *Serial Port* se ha instalado, para esto, vamos a la Pantalla de Administración de Dispositivos en el panel de control y vemos que COM fue asignado al *USB Serial Port* como se muestra en la Figura C12.

Ahora si podemos iniciar CODEC PORTÁTIL en Inicio > Programas > Codec Portatil.exe. La ventana que aparece la primera vez que inicia el programa solicitará al usuario que seleccione el VCP o USB *Serial Port* que Windows le ha asignado al `hardware` instalado como semuestra en la Figura C13.

Si se ha seleccionado un COM equivocado, debemos de buscar el archivo de texto CodecPortatil en C:\Windows\System32\CodecPortatil.txt. Editar el archivo y cambiar el número encontrado por 0, de esta manera al iniciar el programa podremos seleccionar el COM correcto.

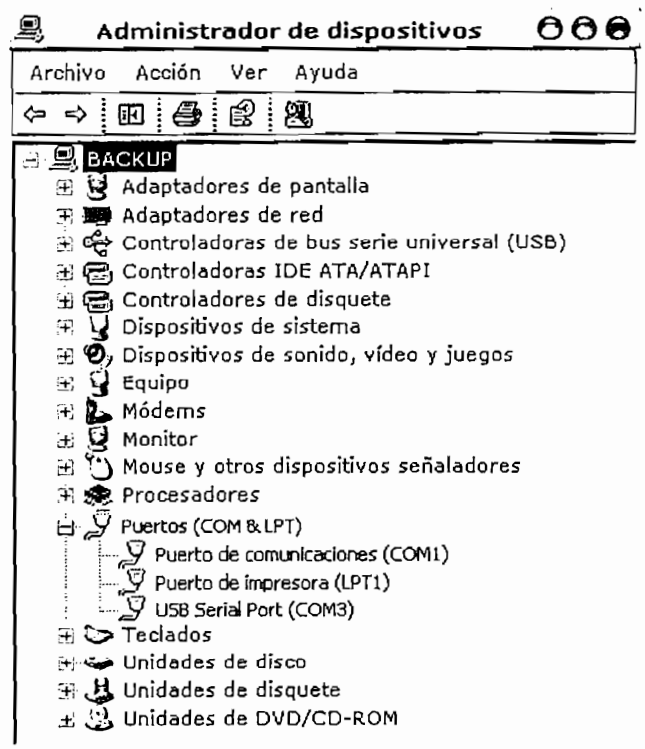


Figura C12 Administrador de Dispositivos de Windows XP.

Figura C13 Seleccionar VCP la primera vez que inicie el Programa.

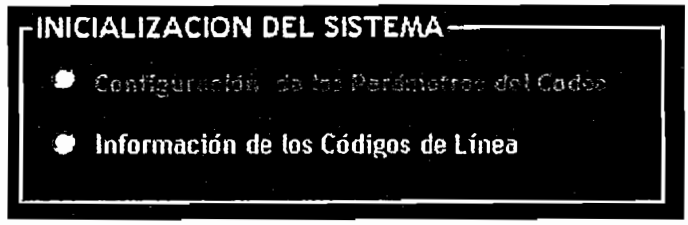
Interfaz de usuario del CODEC PORTÁTIL.

El programa CODEC PORTÁTIL está conformado las siguientes partes principales:

1. Opción de Configuración de los parámetros del CODEC.
2. Opción de Ver Información de los Códigos de Línea.

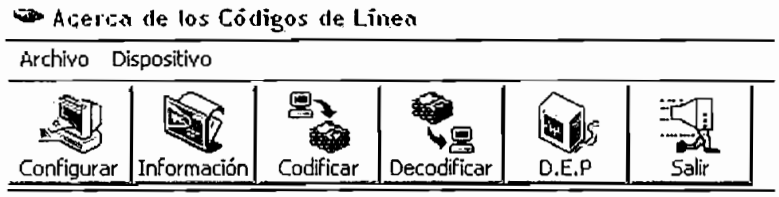
1. Opción de Configuración de los Parámetros del CODEC.

Esta opción se habilitará siempre y cuando el CODEC se haya conectado al PC. Esta opción esta presente en la segunda pantalla al iniciar el programa, Figura C14.a, en la barra de tareas en la opción de dispositivo Figura C14.b, o en el botón de control Configurar, Figura C14.c.



(a)

(b)



(c)

Figura C14 Opciones que habilitan la configuración de los parámetros del CODEC.

Si se ha seleccionado el VCP correcto y se ha conectado el CODEC se cargará la siguiente pantalla con los parámetros para configurar el CODEC como se muestra en la Figura C15.

Los parámetros de configuración del CODEC son: Código de Línea, Velocidad de transmisión y Nivel de señal. Una vez seleccionado estos tres parámetros seleccionamos Enviar Configuración con lo cual enviamos al CODEC la configuración seleccionada. Una vez configurado el CODEC se habilitara la opción de Enviar datos a ser Codificados.

Figura C15 Configuración de Parámetros del CODEC

Al seleccionar Enviar Datos a ser Codificados se presentará una pantalla que permitirá seleccionar el Puerto Serie a utilizarse para enviar datos, los cuales van ha ser codificados y recibir datos que han sido decodificados. Además tenemos las opciones de configurar nuevamente el CODEC y ver información de los códigos de línea como se muestra en la Figura C16.

En la Figura C17 muestra la pantalla que permite visualizar las señales de reloj, datos y datos codificados. Los datos por defecto a ser codificados sin no se ha enviado datos a través del puerto serie, son 1L. Para visualizar las señales de mejor manera tenemos la opción de cambiar la escala de visualización.

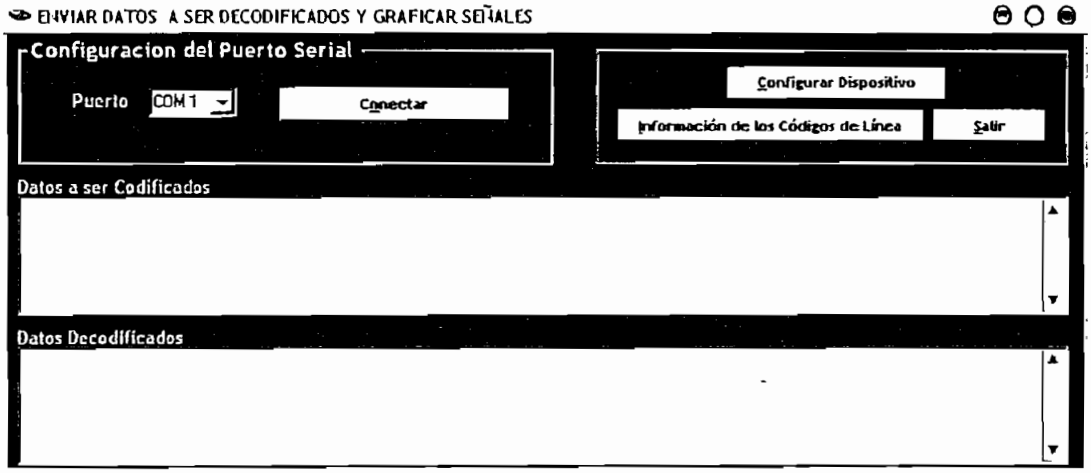


Figura C16 Selección del Puerto Serie para enviar datos a ser codificados y recibir datos decodificados.

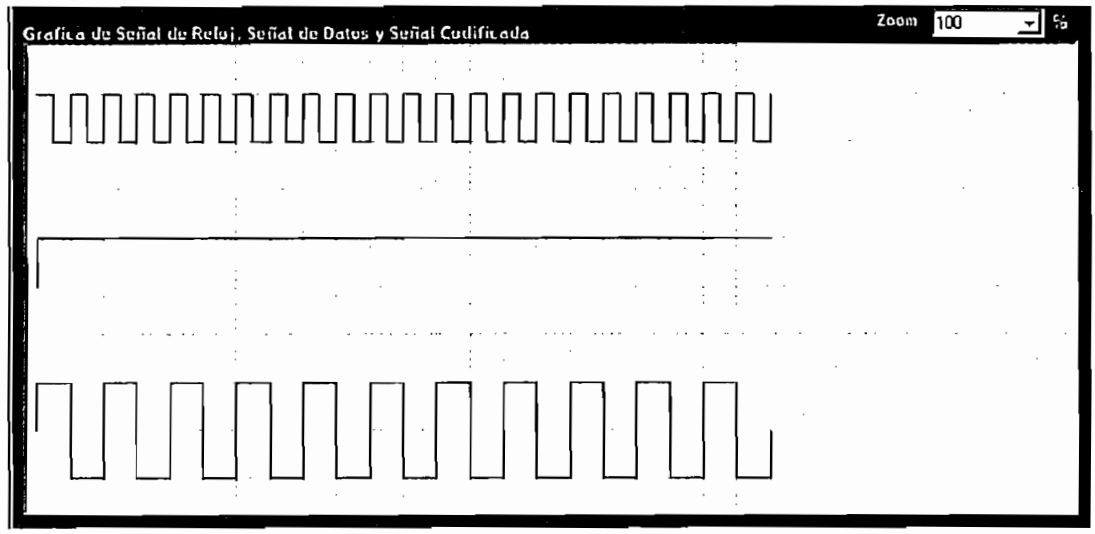


Figura C17 Visualización de la señal de reloj, la señal de datos y la señal codificada.

Las pantallas de Configuración de los Parámetros del CODEC y Envió de Datos a ser Codificados estarán presentes siempre y cuando el CODEC este conectado al PC, si se retira el CODEC, las pantallas se cerrarán y cargaran la pantalla de Información de los Códigos de Línea.

2. Opción de Ver Información de los Códigos de Línea.

Esta opción habilita aplicaciones como la codificación, decodificación e información del código de línea seleccionado, además de una aplicación para visualizar la densidad espectral de potencia como se muestra en la Figura C18.

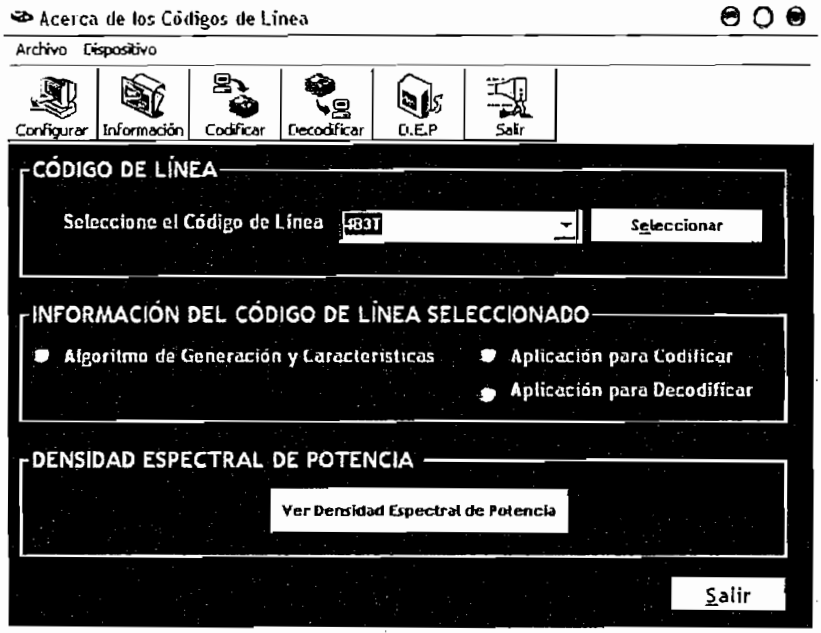


Figura C18 Información de los Códigos de Línea

Para habilitar las opciones de Información del Código de Línea Seleccionado, primero debemos seleccionar el código y luego seleccionamos la aplicación. Para visualizar la densidad espectral de potencia no se necesita seleccionar el código de línea.

Si seleccionamos ver el algoritmo de generación y características del código seleccionado, se presentará la siguiente pantalla.

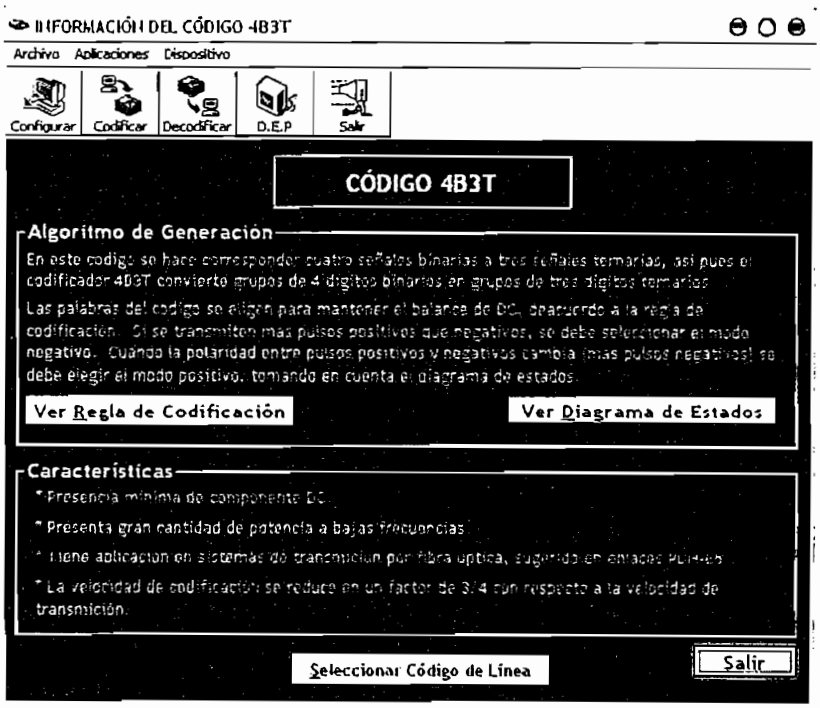


Figura C19 Información del Código Seleccionado

Si el código seleccionado presenta una regla de codificación y/o un diagrama de estados, estos podrán ser visualizados. Además tenemos la opción de ir a las aplicaciones de codificación, decodificación, ver densidad espectral de potencia y de configurar el CODEC si este estuviera conectado.

Si seleccionamos la aplicación para codificar con el código seleccionado, se presentará la siguiente pantalla.

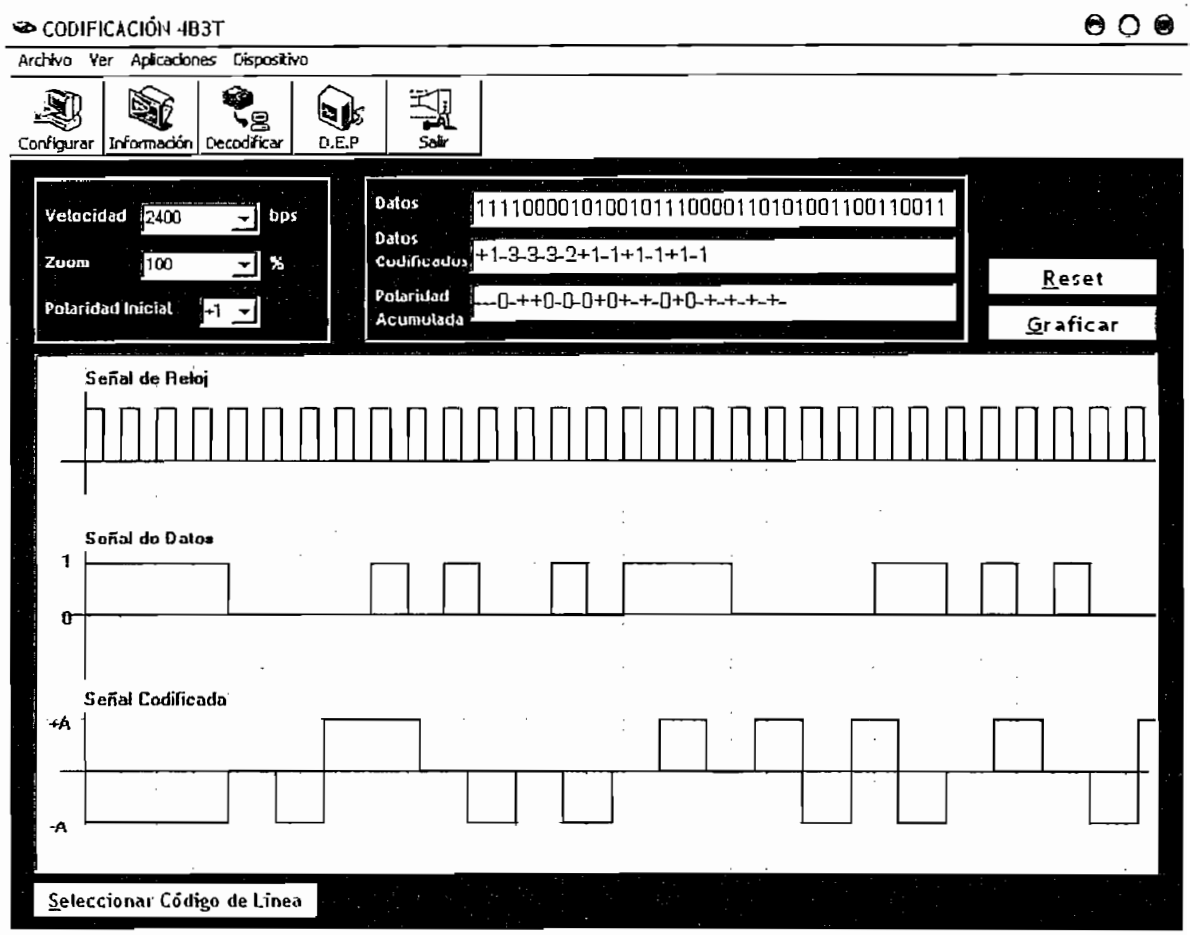


Figura C20 Aplicación para codificar con el código seleccionado.

Esta aplicación permite codificar un máximo de 39 datos para los códigos MS43 y 4B3T y de 40 datos para los restantes códigos. Una vez ingresado los datos a ser codificados seleccionamos graficar, con lo cual podremos visualizar las señales de reloj, de datos, y de datos codificados. Para esta aplicación tenemos la opción de variar la velocidad a la cual se esta codificando, variar la escala de visualización, y dependiendo del código la polaridad inicial o estado inicial. Además tenemos la opción de ir a las aplicaciones de decodificación, ver densidad espectral de potencia, ver información del código seleccionado y de configurar el CODEC si este estuviera conectado.

Si seleccionamos la aplicación para decodificar con el código seleccionado, se presentará la siguiente pantalla.

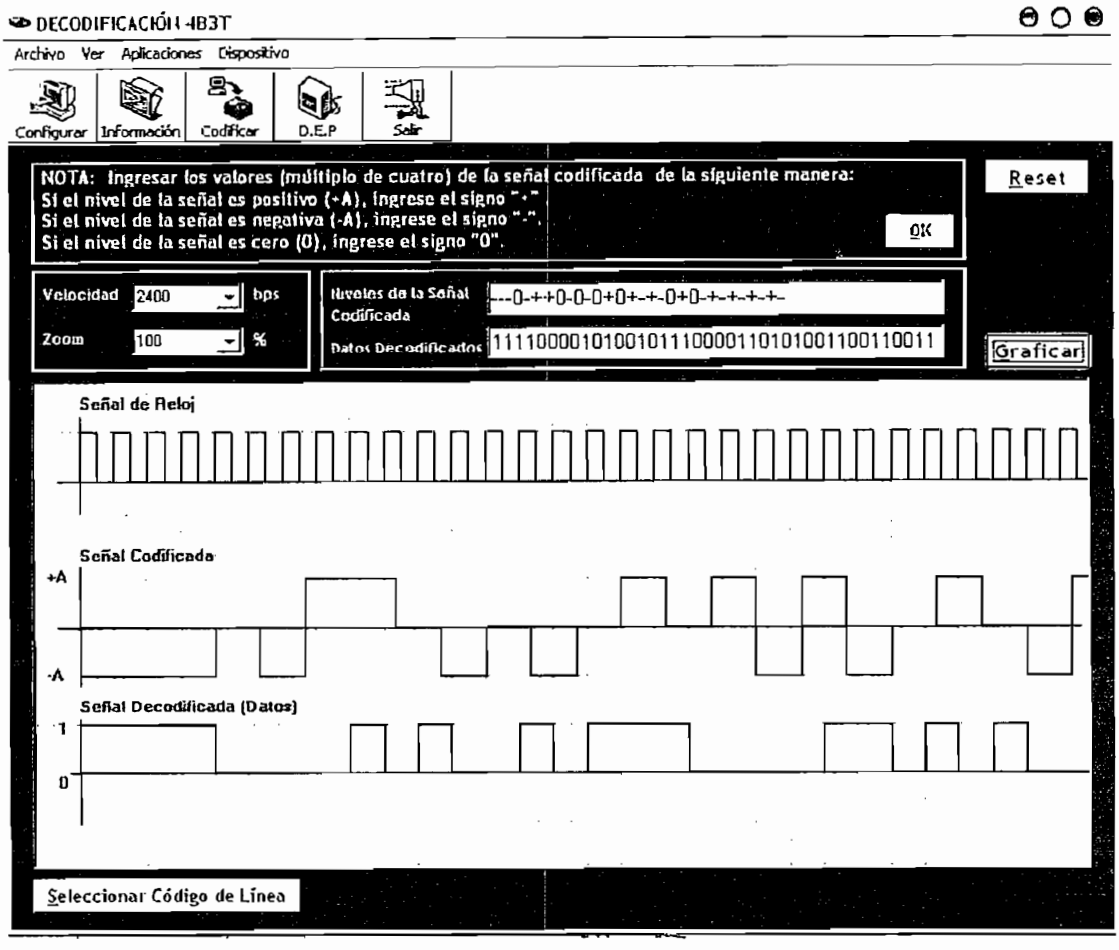


Figura C21 Aplicación para decodificar con el código seleccionado.

Esta aplicación permite decodificar un máximo de 39 datos para los códigos MS43 y 4B3T y de 40 datos para los restantes códigos. Para la utilización de esta aplicación primero debemos leer y entender como ingresar los datos a ser decodificados, luego seleccionamos *OK* para proceder a ingresar los datos a ser decodificados. Una vez ingresado los datos a ser decodificados seleccionamos graficar, con lo cual podremos visualizar las señales de reloj, de datos codificados y de datos decodificados. Para esta aplicación tenemos la opción de variar la velocidad a la cual se esta decodificando y variar la escala de visualización. Además tenemos la opción de ir a las aplicaciones de codificación, ver densidad espectral de potencia, ver información del código seleccionado y de configurar el CODEC si éste estuviera conectado.

Si seleccionamos la aplicación para decodificar con el código seleccionado, se presentará la siguiente pantalla.

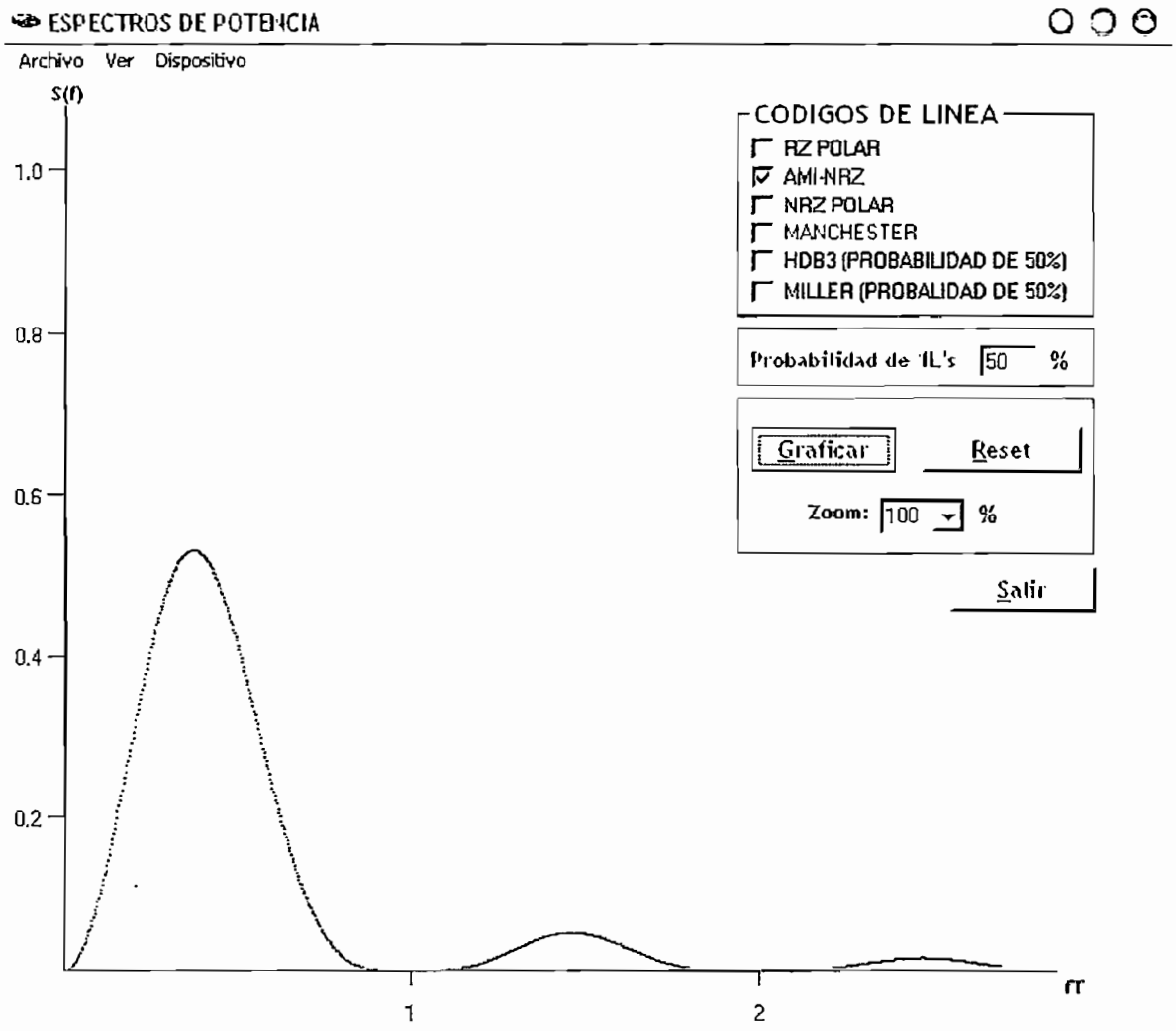


Figura C22 Aplicación para ver la densidad espectral de potencia.

Esta aplicación permite visualizar el espectro de potencia aproximado de los códigos seleccionados. Primero debemos seleccionar el o los códigos, y luego seleccionamos Graficar. Para esta aplicación tenemos la opción de variar la probabilidad de 1L´s y variar la escala de visualización. Además tenemos la opción de ir a las aplicaciones de codificación, aplicación de decodificación, ver información del código seleccionado y de configurar el CODEC si éste estuviera conectado.

ANEXO D

CÓDIGOS FUENTE

CONTENIDO:

CÓDIGO FUENTE DEL HARDWARE CODEC PORTÁTIL TANTO PARA EL MICROCONTROLADOR AT89C52 COMO PARA EL PIC 16F870.

CÓDIGO FUENTE DEL SOFTWARE CODEC PORTÁTIL.

EL CÓDIGO COMPLETO SE ENCUENTRA EN EL CD ADJUNTO.

CÓDIGO FUENTE DEL MICROCONTROLADOR ATMEL 89C52

```
;       ****************************************************************
;       *                      PROGRAMA DEL CPU                        *
;       ****************************************************************

; Se utiliza un cristal de 20 MHz

TXEPIN      EQU   P0.5          ; Indicador de escritura en el MOD. USB
WRPIN       EQU   P0.4          ; Envía la señal de escritura en el MOD. USB
RXFPIN      EQU   P0.6          ; Indicador de lectura en el MOD. USB
RDPIN       EQU   P0.3          ; Envía la señal de lectura en el MOD. USB
DATOS_D     EQU   P3.7          ; Señal de datos decodificados
DATOS       EQU   P3.6          ; Señal de entrada de datos a codificar
SELECT      EQU   P3.5          ; Selección del tipo de nivel lógico TTL/RS232
CLK_C       EQU   P3.4          ; Entrada del Reloj de codificación
CLK_D       EQU   P3.3          ; Entrada del Reloj de decodificación

SIGNO       EQU   P1.0          ; Señal de signo
MAGNITUD    EQU   P1.2          ; Señal de magnitud

MUXB        EQU   P1.3          ; Selector B del Multiplexor analógico
MUXA        EQU   P1.1          ; Selector A del Multiplexor analógico

BCOD        EQU   40H
BDEC        EQU   41H
CODIGO      EQU   42H           ; Almacena el valor de asignación de cada código
VELOCIDAD   EQU   43H           ; Almacena el dato de configuración para el PIC
ALMACEN     EQU   44H           ; Almacena el valor de parada o envío de datos
DATO        EQU   45H           ; Guarda los valores del acumulador usados en la cod.
DECO        EQU   46H           ; Almacena el valor de asignación de cada código
VAR         EQU   48H

TEMP        EQU   49H
DATOC       EQU   4AH
IN_4B       EQU   4BH
OUT_3T      EQU   4CH
MD          EQU   4DH
IN_MS       EQU   4EH
DATOD       EQU   4FH

            LJMP  INICIO        ; Salta al Programa Principal
            ORG   EXTI0         ; Inicio de la Interrupción Externa 0
            LJMP  GRAMS4B
            ORG   TIMER0        ; Inicio de la Interrupción del Timer 0
            LJMP  TEMPO
            ORG   EXTI1         ; Inicio de la Interrupción Externa 1
            LJMP  CODEC
            ORG   TIMER1        ; Inicio de la Interrupción del Timer 1
            RETI
            ORG   SINT          ; Inicio de la Interrupción del Pórtico Serial
            RETI
            ORG   2BH           ; Inicio de la Interrupción Externa 2 o Timer 2
            RETI

;*****************************************************************************
;*                      INICIO DEL PROGRAMA PRINCIPAL                        *
;*****************************************************************************
```

```
            ORG     100H
INICIO:     MOV     SP,#2FH
            MOV     IE,#10000010B       ; Habilita la interrupción del TIMER 0
            MOV     TMOD,#01H
            MOV     TH0,#00H
            MOV     TL0,#00H
            SETB    TR0
            MOV     R5,#01H             ; Indicador de envío de datos al PC
            SETB    SELECT              ; Selección del nivel RS-232
            MOV     CODIGO,#42H         ; Valor codificación AMI (por defecto)

;*************************************************************************
;*                  SELECCION DEL TIPO DE CODIGO                         *
;*************************************************************************

CODIFICAR:  MOV     A,CODIGO
            CJNE    A,#41H,AMI          ; Codificación NRZ Polar?
            MOV     DECO,#41H
            MOV     IE,#10000110B       ; Habilita la interrupción externa 1
            MOV     TCON,#00010101B     ; Habilita la interrupción por transición
            LCALL   INICIAL
            LJMP    NRZ1

AMI:        CJNE    A,#42H,RZ_POLAR     ; Codificación AMI?
            MOV     DECO,#42H
            MOV     IE,#10000110B       ; Habilita la interrupción externa 1
            MOV     TCON,#00010101B     ; Habilita la interrupción por transición
            LCALL   INICIAL
            LJMP    AMI1

RZ_POLAR:   CJNE    A,#43H,MAN_DIF      ; Codificación RZ POLAR?
            MOV     DECO,#43H
            MOV     IE,#10000110B       ; Habilita la interrupción externa 1
            MOV     TCON,#00010101B     ; Habilita la interrupción por transición
            LCALL   INICIAL
            CLR     ACC.1
            LJMP    RZ_P1

MAN_DIF:    CJNE    A,#44H,BI_M         ; Codificación Manchester Diferencial?
            MOV     DECO,#44H
            MOV     IE,#10000010B       ; Habilita la interrupción externa 1
            MOV     TCON,#00010101B     ; Habilita la interrupción por transición
            LCALL   INICIAL
            SETB    ACC.7               ; Transición inicial ascendente
            CLR     ACC.1
            MOV     R4,#00H
            CLR     23H.0
            LJMP    MAN_DIF1

BI_M:       CJNE    A,#45H,MILLER       ; Codificación BIFASE M?
            MOV     DECO,#45H
            MOV     IE,#10000110B       ; Habilita la interrupción externa 1
            MOV     TCON,#00010101B     ; Habilita la interrupción por transición
            LCALL   INICIAL
            LJMP    BI_M1

MILLER:     CJNE    A,#46H,XB3T         ; Codificación MILLER?
            MOV     DECO,#46H
            MOV     IE,#10000110B       ; Habilita la interrupción externa 1
```

```
        MOV     TCON,#00010101B   ; Habilita la interrupción por transición
        LCALL   INICIAL
        LJMP    MILLER1

XB3T:   CJNE    A,#47H,MS43       ; Codificación 4B3T?
        MOV     DECO,#47H
        LCALL   INICIAL
        MOV     R0,#11H           ; Polaridad anterior
        MOV     R1,#00H           ; Contador de datos
        MOV     R2,#00H           ; Inicio de la salida de nivel de datos
        MOV     R3,#00H           ; Contador de datos codificados
        MOV     R4,#00H
        MOV     DATOD,#00H
        MOV     IN_4B,#00H        ; Entrada binaria
        MOV     OUT_3T,#00H       ; Salida ternaria
        MOV     MD,#00H           ; Modo
        MOV     DATOC,#00H        ; Datos codificados
        SETB    20H.0             ; Signo de polaridad
        CLR     C
        MOV     IE,#10000111B     ; Habilita la interrupción externa 0 y 1
        MOV     IP,#00000100B
        MOV     TCON,#00010101B   ; Habilita la interrupción por transición
        SETB    TR0
        LJMP    S_COD_4B

MS43:   CJNE    A,#48H,B3ZS       ; Codificación MS43?
        MOV     DECO,#48H
        LCALL   INICIAL
        MOV     R0,#01H           ; Polaridad anterior
        MOV     R1,#00H           ; Contador de datos
        MOV     R2,#00H           ; Inicio de la salida de nivel de datos
        MOV     R3,#01H           ; Evita dualidad en secuencias 1111 Y
        MOV     R4,#00H           ; 0111
        MOV     DATOD,#00H
        MOV     DATOC,#00H
        MOV     IN_MS,#00H        ; Entrada binaria
        MOV     OUT_3T,#00H       ; Salida ternaria
        MOV     MD,#00H           ; Modo
        SETB    20H.0             ; Signo de polaridad
        CLR     C
        MOV     IE,#10000111B     ; Habilita la interrupción externa 0 y 1
        MOV     IP,#00000100B
        MOV     TCON,#00010101B   ; Habilita la interrupción por transición
        SETB    TR0
        LJMP    S_COD_MS

B3ZS:   CJNE    A,#49H,HDB3       ; Codificación B3ZS?
        MOV     DECO,#49H
        MOV     IE,#10000110B     ; Habilita la interrupción externa 1
        MOV     TCON,#00010101B   ; Habilita la interrupción por transición
        LCALL   INICIAL
        CLR     ACC.0
        MOV     R0,#00H           ; Polaridad anterior
        CLR     20H.0             ; Paridad de 1L
        MOV     R1,#00H           ; Contador de 0L
        MOV     R2,#00H           ; Contador de datos
        MOV     BCOD,#00H
        MOV     R4,#00H           ; Contador de datos
        CLR     21H.0             ; Nivel anterior
```

```
                MOV     BDEC,#00H
                LJMP    B3ZS1
HDB3:           CJNE    A,#4AH,MLT3     ; Codificación HDB3?
                MOV     DECO,#4AH
                MOV     IE,#10000110B   ; Habilita la interrupción externa 1
                MOV     TCON,#00010101B ; Habilita la interrupción por transición
                LCALL   INICIAL
                CLR     ACC.0
                MOV     R0,#00H         ; Polaridad anterior
                CLR     20H.0           ; Paridad de 1L
                MOV     R1,#00H         ; Contador de 0L
                MOV     R2,#00H         ; Contador de datos
                MOV     BCOD,#00H
                MOV     R4,#00H         ; Contador de datos
                CLR     21H.0           ; Nivel anterior
                MOV     BDEC,#00H
                LJMP    HDB31

MLT3:           CJNE    A,#4BH,CMI      ; Codificación MLT3?
                MOV     DECO,#4BH
                MOV     IE,#10000110B   ; Habilita la interrupción externa 1
                MOV     TCON,#00010101B ; Habilita la interrupción por transición
                LCALL   INICIAL
                SETB    ACC.2           ; Indica el nivel del último pulso
                SETB    ACC.6           ; Nivel inicial positivo
                LJMP    MLT31

CMI:            CJNE    A,#4CH,FIN1     ; Codificación CMI?
                MOV     DECO,#4CH
                MOV     IE,#10000110B   ; Habilita la interrupción externa 1
                MOV     TCON,#00010101B ; Habilita la interrupción por transición
                LCALL   INICIAL
                LJMP    CMI1

FIN1:           LJMP    FIN

;*******************************************************************************
;*                          CODIFICACION 4B3T                                  *
;*******************************************************************************

;*******************************************************************************
;*                          CODIFICACION MS43                                  *
;*******************************************************************************

;*******************************************************************************
;*                          CODIFICACION BIFASE M                              *
;*******************************************************************************

;*******************************************************************************
;*                          SUBRUTINAS 4B3T                                    *
;*******************************************************************************

;*******************************************************************************
;*                          SUBRUTINAS MS43                                    *
;*******************************************************************************

;*******************************************************************************
;*                          CODIFICACION HDB3                                  *
;*******************************************************************************
```

```
;*************************************************************************
;*                          CODIFICACION B3ZS                            *
;*************************************************************************

;*************************************************************************
;*                          CODIFICACION MILLER                          *
;*************************************************************************

;*************************************************************************
;*                          CODIFICACION MLT3                            *
;*************************************************************************

;*************************************************************************
;*          CODIFICACION MANCHESTER DIFERENCIAL                          *
;*************************************************************************

;*************************************************************************
;*                          CODIFICACION AMI                             *
;*************************************************************************

AMI1:       JNB     CLK_C,$          ; Espera transición descendente
            JB      CLK_C,$

            JB      DATOS,CUNO_A
            CLR     MUXA             ; Permiten colocar al multiplexor
            SETB    MUXB             ; en el nivel de 0V.
            SJMP    FINAC
CUNO_A:     JNB     ACC.0,CUNO_P_A
            CPL     ACC.0
            SETB    MUXA             ; Permiten colocar al multiplexor
            CLR     MUXB             ; en el nivel de -5V.
            SJMP    FINAC
CUNO_P_A:   CPL     ACC.0
            CLR     MUXA             ; Permiten colocar al multiplexor
            CLR     MUXB             ; en el nivel de +5V.
FINAC:      MOV     R6,#01H
VERIAMI:    JB      RXFPIN,AMI1      ; Censa dato del USB o salta a codificar
            LCALL   LECTURA
            MOV     A,ALMACEN
            CJNE    A,#00H,AMIC
            LJMP    CODIFICAR
AMIC:       MOV     A,DATO
            SJMP    AMI1

;*************************************************************************
;*                       CODIFICACION RZ POLAR                           *
;*************************************************************************

;*************************************************************************
;*                       CODIFICACION NRZ POLAR                          *
;*************************************************************************

;*************************************************************************
;*                          CODIFICACION CMI                             *
;*************************************************************************

;*************************************************************************
;*       SUBRUTINA DE INICIALIZACION DE CODIGO                           *
;*************************************************************************
```

```
;*************************************************************************
;*   SUBRUTINA PARA CONFIGURAR LOS PARAMETROS DE VELOCIDAD Y CODIGO
*
;*************************************************************************

CONFIGURE:  CJNE    A,#4DH,SONDEO1      ; Verifica si se trata de la
            MOV     VELOCIDAD,#41H      ; velocidad de 300
            MOV     ALMACEN,#00H        ; Indica Selección de otro parámetro
            LCALL   TRANSMITIR
            LJMP    EXIT
SONDEO1:    CJNE    A,#4EH,SONDEO2      ; Verifica si se trata de la
            MOV     VELOCIDAD,#42H      ; velocidad de 600
            MOV     ALMACEN,#00H
            LCALL   TRANSMITIR
            LJMP    EXIT
SONDEO2:    CJNE    A,#4FH,SONDEO3      ; Verifica si se trata de la
            MOV     VELOCIDAD,#43H      ; velocidad de 1200
            MOV     ALMACEN,#00H
            LCALL   TRANSMITIR
            LJMP    EXIT
SONDEO3:    CJNE    A,#50H,SONDEO4      ; Verifica si se trata de la
            MOV     VELOCIDAD,#44H      ; velocidad de 2400
            MOV     ALMACEN,#00H
            LCALL   TRANSMITIR
            LJMP    EXIT
SONDEO4:    CJNE    A,#51H,SONDEO5      ; Verifica si se trata de la
            MOV     VELOCIDAD,#45H      ; velocidad de 4800
            MOV     ALMACEN,#00H
            LCALL   TRANSMITIR
            LJMP    EXIT
SONDEO5:    CJNE    A,#52H,SONDEO6      ; Verifica si se trata de la
            MOV     VELOCIDAD,#46H      ; velocidad de 9600
            MOV     ALMACEN,#00H
            LCALL   TRANSMITIR
            LJMP    EXIT
SONDEO6:    CJNE    A,#53H,SONDEO7      ; Verifica si se trata de la
            MOV     VELOCIDAD,#47H      ; velocidad de 19200
            MOV     ALMACEN,#00H
            LCALL   TRANSMITIR
            LJMP    EXIT
SONDEO7:    CJNE    A,#41H,SONDEO8      ; Verifica si se trata de la
            MOV     VELOCIDAD,#49H      ; Codificación NRZ Polar
            MOV     CODIGO,A
            MOV     ALMACEN,#00H
            LCALL   TRANSMITIR
            LJMP    EXIT
SONDEO8:    CJNE    A,#42H,SONDEO9      ; Verifica si se trata de la
            MOV     VELOCIDAD,#49H      ; Codificación AMI
            MOV     CODIGO,A
            MOV     ALMACEN,#00H
            LCALL   TRANSMITIR
            LJMP    EXIT
SONDEO9:    CJNE    A,#43H,SONDEO10     ; Verifica si se trata de la
            MOV     VELOCIDAD,#4AH      ; Codificación RZ Polar
            MOV     CODIGO,A
            LCALL   TRANSMITIR
            LJMP    EXIT
SONDEO10:   CJNE    A,#44H,SONDEO11     ; Verifica si se trata de la
            MOV     VELOCIDAD,#4AH      ; Codificación Manchester Diferencial
```

```
                MOV     CODIGO,A
                MOV     ALMACEN,#00H
                LCALL   TRANSMITIR
                LJMP    EXIT
SONDEO11:       CJNE    A,#45H,SONDEO12     ; Verifica si se trata de la
                MOV     VELOCIDAD,#4AH      ; Codificación Bifase M
                MOV     CODIGO,A
                MOV     ALMACEN,#00H
                LCALL   TRANSMITIR
                LJMP    EXIT
SONDEO12:       CJNE    A,#46H,SONDEO13     ; Verifica si se trata de la
                MOV     VELOCIDAD,#4AH      ; Codificación Miller
                MOV     CODIGO,A
                MOV     ALMACEN,#00H
                LCALL   TRANSMITIR
                LJMP    EXIT
SONDEO13:       CJNE    A,#47H,SONDEO14     ; Verifica si se trata de la
                MOV     VELOCIDAD,#48H      ; Codificación 4B3T
                MOV     CODIGO,A
                MOV     ALMACEN,#00H
                LCALL   TRANSMITIR
                LJMP    EXIT
SONDEO14:       CJNE    A,#48H,SONDEO15     ; Verifica si se trata de la
                MOV     VELOCIDAD,#48H      ; Codificación MS43
                MOV     CODIGO,A
                MOV     ALMACEN,#00H
                LCALL   TRANSMITIR
                LJMP    EXIT
SONDEO15:       CJNE    A,#49H,SONDEO16     ; Verifica si se trata de la
                MOV     VELOCIDAD,#49H      ; Codificación B3ZS
                MOV     CODIGO,A
                MOV     ALMACEN,#00H
                LCALL   TRANSMITIR
                SJMP    EXIT
SONDEO16:       CJNE    A,#4AH,SONDEO17     ; Verifica si se trata de la
                MOV     VELOCIDAD,#49H      ; Codificación HDB3
                MOV     CODIGO,A
                MOV     ALMACEN,#00H
                LCALL   TRANSMITIR
                SJMP    EXIT
SONDEO17:       CJNE    A,#4BH,SONDEO18     ; Verifica si se trata de la
                MOV     VELOCIDAD,#49H      ; Codificación MLT3
                MOV     CODIGO,A
                MOV     ALMACEN,#00H
                LCALL   TRANSMITIR
                SJMP    EXIT
SONDEO18:       CJNE    A,#4CH,SONDEO19     ; Verifica si se trata de la
                MOV     VELOCIDAD,#4AH      ; Codificación CMI
                MOV     CODIGO,A
                MOV     ALMACEN,#00H
                LCALL   TRANSMITIR
                SJMP    EXIT
SONDEO19:       CJNE    A,#5AH,SONDEO20     ; Envío de detección del CODEC?
                SETB    WRPIN               ; Bit de escritura en alto
                JB      TXEPIN,$            ; Chequea el permiso de escritura
                MOV     P2,#43H             ; Envío de la letra C
                CLR     WRPIN               ; Permite la escritura en el buffer
                SETB    WRPIN               ; Bit de lectura en alto
                SJMP    EXIT
```

```
SONDEO20:   CJNE    A,#58H,SONDEO21   ; Verifica si se trata de (X) parar
            MOV     R5,#00H           ; el envío de datos al PC
            MOV     ALMACEN,#01H
            SJMP    EXIT
SONDEO21:   CJNE    A,#59H,SONDEO22   ; Verifica si se trata de (Y) continuar
            MOV     R5,#01H           ; el envío de datos al PC
            MOV     ALMACEN,#01H
            SJMP    EXIT
SONDEO22:   CJNE    A,#57H,SONDEO23   ; Verifica si se trata de (W) RS-232
            SETB    SELECT
            SJMP    EXIT
SONDEO23:   CJNE    A,#56H,EXIT       ; Verifica si se trata de (V) TLL
            CLR     SELECT
EXIT:       RET

;*************************************************************************
;*  SUBRUTINA GENERADORA DE TX SERIAL PARA COMUNICACION CON EL PIC   *
;*************************************************************************

TRANSMITIR: MOV     SCON,#70H         ; Configura el puerto Serial
            MOV     PCON,#80H
            MOV     TMOD,#21H         ; Configura el Timer 1 en modo 2
            MOV     TH1,#0F5H         ; Carga el valor para transmitir a
            SETB    TR1               ; 9600 bits/seg
            MOV     A,VELOCIDAD       ; Mueve el valor cargado en la variable
            MOV     SBUF,A            ; VELOCIDAD
            JNB     TI,$              ; Espera el desbordamiento del
            CLR     TR1               ; el Timer 1
            CLR     TI                ; Borra bandera de desbordamiento de
            RET                       ; Transmisión
;*************************************************************************
;*    SUBRUTINA DE LECTURA DEL DATO EN EL BUFFER EN EL MODULO USB
*
;*************************************************************************
LECTURA:    MOV     DATO,A
            MOV     P2,#0FFH          ; Coloca al puerto P2 A 1L
            CLR     RDPIN             ; Permite leer el dato
            NOP
            MOV     A,P2              ; Guarda el dato en el acumulador
            SETB    RDPIN             ; Bit de Lectura en alto
            LCALL   CONFIGURE
            JB      RXFPIN,SIGUE      ;Censa si existe dato para leer
            MOV     P2,#0FFH          ; Coloca al puerto P2 A 1L
            CLR     RDPIN             ; Permite leer el dato
            NOP
            MOV     A,P2              ; Guarda el dato en el acumulador
            SETB    RDPIN       ; Bit de Lectura en alto
            LCALL   CONFIGURE
            JB      RXFPIN,SIGUE      ;Censa si existe dato para leer
            MOV     P2,#0FFH          ; Coloca al puerto P2 A 1L
            CLR     RDPIN             ; Permite leer el dato
            NOP
            MOV     A,P2              ; Guarda el dato en el acumulador
            SETB    RDPIN             ; Bit de Lectura en alto
            LCALL   CONFIGURE
SIGUE:      RET
;*************************************************************************
;*                    SUBRUTINA DE DECODIFICACION                    *
;*************************************************************************
```

```
CODEC:    PUSH    PSW
          MOV     DATO,A
          MOV     A,DECO
          CJNE    A,#48H,DEC4B      ; Decodificación MS43?
          MOV     A,DATO
          LJMP    GRADECMS
DEC4B:    CJNE    A,#47H,DECHD      ; Decodificación 4B3T?
          MOV     A,DATO
          LJMP    GRADEC4B
DECHD:    CJNE    A,#4AH,DECB3      ; Decodificación HDB3?
          MOV     A,DATO
          LJMP    HBDEC
DECB3:    CJNE    A,#49H,DECMAN     ; Decodificación B3ZS?
          MOV     A,DATO
          LJMP    BZDEC
DECMAN:   CJNE    A,#44H,DECBI      ; Decodificación MAN_D?
          MOV     A,DATO
          LJMP    MDDEC
DECBI:    CJNE    A,#45H,DECMIL     ; Decodificación BI_M?
          MOV     A,DATO
          LJMP    BMDEC
DECMIL:   CJNE    A,#46H,DECCM      ; Decodificación Miller?
          MOV     A,DATO
          LJMP    MILDEC
DECCM:    CJNE    A,#4CH,DECR       ; Decodificación CMI?
          MOV     A,DATO
          LJMP    CMIDEC
DECR:     CJNE    A,#43H,DECNR      ; Decodificación RZ_Polar?
          MOV     A,DATO
          LJMP    RZDEC
DECNR:    CJNE    A,#41H,DECAM      ; Decodificación NRZ_Polar?
          MOV     A,DATO
          LJMP    NRZDEC
DECAM:    CJNE    A,#42H,DECMLT     ; Decodificación AMI?
          MOV     A,DATO
          LJMP    AMIDEC
DECMLT:   CJNE    A,#4BH,FINISH     ; Decodificación MLT3?
          MOV     A,DATO
          LJMP    MLTDEC
FINISH:   RETI
;*************************************************************************
;*                      DECODIFICACION HDB3                              *
;*************************************************************************

;*************************************************************************
;*                      DECODIFICACION B3ZS                              *
;*************************************************************************

;*************************************************************************
;*              DECODIFICACION MANCHESTER DIFERENCIAL                    *
;*************************************************************************
;*************************************************************************
;*                      DECODIFICACION MILLER                            *
;*************************************************************************

;*************************************************************************
;*                      DECODIFICACION BIFASE M                          *
;*************************************************************************
```

```
;*********************************************************************
;*                        DECODIFICACION CMI                         *
;*********************************************************************

;*********************************************************************
;*                      DECODIFICACION MLT3                          *
;*********************************************************************

;*********************************************************************
;*                      DECODIFICACION RZ POLAR                      *
;*********************************************************************

;*********************************************************************
;*                      DECODIFICACION AMI                           *
;*********************************************************************

AMIDEC:     CJNE    R6,#01H,FINAD
            JB      MAGNITUD,N_AB_A
            CLR     DATOS_D          ; Indica que el dato decodificado es 0L
            SJMP    FINAD            ; Regresa a censar la codificación N_AB_A:
            SETB    DATOS_D          ; Indica que el dato decodificado es 1L FINAD:
            POP     PSW
            RETI                     ; Regresa a censar la codificación

;*************************************************************
;*                      DECODIFICACION NRZ POLAR             *
;*************************************************************

;*********************************************************************
;*                      SUBRUTINA DE ENVIO DE 1L                     *
;*********************************************************************

TEMPO:      CJNE    R5,#01H,NOD0
            SETB    WRPIN            ; Bit de escritura en alto
            JB      TXEPIN,NOD0      ; Chequea el permiso de escritura
            MOV     P2,#31H          ; Envío de 1L
            CLR     WRPIN            ; Escritura en el buffer
NOD0:       CLR     TF0
            RETI                     ; Regresa a censar la codificación

            ORG     1000H

;*********************************************************************
;*          SUBRUTINA PARA GRAFICAR CODIFICACION MS43 Y 4B3T         *
;*********************************************************************

;*********************************************************************
;*                      DECODIFICACION MS43                          *
;*********************************************************************

;*********************************************************************
;*                      DECODIFICACION 4B3T                          *
;*********************************************************************
FIN:        END
```

CÓDIGO FUENTE DEL PIC 16F870

```
;*************************************************************************
;*   PROGRAMA DE LAS SEÑALES DE RELOJ DE CODIFICACIÓN Y                  *
;*                        DECODIFICACIÓN                                 *
;*************************************************************************

;RELOJ VERSION 4

                LIST        P=16F870
                RADIX HEX
                INCLUDE     <P16F870.INC>

TIMERHI         EQU         0x21
TIMERLOW        EQU         0x22
CODEC           EQU         0X24        ; Variable que indica el estado anterior del dato
CODIGO          EQU         0X25        ; Variable para indicar el tipo de código
VELOCIDAD       EQU         0X26
#DEFINE         CLKCOD      PORTB,5
#DEFINE         CLKDEC      PORTB,6
#DEFINE         CLKDECI     PORTB,7
#DEFINE         DAT_IN      PORTA,0

;PORTA 0 entrada de datos y PORTB 5 6 salidas de reloj

                ORG         0x00
                goto        inicio
                ORG         0x04
                goto        inter

seleccion       bcf         STATUS,RP0  ; Cambia al banco 0
                bcf         STATUS,RP1
                bcf         STATUS,Z
                movf        RCREG,W     ; Mueve el cont. del Buffer del USART al acumulador.
                movwf       VELOCIDAD
                movlw       0x41        ; Implica velocidad de 300
                subwf       VELOCIDAD,W
                btfss       STATUS,Z
                goto        VEL600
                movlw       0xEF        ; Asignación de
                movwf       TIMERHI     ; valores al
                movlw       0xC0        ; TMR1H y al
                movwf       TIMERLOW    ; TMR1L
                goto        salir1

VEL600          bcf         STATUS,Z
                movlw       0x42        ; Implica velocidad de 600
                subwf       VELOCIDAD,W
                btfss       STATUS,Z
                goto        VEL1200
                movlw       0xF7        ; asignación de
                movwf       TIMERHI     ; valores al
                movlw       0xE3        ; TMR1H y al
                movwf       TIMERLOW    ; TMR1L
                goto        salir1

VEL1200         bcf         STATUS,Z
                movlw       0x43        ; implica velocidad de 1200
```

```
            subwf   VELOCIDAD,W
            btfss   STATUS,Z
            goto    VEL2400
            movlw   0xFC        ; Asignación de
            movwf   TIMERHI     ; valores al
            movlw   0x01        ; TMR1H y al
            movwf   TIMERLOW    ; TMR1L
            goto    salir1

VEL2400     bcf     STATUS,Z
            movlw   0x44        ;implica velocidad de 2400
            subwf   VELOCIDAD,W
            btfss   STATUS,Z
            goto    VEL4800
            movlw   0xFD        ; Asignación de
            movwf   TIMERHI     ; valores al
            movlw   0xF7        ; TMR1H y al(FDFE)
            movwf   TIMERLOW    ; TMR1L
            goto    salir1

VEL4800     bcf     STATUS,Z
            movlw   0x45        ; Implica velocidad de 4800
            subwf   VELOCIDAD,W
            btfss   STATUS,Z
            goto    VEL9600
            movlw   0xFF        ; Asignación de
            movwf   TIMERHI     ; valores al
            movlw   0x00        ; TMR1H y al
            movwf   TIMERLOW    ; TMR1L
            goto    salir1

VEL9600     bcf     STATUS,Z
            movlw   0x46        ; Implica velocidad de 9600
            subwf   VELOCIDAD,W
            btfss   STATUS,Z
            goto    VEL19200
            movlw   0xFF        ; Asignación de
            movwf   TIMERHI     ; valores al
            movlw   0x7F        ; TMR1H y al(FF84)
            movwf   TIMERLOW    ; TMR1L
            goto    salir1

VEL19200    bcf     STATUS,Z
            movlw   0x47        ; Implica velocidad de 19200
            subwf   VELOCIDAD,W
            btfss   STATUS,Z
            goto    CODIGO1
            movlw   0xFF        ; Asignación de
            movwf   TIMERHI     ; valores al
            movlw   0xC6        ; TMR1H y al
            movwf   TIMERLOW    ; TMR1L

CODIGO1     bcf     STATUS,Z
            movlw   0x49        ;implica código AMI
            subwf   VELOCIDAD,W
            btfss   STATUS,Z
            goto    CODIGO2
            movlw   0x00        ; Asignación del valor
            movwf   CODIGO      ; de los códigos AMI, NRZ Polar y otros
```

```
                bcf     T1CON,T1CKPS1
                bsf     T1CON,T1CKPS0

CODIGO2         bcf     STATUS,Z
                movlw   0x4A            ; Implica código RZ Polar
                subwf   VELOCIDAD,W
                btfss   STATUS,Z
                goto    CODIGO3
                movlw   0x01            ; Asignación del valor
                movwf   CODIGO          ; de los códigos diferenciales
                bcf     T1CON,T1CKPS1
                bcf     T1CON,T1CKPS0

CODIGO3         bcf     STATUS,Z
                movlw   0x48            ; Implica código MS43 y 4B3T
                subwf   VELOCIDAD,W
                btfss   STATUS,Z
                goto    salir1
                movlw   0x02            ; Asignación del valor
                movwf   CODIGO          ; de los códigos MS43 y 4B3T
                movlw   0xF5
                movwf   TIMERHI         ; valores al
                movlw   0x2B            ; TMR1H y al
                movwf   TIMERLOW        ; TMR1L
                bcf     T1CON,T1CKPS1
                bcf     T1CON,T1CKPS0
salir1          return

lower           bcf     PIR1,TMR1IF                     ;1
                bcf     CLKCOD                          ;1
                movf    TIMERLOW,W                      ;1
                movwf   TMR1L                           ;1
                movf    TIMERHI,W                       ;1
                movwf   TMR1H                           ;1
                return                                  ;2

higher          bcf     PIR1,TMR1IF                     ;1
                bsf     CLKCOD                          ;1
                movf    TIMERLOW,W                      ;1
                movwf   TMR1L                           ;1
                movf    TIMERHI,W                       ;1
                movwf   TMR1H                           ;1
                return                                  ;2

inter           btfss   PIR1,RCIF
                goto    fin
                bcf     PIR1,RCIF
                call    seleccion
fin             retfie


inicio          clrwdt
                clrf    PORTA ; Coloca en cero el puerto A
                clrf    PORTB ; Coloca en cero el puerto B
                clrf    PORTC ; Coloca en cero el puerto C
                bsf     STATUS,RP0      ; Cambia al
                bcf     STATUS,RP1      ; BANCO 1
                movlw   0X06            ; Configura el puerto A como I/O digitales
                movwf   ADCON1
```

```
                movlw   0x01            ; Configura el Puerto A 1 como salida
                movwf   TRISA           ;y 0 como entrada
                movlw   0xCF            ;Presivisor wdt 1:128
                movwf   OPTION_REG ;desactiva las R's de pull up del PORTB
                movlw   0x1F            ;configura el Puerto B 5 6
                movwf   TRISB           ;como salida y 0 1 2 3 4 5 como entrada
                movlw   0xBF            ;configura el RC7/RX como entrada y
                movwf   TRISC           ;RC6/TX como salida
                movlw   0x20            ;activa la Tx asincrona y
                movwf   TXSTA ;velocidad baja
                movlw   0x1F            ;fija la velocidad en el registro
                movwf   SPBRG ;SPBRG a 9600 (0sc 20 Mhz 1F)
                bcf     STATUS,RP0      ;cambia al BANCO 0
                movlw   0x90            ;habilita el puerto serial con el
                movwf   RCSTA ;bit SPEN
                movlw   0xC0
                movwf   INTCON          ;activa las interrupciones
                bsf     STATUS,RP0      ;cambia al BANCO 1
                movlw   0x20
                movwf   PIE1
                bcf     STATUS,RP0      ;cambia al BANCO 0
                bcf     PIR1,RCIF
                movlw   0xEF            ;asignación de valores al
                movwf   TIMERHI         ;TMR1H y al TMR1L
                movlw   0xC0            ;para la velocidad de
                movwf   TIMERLOW        ;300 bits/s
                movlw   0x00
                movwf   CODEC;Indica el estado anterior
                movlw   0x00
                movwf   CODIGO          ;Indica Codificación AMI
                movlw   0x00
                movwf   T1CON
                bcf     T1CON,T1CKPS1
                bsf     T1CON,T1CKPS0
                bsf     T1CON,TMR1ON

;*****************************************************************************
;*              RELOJ  DE CODIFICACION Y DECODIFICACION                      *
;*****************************************************************************
;               Ciclos de Máquina
;*****************************************************************************
;*              NIVEL DEL RELOJ DE CODIFICACION EN ALTO                      *
;*****************************************************************************

saltar4         btfss   CODIGO,0                                        ;2
                goto    saltar
                bcf     CLKDEC
saltar          call    higher                                          ;2
                btfsc   CODIGO,0                                        ;2
                goto    inversor
                nop
                bcf     CLKDEC
                goto    hi
inversor        bsf     CLKDEC
                nop
                nop
hi              btfsc   PIR1,TMR1IF                                     ;2
                goto    saltar1
                btfsc   DAT_IN          ;chequea si la entrada a codificar      ;2
```

```
                goto    levehi      ;es cero si lo es brinca un espacio     ;2
                bcf     CODEC,1
                btfss   CODEC,0     ;chequea si la entrada a codificar      ;2
                goto    hi
                rrf     CODEC
                goto    saltar
levehi          bsf     CODEC,1
                btfsc   CODEC,0     ;chequea si la entrada a codificar      ;2
                goto    hi
                rrf     CODEC
                goto    saltar

saltar1         btfss   CODIGO,0                                            ;2
                goto    saltar2
                nop
                call    higher                                              ;2
                nop
                nop
                nop
                bcf     CLKDEC
                nop
                nop
hi1             btfsc   PIR1,TMR1IF                                         ;2
                goto    saltar3
                btfsc   DAT_IN      ;chequea si la entrada a codificar      ;2
                goto    levehi1     ;es cero si lo es brinca un espacio     ;2
                bcf     CODEC,1
                btfss   CODEC,0     ;chequea si la entrada a codificar      ;2
                goto    hi1
                rrf     CODEC
                goto    saltar5
levehi1         bsf     CODEC,1
                btfsc   CODEC,0     ;chequea si la entrada a codificar      ;2
                goto    hi1
                rrf     CODEC
                goto    saltar5

;*************************************************************************
;*          NIVEL DEL RELOJ DE CODIFICACION EN BAJO                      *
;*************************************************************************

saltar3         nop
                nop
                nop
                call    lower                                               ;2
                nop
                nop
                nop
                bsf     CLKDEC
                nop
                nop
lowi            btfsc   PIR1,TMR1IF                                         ;2
                goto    saltar2
                btfss   DAT_IN      ;chequea si la entrada a codificar      ;2
                goto    leveow      ;es cero si lo es brinca un espacio     ;2
                bsf     CODEC,1
                btfsc   CODEC,0     ;chequea si la entrada a codificar      ;2
                goto    lowi
                rrf     CODEC
```

```
                goto    saltar4
leveow          bcf     CODEC,1
                btfss   CODEC,0     ;chequea si la entrada a codificar      ;2
                goto    lowi
                rrf     CODEC
                goto    saltar4

saltar2         nop
                nop
                nop
                call    lower                                               ;2
                btfsc   CODIGO,0                                            ;2
                goto    inversor1
                nop
                bsf     CLKDEC
                goto    lowi1
inversor1       bcf     CLKDEC
                nop
                nop
lowi1           btfsc   PIR1,TMR1IF                                         ;2
                goto    saltar6
                btfss   DAT_IN      ;chequea si la entrada a codificar      ;2
                goto    leveow1 ;es cero si lo es brinca un espacio         ;2
                bsf     CODEC,1
                btfsc   CODEC,0     ;chequea si la entrada a codificar      ;2

                goto    lowi1
                rrf     CODEC
                goto    saltar
leveow1         bcf     CODEC,1
                btfss   CODEC,0     ;chequea si la entrada a codificar      ;2
                goto    lowi1
                rrf     CODEC
                goto    saltar
saltar6         btfss   CODIGO,1
                goto    saltar4
                goto    salto

;*********************************************************************
;*          SALTO ALTERNATIVO DE CODIFICACION                        *
;*********************************************************************

saltar5         call    higher                                              ;2
                btfsc   CODIGO,0                                            ;2
                goto    inversor2
                nop
                bcf     CLKDEC
                goto    alta6
inversor2       bcf     CLKDEC
                nop
                nop
alta6           btfsc   PIR1,TMR1IF                                         ;2
                goto    saltar1
                btfsc   DAT_IN      ;chequea si la entrada a codificar      ;2
                goto    levehi2     ;es cero si lo es brinca un espacio     ;2
                bcf     CODEC,1
                btfss   CODEC,0     ;chequea si la entrada a codificar      ;2
                goto    alta6
                rrf     CODEC
```

```
                goto    saltar4
levehi2         bsf     CODEC,1
                btfsc   CODEC,0     ;chequea si la entrada a codificar      ;2
                goto    alta6
                rrf     CODEC
                goto    saltar4

;*************************************************************************
;*                      NIVEL EN ALTO 1                                  *
;*************************************************************************

salto           bcf     CODIGO,7    ;Indica primera lectura
                btfss   CODIGO,1                                            ;2
                goto    saltar4
salto3          call    higher                                              ;2
                bsf     CLKDEC
                bcf     CLKDECI
alto            btfsc   PIR1,TMR1IF                                         ;2
                goto    salto1
                btfsc   DAT_IN      ;chequea si la entrada a codificar      ;2
                goto    levelhi     ;es cero si lo es brinca un espacio     ;2
                bcf     CODEC,1
                btfss   CODEC,0     ;chequea si la entrada a codificar      ;2
                goto    alto
                rrf     CODEC
                goto    salto
levelhi         bsf     CODEC,1
                btfsc   CODEC,0     ;chequea si la entrada a codificar      ;2
                goto    alto
                rrf     CODEC
                goto    salto
salto1          btfsc   CODIGO,7                                            ;2
                goto    salto2
                bsf     CODIGO,7
                nop
                nop
                nop
                goto    salto3

salto2          bcf     CODIGO,7
                btfss   CODIGO,1                                            ;2
                goto    saltar4
                call    higher                                              ;2
                bsf     CLKDEC
                bcf     CLKDECI
alto2           btfsc   PIR1,TMR1IF                                         ;2
                goto    alto1
                btfsc   DAT_IN      ;chequea si la entrada a codificar      ;2
                goto    levelhi1 ;es cero si lo es brinca un espacio        ;2
                bcf     CODEC,1
                btfss   CODEC,0     ;chequea si la entrada a codificar      ;2
                goto    alto2
                rrf     CODEC
                goto    salto
levelhi1        bsf     CODEC,1
                btfsc   CODEC,0     ;chequea si la entrada a codificar      ;2
                goto    alto2
                rrf     CODEC
                goto    salto
```

```
;*************************************************************************
;*                        NIVEL EN BAJO 1                                *
;*************************************************************************

alto1       bcf     CODIGO,7
            btfss   CODIGO,1    ;otros códigos                          ;2
            goto    saltar4
            call    lower                                               ;2
            bsf     CLKDEC
            bcf     CLKDECI
bajo1       btfsc   PIR1,TMR1IF                                         ;2
            goto    bajo
            btfsc   DAT_IN      ;chequea si la entrada a codificar      ;2
            goto    levelow ;es cero si lo es brinca un espacio         ;2
            bcf     CODEC,1
            btfss   CODEC,0     ;chequea si la entrada a codificar      ;2
            goto    bajo1
            rrf     CODEC
            goto    salto5
levelow     bsf     CODEC,1
            btfsc   CODEC,0     ;chequea si la entrada a codificar      ;2
            goto    bajo1
            rrf     CODEC
            goto    salto5

bajo        bcf     CODIGO,7
            btfss   CODIGO,1    ;otros códigos                          ;2
            goto    saltar4
salto6      call    lower                                               ;2
            bcf     CLKDEC
            bsf     CLKDECI
bajo2       btfsc   PIR1,TMR1IF                                         ;2
            goto    salto4
            btfss   DAT_IN      ;chequea si la entrada a codificar      ;2
            goto    levelow1    ;es cero si lo es brinca un espacio     ;2
            bsf     CODEC,1
            btfsc   CODEC,0     ;chequea si la entrada a codificar      ;2
            goto    bajo2
            rrf     CODEC
            goto    salto5
levelow1    bcf     CODEC,1
            btfss   CODEC,0     ;chequea si la entrada a codificar      ;2
            goto    bajo2
            rrf     CODEC
            goto    salto5
salto4      btfsc   CODIGO,7                                            ;2
            goto    salto5
            bsf     CODIGO,7
            nop
            nop
            nop
            goto    salto6

;*************************************************************************
;*                        NIVEL EN ALTO2                                 *
;*************************************************************************

salto5      bcf     CODIGO,7
```

```
                btfss   CODIGO,1                                        ;2
                goto    saltar4         ;otros códigos
salto8          call    higher                                          ;2
                bcf     CLKDEC
                bsf     CLKDECI
alto3           btfsc   PIR1,TMR1IF                                     ;2
                goto    salto9
                btfsc   DAT_IN          ;chequea si la entrada a codificar ;2
                goto    levelhi2 ;es cero si lo es brinca un espacio    ;2
                bcf     CODEC,1
                btfss   CODEC,0         ;chequea si la entrada a codificar ;2
                goto    alto3
                rrf     CODEC
                goto    salto5
levelhi2        bsf     CODEC,1
                btfsc   CODEC,0         ;chequea si la entrada a codificar ;2
                goto    alto3
                rrf     CODEC
                goto    salto5
salto9          btfsc   CODIGO,7                                        ;2
                goto    salto7
                bsf     CODIGO,7
                nop
                nop
                nop
                goto    salto8

salto7          bcf     CODIGO,7
                btfss   CODIGO,1                                        ;2
                goto    saltar4
                call    higher                                          ;2
                bsf     CLKDEC
                bcf     CLKDECI
alto4           btfsc   PIR1,TMR1IF                                     ;2
                goto    alto5
                btfsc   DAT_IN          ;chequea si la entrada a codificar ;2
                goto    levelhi3 ;es cero si lo es brinca un espacio    ;2
                bcf     CODEC,1
                btfss   CODEC,0         ;chequea si la entrada a codificar ;2
                goto    alto4
                rrf     CODEC
                goto    salto
levelhi3        bsf     CODEC,1
                btfsc   CODEC,0         ;chequea si la entrada a codificar ;2
                goto    alto4
                rrf     CODEC
                goto    salto

;*********************************************************************
;*                      NIVEL EN BAJO 2                              *
;*********************************************************************

alto5           bcf     CODIGO,7
                btfss   CODIGO,1        ;otros códigos                  ;2
                goto    saltar4
                call    lower                                           ;2
                bsf     CLKDEC
                bcf     CLKDECI
bajo3           btfsc   PIR1,TMR1IF                                     ;2
```

```
                goto    bajo4
                btfsc   DAT_IN      ;chequea si la entrada a codificar      ;2
                goto    levelow2    ;es cero si lo es brinca un espacio     ;2
                bcf     CODEC,1
                btfss   CODEC,0     ;chequea si la entrada a codificar      ;2
                goto    bajo3
                rrf     CODEC
                goto    salto10
levelow2        bsf     CODEC,1
                btfsc   CODEC,0     ;chequea si la entrada a codificar      ;2
                goto    bajo3
                rrf     CODEC
                goto    salto10

bajo4           bcf     CODIGO,7
                btfss   CODIGO,1    ;otros códigos                          ;2
                goto    saltar4
salto12         call    lower                                               ;2
                bsf     CLKDEC
                bcf     CLKDECI
bajo5           btfsc   PIR1,TMR1IF                                         ;2
                goto    salto11
                btfsc   DAT_IN      ;chequea si la entrada a codificar      ;2
                goto    levelow3    ;es cero si lo es brinca un espacio     ;2
                bcf     CODEC,1
                btfss   CODEC,0     ;chequea si la entrada a codificar      ;2
                goto    bajo5
                rrf     CODEC
                goto    salto10
levelow3        bsf     CODEC,1
                btfsc   CODEC,0     ;chequea si la entrada a codificar      ;2
                goto    bajo5
                rrf     CODEC
                goto    salto10
salto11         btfsc   CODIGO,7                                            ;2
                goto    salto10
                bsf     CODIGO,7
                nop
                nop
                nop
                goto    salto12

;*********************************************************************
;*                      NIVEL EN ALTO 3                              *
;*********************************************************************

salto10         bcf     CODIGO,7
                btfss   CODIGO,1                                            ;2
                goto    saltar4     ;otros códigos
salto14         call    higher                                              ;2
                bcf     CLKDEC
                bsf     CLKDECI
alto6           btfsc   PIR1,TMR1IF                                         ;2
                goto    salto23
                btfsc   DAT_IN      ;chequea si la entrada a codificar      ;2
                goto    levelhi4 ;es cero si lo es brinca un espacio        ;2
                bcf     CODEC,1
                btfss   CODEC,0     ;chequea si la entrada a codificar      ;2
```

```
                goto    alto6
                rrf     CODEC
                goto    salto10
levelhi4        bsf     CODEC,1
                btfsc   CODEC,0     ;chequea si la entrada a codificar      ;2
                goto    alto6
                rrf     CODEC
                goto    salto10
salto23         btfsc   CODIGO,7                                            ;2
                goto    salto13
                bsf     CODIGO,7
                nop
                nop
                nop
                goto    salto14

salto13         bcf     CODIGO,7
                btfss   CODIGO,1    ;otros códigos                          ;2
                goto    saltar4
                call    higher                                              ;2
                bcf     CLKDEC
                bsf     CLKDECI
alto8           btfsc   PIR1,TMR1IF                                         ;2
                goto    alto7
                btfsc   DAT_IN      ;chequea si la entrada a codificar      ;2
                goto    levelhi5 ;es cero si lo es brinca un espacio        ;2
                bcf     CODEC,1
                btfss   CODEC,0     ;chequea si la entrada a codificar      ;2
                goto    alto8
                rrf     CODEC
                goto    salto10
levelhi5        bsf     CODEC,1
                btfsc   CODEC,0     ;chequea si la entrada a codificar      ;2
                goto    alto8
                rrf     CODEC
                goto    salto10

;*****************************************************************************
;*                          NIVEL EN BAJO 3                                  *
;*****************************************************************************

alto7           bcf     CODIGO,7
                btfss   CODIGO,1    ;otros códigos                          ;2
                goto    saltar4
                call    lower                                               ;2
                bcf     CLKDEC
                bsf     CLKDECI
bajo7           btfsc   PIR1,TMR1IF                                         ;2
                goto    bajo6
                btfsc   DAT_IN      ;chequea si la entrada a codificar      ;2
                goto    levelow4    ;es cero si lo es brinca un espacio     ;2
                bcf     CODEC,1
                btfss   CODEC,0     ;chequea si la entrada a codificar      ;2
                goto    bajo7
                rrf     CODEC
                goto    salto15
levelow4        bsf     CODEC,1
                btfsc   CODEC,0     ;chequea si la entrada a codificar      ;2
                goto    bajo7
```

```
                rrf             CODEC
                goto            salto15


bajo6           bcf             CODIGO,7
                btfss           CODIGO,1        ;otros códigos                  ;2
                goto            saltar4
salto16         call            lower                                           ;2
                bsf             CLKDEC
                bcf             CLKDECI
bajo8           btfsc           PIR1,TMR1IF                                     ;2
                goto            salto17
                btfss           DAT_IN          ;chequea si la entrada a codificar ;2
                goto            levelow5        ;es cero si lo es brinca un espacio ;2
                bsf             CODEC,1
                btfsc           CODEC,0         ;chequea si la entrada a codificar ;2
                goto            bajo8
                rrf             CODEC
                goto            salto15
levelow5        bcf             CODEC,1
                btfss           CODEC,0         ;chequea si la entrada a codificar ;2
                goto            bajo8
                rrf             CODEC
                goto            salto15
salto17         btfsc           CODIGO,7                                        ;2
                goto            salto15
                bsf             CODIGO,7
                nop
                nop
                nop
                goto            salto16

;*********************************************************************************
;*                          NIVEL EN ALTO 4                                      *
;*********************************************************************************

salto15         bcf             CODIGO,7
                btfss           CODIGO,1                                        ;2
                goto            saltar          ; Otros códigos
                bcf             CODIGO,7        ; Indica lectura en alto
salto19         call            higher                                          ;2
                bsf             CLKDEC
                bcf             CLKDECI
alto9           btfsc           PIR1,TMR1IF                                     ;2
                goto            salto20
                btfsc           DAT_IN          ;chequea si la entrada a codificar ;2
                goto            levelhi6 ;es cero si lo es brinca un espacio    ;2
                bcf             CODEC,1
                btfss           CODEC,0         ;chequea si la entrada a codificar ;2
                goto            alto9
                rrf             CODEC
                goto            salto15
levelhi6        bsf             CODEC,1
                btfsc           CODEC,0         ;chequea si la entrada a codificar ;2
                goto            alto9
                rrf             CODEC
                goto            salto15
salto20         btfsc           CODIGO,7                                        ;2
                goto            salto18
```

```
                bsf     CODIGO,7
                nop
                nop
                nop
                goto    salto19

salto18         bcf     CODIGO,7
                btfss   CODIGO,1                                        ;2
                goto    saltar4
                call    higher                                          ;2
                bcf     CLKDEC
                bsf     CLKDECI
alto11          btfsc   PIR1,TMR1IF                                     ;2
                goto    alto10
                btfsc   DAT_IN          ;chequea si la entrada a codificar      ;2
                goto    levelhi7 ;es cero si lo es brinca un espacio    ;2
                bcf     CODEC,1
                btfss   CODEC,0         ;chequea si la entrada a codificar      ;2

                goto    alto11
                rrf     CODEC
                goto    salto
levelhi7        bsf     CODEC,1
                btfsc   CODEC,0         ;chequea si la entrada a codificar      ;2
                goto    alto11
                rrf     CODEC
                goto    salto

;*************************************************************************
;*                      NIVEL EN BAJO 3                                  *
;*************************************************************************

alto10          bcf     CODIGO,7
                btfss   CODIGO,1                                        ;2
                goto    saltar4
                call    lower                                           ;2
                bcf     CLKDEC
                bsf     CLKDECI
bajo10          btfsc   PIR1,TMR1IF                                     ;2
                goto    bajo9
                btfsc   DAT_IN          ;chequea si la entrada a codificar      ;2
                goto    levelow6        ;es cero si lo es brinca un espacio     ;2
                bcf     CODEC,1
                btfss   CODEC,0         ;chequea si la entrada a codificar      ;2
                goto    bajo10
                rrf     CODEC
                goto    salto
levelow6        bsf     CODEC,1
                btfsc   CODEC,0         ;chequea si la entrada a codificar      ;2
                goto    bajo10
                rrf     CODEC
                goto    salto
bajo9           bcf     CODIGO,7
                btfss   CODIGO,1        ;otros códigos                  ;2
                goto    saltar4
salto22         call    lower                                           ;2
                bcf     CLKDEC
                bsf     CLKDECI
bajo11          btfsc   PIR1,TMR1IF                                     ;2
```

```
            goto        salto21
            btfss       DAT_IN      ;chequea si la entrada a codificar      ;2
            goto        levelow7    ;es cero si lo es brinca un espacio     ;2
            bsf         CODEC,1
            btfsc       CODEC,0     ;chequea si la entrada a codificar      ;2
            goto        bajo11
            rrf         CODEC
            goto        salto
levelow7    bcf         CODEC,1
            btfss       CODEC,0     ;chequea si la entrada a codificar      ;2
            goto        bajo11
            rrf         CODEC
            goto        salto
salto21     btfsc       CODIGO,7                                            ;2
            goto        salto
            bsf         CODIGO,7
            nop
            nop
            nop
            goto        salto22

            ORG         0x2000
            DATA        0x0F
            DATA        0x0F
            DATA        0x0F
            DATA        0x0F


            ORG         0x2007
            DATA        0x12

            END
```

CÓDIGO FUENTE DEL SOFTWARE CODEC PORTÁTIL

FORMULARIO ACERCA DE LOS CÓDIGOS DE LÍNEA

```
Dim letra As Variant
Dim p As Variant

Option Explicit

Private Sub Command1_Click()
'llama al formulario para visualizar los espectros de potencia
   Unload Me
   Espectros.Show
End Sub

Private Sub Command2_Click()
'Permite seleccionar el código de línea
   If Combo1.Text <> "" Then
      If Command2.Caption = "S&eleccionar" Then  'Permite confirmar la selección del código de línea
         Option1.Enabled = True
         Option2.Enabled = True
         Option3.Enabled = True
         Command2.Caption = "&Cambiar"           'Cambia la etiqueta del comando boton seleccionar
```

```
        Label1.Caption = "Código de Línea"   'Cambia la etiqueta "Seleccione el código de línea" por
        "Código de Línea"
        Combo1.Enabled = False
        'Permite habilitar los botones para ir a la accion correspondiente
        Toolbar1.Enabled = True
    ElseIf Command2.Caption = "&Cambiar" Then 'Permite cambiar el código anteriormente seleccionado
        Option1 = False
        Option2 = False
        Option3 = False
        Combo1.Enabled = True
        Option1.Enabled = False
        Option2.Enabled = False
        Option3.Enabled = False
        Command2.Caption = "S&eleccionar"          'Cambia la etiqueta del comando boton seleccionar
anteriormente ya cambiado
        Label1.Caption = "Seleccione Código de Línea"  'Cambia la etiqueta "Código de Línea" por
"Seleccione el código de línea"
        'Permite desabilitar los botones para ir a la accion correspondiente
        Toolbar1.Enabled = False
    End If
  ElseIf Combo1.Text = "" Then 'Si no ha seleccionado un código de línea, le indica que debe escoger uno
    p = MsgBox("Por favor seleccione el código de línea", vbCritical, "Error")
  End If
End Sub

Private Sub Command4_Click()
'Termina el programa cerrando el puerto si es necesario
  If MSComm1.PortOpen = True Then
      Unload Me
      End
  Else
      End
  End If
End Sub

Private Sub mnuArchivoSalir_Click()
'Termina el programa cerrando el puerto si es necesario
  If MSComm1.PortOpen = True Then
      Unload Me
      End
  Else
      End
  End If
End Sub

Private Sub mnuDispositivoDetectado_Click()
'Llama al formulario para configurar el dispositivo
  Unload Me
  Main.Show
End Sub

Private Sub Option1_Click()
' Permite Visualizar el formulario de Información del código seleccionado
  If Combo1.Text = "NRZ Polar" Then
    Unload Me
    InfNRZPolar.Show
  ElseIf Combo1.Text = "AMI" Then
    Unload Me
    InfAMI.Show
```

```
    ElseIf Combo1.Text = "RZ Polar" Then
        Unload Me
        InfRZPolar.Show
    ElseIf Combo1.Text = "Manchester Diferencial" Then
        Unload Me
        InfManchesterDiferencial.Show
    ElseIf Combo1.Text = "Bifase M" Then
        Unload Me
        InfBifaseM.Show
    ElseIf Combo1.Text = "Miller" Then
        Unload Me
        InfMiller.Show
    ElseIf Combo1.Text = "4B3T" Then
        Unload Me
        Inf4B3T.Show
    ElseIf Combo1.Text = "MS43" Then
        Unload Me
        InfMS43.Show
    ElseIf Combo1.Text = "B3ZS" Then
        Unload Me
        InfB3ZS.Show
    ElseIf Combo1.Text = "HDB3" Then
        Unload Me
        InfHDB3.Show
    ElseIf Combo1.Text = "MLT3" Then
        Unload Me
        InfMLT3.Show
    ElseIf Combo1.Text = "CMI" Then
        Unload Me
        InfCMI.Show
    End If
End Sub

Private Sub Option2_Click()
' Permite visualizar el formulario de la aplicación para la codificacion con el código seleccionado
    If Combo1.Text = "NRZ Polar" Then
        Unload Me
        CodNRZPolar.Show
    ElseIf Combo1.Text = "AMI" Then
        Unload Me
        CodAMI.Show
    ElseIf Combo1.Text = "RZ Polar" Then
        Unload Me
        CodRZPolar.Show
    ElseIf Combo1.Text = "Manchester Diferencial" Then
        Unload Me
        CodManchesterDiferencial.Show
    ElseIf Combo1.Text = "Bifase M" Then
        Unload Me
        CodBifaseM.Show
    ElseIf Combo1.Text = "Miller" Then
        Unload Me
        CodMiller.Show
    ElseIf Combo1.Text = "4B3T" Then
        Unload Me
        Cod4B3T.Show
    ElseIf Combo1.Text = "MS43" Then
        Unload Me
        CodMS43.Show
```

```
    ElseIf Combo1.Text = "B3ZS" Then
        Unload Me
        CodB3ZS.Show
    ElseIf Combo1.Text = "HDB3" Then
        Unload Me
        CodHDB3.Show
    ElseIf Combo1.Text = "MLT3" Then
        Unload Me
        CodMLT3.Show
    ElseIf Combo1.Text = "CMI" Then
        Unload Me
        CodCMI.Show
    End If
End Sub

Private Sub Option3_Click()
' Permite visualizar el formulario de la aplicación para la decodificacion con el código seleccionado
    If Combo1.Text = "NRZ Polar" Then
        Unload Me
        DecNRZPolar.Show
    ElseIf Combo1.Text = "AMI" Then
        Unload Me
        DecAMI.Show
    ElseIf Combo1.Text = "RZ Polar" Then
        Unload Me
        DecRZPolar.Show
    ElseIf Combo1.Text = "Manchester Diferencial" Then
        Unload Me
        DecManchesterDiferencial.Show
    ElseIf Combo1.Text = "Bifase M" Then
        Unload Me
        DecBifaseM.Show
    ElseIf Combo1.Text = "Miller" Then
        Unload Me
        DecMiller.Show
    ElseIf Combo1.Text = "4B3T" Then
        Unload Me
        Dec4B3T.Show
    ElseIf Combo1.Text = "MS43" Then
        Unload Me
        DecMS43.Show
    ElseIf Combo1.Text = "B3ZS" Then
        Unload Me
        DecB3ZS.Show
    ElseIf Combo1.Text = "HDB3" Then
        Unload Me
        DecHDB3.Show
    ElseIf Combo1.Text = "MLT3" Then
        Unload Me
        DecMLT3.Show
    ElseIf Combo1.Text = "CMI" Then
        Unload Me
        DecCMI.Show
    End If
End Sub

Private Sub Form_Load()
'Permite habilitar la opcion de configurar el dispositivo una vez que el codec este conectado
    If CODEC = 1 Then
```

```
      mnuDispositivoDetectado.Enabled = True
      'Configura y abre el puerto para la comunicación con el dispositivo
      MSComm1.CommPort = COM
      MSComm1.PortOpen = True ' ABRO EL PUERTO
   Else
      mnuDispositivoDetectado.Enabled = False
      Timer1.Enabled = False              'Detiene el timer
   End If
End Sub

Private Sub Form_Unload(Cancel As Integer)
' Cierra el puerto al momento de descargar el formulario
   If CODEC = 1 Then
      MSComm1.PortOpen = False
   Else
   End If
End Sub

Private Sub MSComm1_OnComm()
'SE EJECUTA SIEMPRE HAY UN DATO EN PUERTO SERIAL
'Permite la recepción de un dato para detectar si el codec esta o no conectado
   Timer1.Enabled = False 'Detiene el timer
   letra = MSComm1.Input
   If letra = "C" Then
      CODEC = 1

   Else
   End If
   Timer1.Enabled = True   'Inicializa el timer
End Sub

Private Sub Timer1_Timer()
'Timer que permite detectar si el codec portátil fue desconectado
   CODEC = 0
   p = MsgBox("El CODEC ha sido Desconectado", vbInformation, "Información")
   MSComm1.PortOpen = False ' CIERRO EL PUERTO
   Timer1.Enabled = False
   mnuDispositivoDetectado.Enabled = False
End Sub
Private Sub SysInfo1_DeviceArrival(ByVal tipoDispositivo As Long, ByVal IdDispositivo As Long, ByVal
nombreDispositivo As String, ByVal datosDispositivo As Long)
'Se produce cuando se agrega un nuevo dispositivo al sistema
   p = MsgBox("Nuevo Hardware encontrado", vbInformation, "Información")
   p = MsgBox("El CODEC esta listo para ser usado", vbInformation, "Información")
   mnuDispositivoDetectado.Enabled = True
 'Configura y abre el puerto para la comunicación con el dispositivo
   MSComm1.CommPort = COM
   MSComm1.PortOpen = True ' ABRO EL PUERTO
   MSComm1.Output = "Z"
   CODEC = 1
End Sub

Private Sub Toolbar1_ButtonClick(ByVal Button As MSComctlLib.Button)
'Permite seleccionar la accion a realizar
   Select Case Button.Key
      Case "Configurar"
         'Llama al formulario para configurar el dispositivo
         If CODEC = 1 Then
            Unload Me
```

```
            Main.Show
        Else
            p = MsgBox("No se puede configurar el CODEC porque aun no esta conectado", vbInformation,
"Información")
        End If
    Case "Información"
        ' Permite Visualizar el formulario de Información del código seleccionado
        If Combo1.Text = "NRZ Polar" Then
            Unload Me
            InfNRZPolar.Show
        ElseIf Combo1.Text = "AMI" Then
            Unload Me
            InfAMI.Show
        ElseIf Combo1.Text = "RZ Polar" Then
            Unload Me
            InfRZPolar.Show
        ElseIf Combo1.Text = "Manchester Diferencial" Then
            Unload Me
            InfManchesterDiferencial.Show
        ElseIf Combo1.Text = "Bifase M" Then
            Unload Me
            InfBifaseM.Show
        ElseIf Combo1.Text = "Miller" Then
            Unload Me
            InfMiller.Show
        ElseIf Combo1.Text = "4B3T" Then
            Unload Me
            Inf4B3T.Show
        ElseIf Combo1.Text = "MS43" Then
            Unload Me
            InfMS43.Show
        ElseIf Combo1.Text = "B3ZS" Then
            Unload Me
            InfB3ZS.Show
        ElseIf Combo1.Text = "HDB3" Then
            Unload Me
            InfHDB3.Show
        ElseIf Combo1.Text = "MLT3" Then
            Unload Me
            InfMLT3.Show
        ElseIf Combo1.Text = "CMI" Then
            Unload Me
            InfCMI.Show
        End If
    Case "Codificar"
        ' Permite visualizar el formulario de la aplicación para la codificacion con el código seleccionado
        If Combo1.Text = "NRZ Polar" Then
            Unload Me
            CodNRZPolar.Show
        ElseIf Combo1.Text = "AMI" Then
            Unload Me
            CodAMI.Show
        ElseIf Combo1.Text = "RZ Polar" Then
            Unload Me
            CodRZPolar.Show
        ElseIf Combo1.Text = "Manchester Diferencial" Then
            Unload Me
            CodManchesterDiferencial.Show
        ElseIf Combo1.Text = "Bifase M" Then
```

```
        Unload Me
        CodBifaseM.Show
    ElseIf Combo1.Text = "Miller" Then
        Unload Me
        CodMiller.Show
    ElseIf Combo1.Text = "4B3T" Then
        Unload Me
        Cod4B3T.Show
    ElseIf Combo1.Text = "MS43" Then
        Unload Me
        CodMS43.Show
    ElseIf Combo1.Text = "B3ZS" Then
        Unload Me
        CodB3ZS.Show
    ElseIf Combo1.Text = "HDB3" Then
        Unload Me
        CodHDB3.Show
    ElseIf Combo1.Text = "MLT3" Then
        Unload Me
        CodMLT3.Show
    ElseIf Combo1.Text = "CMI" Then
        Unload Me
        CodCMI.Show
    End If
Case "Decodificar"
    ' Permite visualizar el formulario de la aplicación para la decodificacion con el código seleccionado
    If Combo1.Text = "NRZ Polar" Then
        Unload Me
        DecNRZPolar.Show
    ElseIf Combo1.Text = "AMI" Then
        Unload Me
        DecAMI.Show
    ElseIf Combo1.Text = "RZ Polar" Then
        Unload Me
        DecRZPolar.Show
    ElseIf Combo1.Text = "Manchester Diferencial" Then
        Unload Me
        DecManchesterDiferencial.Show
    ElseIf Combo1.Text = "Bifase M" Then
        Unload Me
        DecBifaseM.Show
    ElseIf Combo1.Text = "Miller" Then
        Unload Me
        DecMiller.Show
    ElseIf Combo1.Text = "4B3T" Then
        Unload Me
        Dec4B3T.Show
    ElseIf Combo1.Text = "MS43" Then
        Unload Me
        DecMS43.Show
    ElseIf Combo1.Text = "B3ZS" Then
        Unload Me
        DecB3ZS.Show
    ElseIf Combo1.Text = "HDB3" Then
        Unload Me
        DecHDB3.Show
    ElseIf Combo1.Text = "MLT3" Then
        Unload Me
        DecMLT3.Show
```

```
        ElseIf Combo1.Text = "CMI" Then
           Unload Me
           DecCMI.Show
        End If
      Case "D.E.P"
        'llama al formulario para visualizar los espectros de potencia
        Unload Me
        Espectros.Show
      Case "Salir"
        'Termina el programa cerrando el puerto si es necesario
        If MSComm1.PortOpen = True Then
           Unload Me
           End
        Else
           End
        End If

   End Select
End Sub
```

FORMULARIO CODIFICAR CON EL CÓDIGO DE LÍNEA AMI

```
'Declaración de variables
Dim vec1(1 To 100) As Variant   'Vector de datos
Dim vec2(1 To 200) As Variant   'Vector de respuesta

Dim codigo As Variant
Dim aux As Integer

Dim d As Variant
Dim r As Variant
Dim f As Variant
Dim m As Variant
Dim i As Variant
Dim p As Variant
Dim l As Variant
Dim letra As Variant

Option Explicit

Private Sub Command2_Click()
'Graficara las señales de: reloj, datos y codificada
   Call GRAFICA 'Llama al procedimiento Sub Grafica
End Sub

Sub GRAFICA()
'Procedimiento Sub Grafica, que graficará las señales de: reloj, datos y codificada
aux = 1
r = 1
Picture1.Cls
codigo = ""
Text2.Text = ""

'Algoritmo de codificación
      For d = 1 To Len(Text1.Text)
         vec1(d) = Mid(Text1.Text, d, 1)
         If vec1(d) = 1 And aux = 1 Then 'Si el dato es 1L y (aux =1), el nivel de la señal codificada es
positivo "+"
               vec2(r) = "+"
```

```
            aux = 0  'Guardamos valor 0 para indicar que el siguiente 1L tendrá un nivel de señal codificada
negativa "-"
            r = r + 1
        ElseIf vec1(d) = 1 And aux = 0 Then 'Si el dato es 1L y (aux =0), el nivel de la señal codificada es
negativo "-"
            vec2(r) = "-"
            aux = 1  'Guardamos valor 0 para indicar que el siguiente 1L tendrá un nivel de señal codificada
positiva "-"
            r = r + 1
        Else   'Si el dato es 0L, el nivel de la señal codificada sera cero "0"
          vec2(r) = 0
          r = r + 1
        End If

      Next

'Imprimo los valores codificados
   For d = 1 To Len(Text1.Text)
      codigo = codigo & vec2(d)
   Next
   Text2.Text = codigo

'Escoje los valores seleccionados por el usuario
If Val(Combo1.Text) = 19200 And Val(Combo2.Text) = 100 Then f = 0.125
If Val(Combo1.Text) = 19200 And Val(Combo2.Text) = 25 Then f = 0.125 / 4
If Val(Combo1.Text) = 19200 And Val(Combo2.Text) = 50 Then f = 0.125 / 2
If Val(Combo1.Text) = 19200 And Val(Combo2.Text) = 75 Then f = 0.125 * 3 / 4
If Val(Combo1.Text) = 19200 And Val(Combo2.Text) = 150 Then f = 0.125 * 3 / 2
If Val(Combo1.Text) = 19200 And Val(Combo2.Text) = 300 Then f = 0.125 * 3
If Val(Combo1.Text) = 19200 And Val(Combo2.Text) = 400 Then f = 0.125 * 4

If Val(Combo1.Text) = 9600 And Val(Combo2.Text) = 100 Then f = 0.25
If Val(Combo1.Text) = 9600 And Val(Combo2.Text) = 25 Then f = 0.25 / 4
If Val(Combo1.Text) = 9600 And Val(Combo2.Text) = 50 Then f = 0.25 / 2
If Val(Combo1.Text) = 9600 And Val(Combo2.Text) = 75 Then f = 0.25 * 3 / 4
If Val(Combo1.Text) = 9600 And Val(Combo2.Text) = 150 Then f = 0.25 * 3 / 2
If Val(Combo1.Text) = 9600 And Val(Combo2.Text) = 300 Then f = 0.25 * 3
If Val(Combo1.Text) = 9600 And Val(Combo2.Text) = 400 Then f = 0.25 * 4

If Val(Combo1.Text) = 4800 And Val(Combo2.Text) = 100 Then f = 0.5
If Val(Combo1.Text) = 4800 And Val(Combo2.Text) = 25 Then f = 0.5 / 4
If Val(Combo1.Text) = 4800 And Val(Combo2.Text) = 50 Then f = 0.5 / 2
If Val(Combo1.Text) = 4800 And Val(Combo2.Text) = 75 Then f = 0.5 * 3 / 4
If Val(Combo1.Text) = 4800 And Val(Combo2.Text) = 150 Then f = 0.5 * 3 / 2
If Val(Combo1.Text) = 4800 And Val(Combo2.Text) = 300 Then f = 0.5 * 3
If Val(Combo1.Text) = 4800 And Val(Combo2.Text) = 400 Then f = 0.5 * 4


If Val(Combo1.Text) = 2400 And Val(Combo2.Text) = 100 Then f = 1
If Val(Combo1.Text) = 2400 And Val(Combo2.Text) = 25 Then f = 1 / 4
If Val(Combo1.Text) = 2400 And Val(Combo2.Text) = 50 Then f = 1 / 2
If Val(Combo1.Text) = 2400 And Val(Combo2.Text) = 75 Then f = 1 * 3 / 4
If Val(Combo1.Text) = 2400 And Val(Combo2.Text) = 150 Then f = 1 * 3 / 2
If Val(Combo1.Text) = 2400 And Val(Combo2.Text) = 300 Then f = 1 * 3
If Val(Combo1.Text) = 2400 And Val(Combo2.Text) = 400 Then f = 1 * 4

If Val(Combo1.Text) = 1200 And Val(Combo2.Text) = 100 Then f = 2
If Val(Combo1.Text) = 1200 And Val(Combo2.Text) = 25 Then f = 2 / 4
If Val(Combo1.Text) = 1200 And Val(Combo2.Text) = 50 Then f = 2 / 2
```

```
If Val(Combo1.Text) = 1200 And Val(Combo2.Text) = 75 Then f = 2 * 3 / 4
If Val(Combo1.Text) = 1200 And Val(Combo2.Text) = 150 Then f = 2 * 3 / 2
If Val(Combo1.Text) = 1200 And Val(Combo2.Text) = 300 Then f = 2 * 3
If Val(Combo1.Text) = 1200 And Val(Combo2.Text) = 400 Then f = 2 * 4

If Val(Combo1.Text) = 600 And Val(Combo2.Text) = 100 Then f = 4
If Val(Combo1.Text) = 600 And Val(Combo2.Text) = 25 Then f = 4 / 4
If Val(Combo1.Text) = 600 And Val(Combo2.Text) = 50 Then f = 4 / 2
If Val(Combo1.Text) = 600 And Val(Combo2.Text) = 75 Then f = 4 * 3 / 4
If Val(Combo1.Text) = 600 And Val(Combo2.Text) = 150 Then f = 4 * 3 / 2
If Val(Combo1.Text) = 600 And Val(Combo2.Text) = 300 Then f = 4 * 3
If Val(Combo1.Text) = 600 And Val(Combo2.Text) = 400 Then f = 4 * 4

If Val(Combo1.Text) = 300 And Val(Combo2.Text) = 100 Then f = 8
If Val(Combo1.Text) = 300 And Val(Combo2.Text) = 25 Then f = 8 / 4
If Val(Combo1.Text) = 300 And Val(Combo2.Text) = 50 Then f = 8 / 2
If Val(Combo1.Text) = 300 And Val(Combo2.Text) = 75 Then f = 8 * 3 / 4
If Val(Combo1.Text) = 300 And Val(Combo2.Text) = 150 Then f = 8 * 3 / 2
If Val(Combo1.Text) = 300 And Val(Combo2.Text) = 300 Then f = 8 * 3
If Val(Combo1.Text) = 300 And Val(Combo2.Text) = 400 Then f = 8 * 4


If Val(Combo2.Text) = 25 Then m = 1 / 4
If Val(Combo2.Text) = 50 Then m = 1 / 2
If Val(Combo2.Text) = 75 Then m = 3 / 4
If Val(Combo2.Text) = 100 Then m = 1
If Val(Combo2.Text) = 150 Then m = 3 / 2
If Val(Combo2.Text) = 300 Then m = 3
If Val(Combo2.Text) = 400 Then m = 4

' Grafico de lineas de fondo
For i = 0 To 1000
Picture1.DrawStyle = 2
Picture1.Line (475 + 365 * i * m, 0)-(475 + 365 * i * m, 5400), QBColor(7)
Picture1.Line (0, 540 * i)-(12000, 540 * i), QBColor(7)
Next i

'Grafico de ejes
Picture1.DrawStyle = 6
Picture1.Line (240, 1080)-(11300, 1080), RGB(78, 123, 234)
Picture1.Line (480, 360)-(480, 1440), RGB(78, 123, 234)

Picture1.Line (240, 2700)-(11300, 2700), RGB(78, 123, 234)
Picture1.Line (480, 2040)-(480, 3360), RGB(78, 123, 234)

Picture1.Line (240, 4320)-(11300, 4320), RGB(78, 123, 234)
Picture1.Line (480, 3720)-(480, 4680), RGB(78, 123, 234)

'Señal de Reloj
For i = 0 To Len(Text1.Text) - 1
   Picture1.Line (475 + 365 * i * f, 540)-(475 + 365 * (i + 0.5) * f, 540), QBColor(12)
   Picture1.Line (475 + 365 * (i + 0.5) * f, 1080)-(475 + 365 * (i + 0.5) * f, 540), QBColor(12)
   Picture1.Line (475 + 365 * (i + 0.5) * f, 1080)-(475 + 365 * (i + 1) * f, 1080), QBColor(12)
   Picture1.Line (475 + 365 * (i + 1) * f, 1080)-(475 + 365 * (i + 1) * f, 540), QBColor(12)
Next i

'Señal de Datos
aux = 2
For d = 1 To Len(Text1.Text)
```

```
    Picture1.DrawStyle = 6
    If vec1(d) = 0 And aux <> 0 Then
        Picture1.Line (475 + 365 * d * f, 2700)-(475 + 365 * (d - 1) * f, 2700), QBColor(12)
        Picture1.Line (475 + 365 * (d - 1) * f, 2700)-(475 + 365 * (d - 1) * f, 2160), QBColor(12)
    ElseIf vec1(d) = 1 And aux <> 1 Then
        Picture1.Line (475 + 365 * (d - 1) * f, 2700)-(475 + 365 * (d - 1) * f, 2160), QBColor(12)
        Picture1.Line (475 + 365 * d * f, 2160)-(475 + 365 * (d - 1) * f, 2160), QBColor(12)
    ElseIf vec1(d) = 0 And aux = 0 Then
        Picture1.Line (475 + 365 * d * f, 2700)-(475 + 365 * (d - 1) * f, 2700), QBColor(12)
    ElseIf vec1(d) = 1 And aux = 1 Then
       Picture1.Line (475 + 365 * d * f, 2160)-(475 + 365 * (d - 1) * f, 2160), QBColor(12)
    End If
      aux = vec1(d)
 Next d

'Señal Codificada
For r = 1 To Len(Text1.Text)
    Picture1.DrawStyle = 6
    If vec2(r) = 0 Then
        Picture1.Line (475 + 365 * r * f, 4320)-(475 + 365 * (r - 1) * f, 4320), QBColor(12)
    ElseIf vec2(r) = "+" Then
        Picture1.Line (475 + 365 * (r - 1) * f, 4320)-(475 + 365 * (r - 1) * f, 3780), QBColor(12)
        Picture1.Line (475 + 365 * (r - 1) * f, 3780)-(475 + 365 * r * f, 3780), QBColor(12)
        Picture1.Line (475 + 365 * r * f, 3780)-(475 + 365 * r * f, 4320), QBColor(12)
    ElseIf vec2(r) = "-" Then
        Picture1.Line (475 + 365 * (r - 1) * f, 4320)-(475 + 365 * (r - 1) * f, 4860), QBColor(12)
        Picture1.Line (475 + 365 * (r - 1) * f, 4860)-(475 + 365 * r * f, 4860), QBColor(12)
        Picture1.Line (475 + 365 * r * f, 4860)-(475 + 365 * r * f, 4320), QBColor(12)
    End If
Next r

'Visualizo los valores de los niveles superior e inferior de las señales
Label14.Visible = True
Label11.Visible = True
Label12.Visible = True
Label13.Visible = True

'Uniformidad del color de los ejes de las señales
Picture1.Line (475, 2700)-(475, 2160), RGB(78, 123, 234)
Picture1.Line (475, 4320)-(475, 3780), RGB(78, 123, 234)

End Sub

Private Sub Command3_Click()
'Permite iniciar una nueva codificación
    I = 0
    Text1.Text = ""
    Text2.Text = ""
    Picture1.Cls
End Sub

Private Sub mnuApliEsp_Click()
'llana al formulario para visualizar los espectros de potencia
    Unload Me
    Espectros.Show
End Sub

Private Sub mnuDispositivoDetectado_Click()
'Llama al formulario para configurar el dispositivo
```

```
    Unload Me
    Main.Show
End Sub

Private Sub Text1_KeyPress(KeyAscii As Integer)
'Chequeo que los valores ingresados sean solo los permitidos
    If (KeyAscii = &H30 Or KeyAscii = &H31 Or KeyAscii = &H8) Then
        If (KeyAscii = &H30 Or KeyAscii = &H31) Then
            l = l + 1
        Else
            l = l - 1
        End If

        If l < 40 Then
        Else
        'Chequea que el numero de caracteres ingresados sean los permitidos
            l = l - 1
            p = MsgBox("El número de caracteres permitidos, de la cadena ingresada es 39", vbInformation,
"Información")
            KeyAscii = 0
        End If

    Else
        p = MsgBox("Solo se ingresan datos binarios (0, 1)", vbCritical, "Error")
        KeyAscii = 0
    End If
End Sub

Private Sub Command1_Click()
'Regresar al formulario para seleccionar el código de línea
    Unload Me
    Acerca.Show
End Sub

Private Sub mnuApliDecodificar_Click()
'Visualizo el formulario de la aplicación para decodificar
    Unload Me
    DecAMI.Show
End Sub

Private Sub mnuArchivoSalir_Click()
'Termina el programa cerrando el puerto si es necesario
    If MSComm1.PortOpen = True Then
            Unload Me
            End
    Else
            End
    End If
End Sub

Private Sub mnuVerInfo_Click()
'Visualizo el formulario que presenta la información del código de línea
    Unload Me
    InfAMI.Show
End Sub

Private Sub Combo1_Click()
'Graficara las señales de: reloj, datos y codificada a la escala seleccionada
    Call GRAFICA 'Llama al procedimiento Sub Grafica
```

```
End Sub

Private Sub Combo2_Click()
'Graficara las señales de: reloj, datos y codificada a la escala seleccionada
   Call GRAFICA 'Llama al procedimiento Sub Grafica
End Sub

Private Sub Form_Load()
'Permite habilitar la opcion de configurar el dispositivo una vez que el codec este conectado
   If CODEC = 1 Then
      mnuDispositivoDetectado.Enabled = True
      'Configura y abre el puerto para la comunicación con el dispositivo
      MSComm1.CommPort = COM
      MSComm1.PortOpen = True ' ABRO EL PUERTO
   Else
      mnuDispositivoDetectado.Enabled = False
      Timer1.Enabled = False              'Detiene el timer
   End If
 'Inicializamos el contador de caracteres ingresados
   l = 0
End Sub

Private Sub Form_Unload(Cancel As Integer)
'Cierra el puerto al momento de descargar el formulario
   If CODEC = 1 Then
      MSComm1.PortOpen = False
   Else
   End If
End Sub

Private Sub MSComm1_OnComm()
'SE EJECUTA SIEMPRE HAY UN DATO EN PUERTO SERIAL
'Permite la recepción de un dato para detectar si el codec esta o no conectado
   Timer1.Enabled = False 'Detiene el timer
   letra = MSComm1.Input
   If letra = "C" Then
      CODEC = 1

   Else
   End If
   Timer1.Enabled = True   'Inicializa el timer
End Sub

Private Sub Timer1_Timer()
'Timer que permite detectar si el codec portátil fue desconectado
   CODEC = 0
   p = MsgBox("El CODEC ha sido Desconectado", vbInformation, "Información")
   MSComm1.PortOpen = False ' CIERRO EL PUERTO
   Timer1.Enabled = False
   mnuDispositivoDetectado.Enabled = False
End Sub

Private Sub SysInfo1_DeviceArrival(ByVal tipoDispositivo As Long, ByVal IdDispositivo As Long, ByVal
nombreDispositivo As String, ByVal datosDispositivo As Long)
'Se produce cuando se agrega un nuevo dispositivo al sistema
   p = MsgBox("Nuevo Hardware encontrado", vbInformation, "Información")
   p = MsgBox("El CODEC esta listo para ser usado", vbInformation, "Información")
   mnuDispositivoDetectado.Enabled = True
 'Configura y abre el puerto para la comunicación con el dispositivo
```

```
    MSComm1.CommPort = COM
    MSComm1.PortOpen = True ' ABRO EL PUERTO
    MSComm1.Output = "Z"
    CODEC = 1
End Sub

Private Sub Toolbar1_ButtonClick(ByVal Button As MSComctlLib.Button)
'Permite seleccionar la accion a realizar
    Select Case Button.Key
        Case "Configurar"
            'Llama al formulario para configurar el dispositivo
            If CODEC = 1 Then
                Unload Me
                Main.Show
            Else
                p = MsgBox("No se puede configurar el CODEC porque aun no esta conectado", vbInformation,
"Información")
            End If
        Case "Información"
            ' Permite Visualizar el formulario de Información del código seleccionado
            Unload Me
            InfAMI.Show
        Case "Decodificar"
            ' Permite visualizar el formulario de la aplicación para la decodificacion con el código seleccionado
            Unload Me
            DecAMI.Show
        Case "D.E.P"
            'llama al formulario para visualizar los espectros de potencia
            Unload Me
            Espectros.Show
        Case "Salir"
            'Termina el programa cerrando el puerto si es necesario
            If MSComm1.PortOpen = True Then
                Unload Me
                End
            Else
                End
            End If

    End Select
End Sub
```

FORMULARIO CODIFICAR CON EL CÓDIGO DE LÍNEA 4B3T
FORMULARIO CODIFICAR CON EL CÓDIGO DE LÍNEA B3ZS
FORMULARIO CODIFICAR CON EL CÓDIGO DE LÍNEA BIFASE-M
FORMULARIO CODIFICAR CON EL CÓDIGO DE LÍNEA CMI
FORMULARIO CODIFICAR CON EL CÓDIGO DE LÍNEA HDB3
FORMULARIO CODIFICAR CON EL CÓDIGO DE LÍNEA MANCHESTER DIFERENCIAL
FORMULARIO CODIFICAR CON EL CÓDIGO DE LÍNEA MILLER
FORMULARIO CODIFICAR CON EL CÓDIGO DE LÍNEA MLT-3
FORMULARIO CODIFICAR CON EL CÓDIGO DE LÍNEA MS43

FORMULARIO CODIFICAR CON EL CÓDIGO DE LÍNEA NRZ POLAR

FORMULARIO CODIFICAR CON EL CÓDIGO DE LÍNEA RZ POLAR

FORMULARIO DECODIFICAR CON EL CÓDIGO DE LÍNEA 4B3T

FORMULARIO DECODIFICAR CON EL CÓDIGO DE LÍNEA AMI

```
'Declaración de variables
Dim vec1(0 To 100) As Variant  'Vector de Niveles de la señal codificada
Dim vec2(1 To 200) As Variant  'Vector de respuesta

Dim codigo As Variant
Dim aux As Integer

Dim d As Variant
Dim r As Variant
Dim f As Variant
Dim m As Variant
Dim i As Variant
Dim p As Variant
Dim l As Variant
Dim letra As Variant

Option Explicit
Private Sub Command2_Click()
'Graficara las señales de: reloj, codificada y decodificada (datos)
   Call GRAFICA 'Llama al procedimiento Sub Grafica
End Sub

Sub GRAFICA()
'Procedimiento Sub Grafica, que graficará las señales de: reloj, codificada y decodificada (datos)

aux = 2
r = 1
Picture1.Cls
codigo = ""
Text2.Text = ""

'Algoritmo de decodificación
   For d = 1 To Len(Text1.Text)
        vec1(d) = Mid(Text1.Text, d, 1)
        If vec1(d) = "+" And aux <> 1 Then 'Si la señal tiene nivel positivo (+A), el dato es 1
             vec2(r) = 1
             r = r + 1
             aux = 1
        ElseIf vec1(d) = "+" And aux = 1 Then 'Si la señal tuvo nivel positivo, chequea que el siguiente nivel
de señal no sea positivo
             p = MsgBox("No es una secuencia válida a ser decodificada", vbCritical, "Error")
             p = MsgBox("Los niveles de señal son alternados, y se encontró dos niveles de señal positivos
(+)", vbInformation, "Información")
             Text1.Text = Text1.Text + ""
             GoTo 1

        ElseIf vec1(d) = "-" And aux <> 0 Then 'Si la señal tiene nivel negativo (-A), el dato es 1
             vec2(r) = 1
             r = r + 1
             aux = 0
```

```
        ElseIf vec1(d) = "-" And aux = 0 Then 'Si la señal tuvo nivel negativo, chequea que el siguiente nivel de señal no sea negativo
            p = MsgBox("No es una secuencia válida a ser decodificada", vbCritical, "Error")
            p = MsgBox("Los niveles de señal son alternados, y se encontró dos niveles de señal negativos (-)", vbInformation, "Información")
            Text1.Text = Text1.Text + ""
            GoTo 1

        ElseIf vec1(d) = 0 Then 'Si la señal tiene nivel de señal cero, el dato es 0
            vec2(r) = 0
            r = r + 1
        End If

    Next d

'Imprimo los valores decodificados (Datos)
  For d = 1 To Len(Text1.Text)
    codigo = codigo & vec2(d)
  Next
  Text2.Text = codigo

'Escoje los valores seleccionados por el usuario

If Val(Combo1.Text) = 19200 And Val(Combo2.Text) = 100 Then f = 0.125
If Val(Combo1.Text) = 19200 And Val(Combo2.Text) = 25 Then f = 0.125 / 4
If Val(Combo1.Text) = 19200 And Val(Combo2.Text) = 50 Then f = 0.125 / 2
If Val(Combo1.Text) = 19200 And Val(Combo2.Text) = 75 Then f = 0.125 * 3 / 4
If Val(Combo1.Text) = 19200 And Val(Combo2.Text) = 150 Then f = 0.125 * 3 / 2
If Val(Combo1.Text) = 19200 And Val(Combo2.Text) = 300 Then f = 0.125 * 3
If Val(Combo1.Text) = 19200 And Val(Combo2.Text) = 400 Then f = 0.125 * 4

If Val(Combo1.Text) = 9600 And Val(Combo2.Text) = 100 Then f = 0.25
If Val(Combo1.Text) = 9600 And Val(Combo2.Text) = 25 Then f = 0.25 / 4
If Val(Combo1.Text) = 9600 And Val(Combo2.Text) = 50 Then f = 0.25 / 2
If Val(Combo1.Text) = 9600 And Val(Combo2.Text) = 75 Then f = 0.25 * 3 / 4
If Val(Combo1.Text) = 9600 And Val(Combo2.Text) = 150 Then f = 0.25 * 3 / 2
If Val(Combo1.Text) = 9600 And Val(Combo2.Text) = 300 Then f = 0.25 * 3
If Val(Combo1.Text) = 9600 And Val(Combo2.Text) = 400 Then f = 0.25 * 4

If Val(Combo1.Text) = 4800 And Val(Combo2.Text) = 100 Then f = 0.5
If Val(Combo1.Text) = 4800 And Val(Combo2.Text) = 25 Then f = 0.5 / 4
If Val(Combo1.Text) = 4800 And Val(Combo2.Text) = 50 Then f = 0.5 / 2
If Val(Combo1.Text) = 4800 And Val(Combo2.Text) = 75 Then f = 0.5 * 3 / 4
If Val(Combo1.Text) = 4800 And Val(Combo2.Text) = 150 Then f = 0.5 * 3 / 2
If Val(Combo1.Text) = 4800 And Val(Combo2.Text) = 300 Then f = 0.5 * 3
If Val(Combo1.Text) = 4800 And Val(Combo2.Text) = 400 Then f = 0.5 * 4


If Val(Combo1.Text) = 2400 And Val(Combo2.Text) = 100 Then f = 1
If Val(Combo1.Text) = 2400 And Val(Combo2.Text) = 25 Then f = 1 / 4
If Val(Combo1.Text) = 2400 And Val(Combo2.Text) = 50 Then f = 1 / 2
If Val(Combo1.Text) = 2400 And Val(Combo2.Text) = 75 Then f = 1 * 3 / 4
If Val(Combo1.Text) = 2400 And Val(Combo2.Text) = 150 Then f = 1 * 3 / 2
If Val(Combo1.Text) = 2400 And Val(Combo2.Text) = 300 Then f = 1 * 3
If Val(Combo1.Text) = 2400 And Val(Combo2.Text) = 400 Then f = 1 * 4

If Val(Combo1.Text) = 1200 And Val(Combo2.Text) = 100 Then f = 2
If Val(Combo1.Text) = 1200 And Val(Combo2.Text) = 25 Then f = 2 / 4
If Val(Combo1.Text) = 1200 And Val(Combo2.Text) = 50 Then f = 2 / 2
```

```
If Val(Combo1.Text) = 1200 And Val(Combo2.Text) = 75 Then f = 2 * 3 / 4
If Val(Combo1.Text) = 1200 And Val(Combo2.Text) = 150 Then f = 2 * 3 / 2
If Val(Combo1.Text) = 1200 And Val(Combo2.Text) = 300 Then f = 2 * 3
If Val(Combo1.Text) = 1200 And Val(Combo2.Text) = 400 Then f = 2 * 4

If Val(Combo1.Text) = 600 And Val(Combo2.Text) = 100 Then f = 4
If Val(Combo1.Text) = 600 And Val(Combo2.Text) = 25 Then f = 4 / 4
If Val(Combo1.Text) = 600 And Val(Combo2.Text) = 50 Then f = 4 / 2
If Val(Combo1.Text) = 600 And Val(Combo2.Text) = 75 Then f = 4 * 3 / 4
If Val(Combo1.Text) = 600 And Val(Combo2.Text) = 150 Then f = 4 * 3 / 2
If Val(Combo1.Text) = 600 And Val(Combo2.Text) = 300 Then f = 4 * 3
If Val(Combo1.Text) = 600 And Val(Combo2.Text) = 400 Then f = 4 * 4

If Val(Combo1.Text) = 300 And Val(Combo2.Text) = 100 Then f = 8
If Val(Combo1.Text) = 300 And Val(Combo2.Text) = 25 Then f = 8 / 4
If Val(Combo1.Text) = 300 And Val(Combo2.Text) = 50 Then f = 8 / 2
If Val(Combo1.Text) = 300 And Val(Combo2.Text) = 75 Then f = 8 * 3 / 4
If Val(Combo1.Text) = 300 And Val(Combo2.Text) = 150 Then f = 8 * 3 / 2
If Val(Combo1.Text) = 300 And Val(Combo2.Text) = 300 Then f = 8 * 3
If Val(Combo1.Text) = 300 And Val(Combo2.Text) = 400 Then f = 8 * 4


If Val(Combo2.Text) = 25 Then m = 1 / 4
If Val(Combo2.Text) = 50 Then m = 1 / 2
If Val(Combo2.Text) = 75 Then m = 3 / 4
If Val(Combo2.Text) = 100 Then m = 1
If Val(Combo2.Text) = 150 Then m = 3 / 2
If Val(Combo2.Text) = 300 Then m = 3
If Val(Combo2.Text) = 400 Then m = 4

' Grafico de lineas de fondo
For i = 0 To 1000
Picture1.DrawStyle = 2
Picture1.Line (475 + 365 * i * m, 0)-(475 + 365 * i * m, 5400), QBColor(7)
Picture1.Line (0, 540 * i)-(12000, 540 * i), QBColor(7)
Next i

'Grafico los ejes
Picture1.DrawStyle = 6
Picture1.Line (240, 1080)-(11300, 1080), RGB(78, 123, 234)
Picture1.Line (480, 360)-(480, 1440), RGB(78, 123, 234)

Picture1.Line (240, 2700)-(11300, 2700), RGB(78, 123, 234)
Picture1.Line (480, 2040)-(480, 3360), RGB(78, 123, 234)

Picture1.Line (240, 4320)-(11300, 4320), RGB(78, 123, 234)
Picture1.Line (480, 3720)-(480, 4680), RGB(78, 123, 234)

'Señal de Reloj
For i = 0 To Len(Text1.Text) - 1
    Picture1.Line (475 + 365 * i * f, 540)-(475 + 365 * (i + 0.5) * f, 540), QBColor(12)
    Picture1.Line (475 + 365 * (i + 0.5) * f, 1080)-(475 + 365 * (i + 0.5) * f, 540), QBColor(12)
    Picture1.Line (475 + 365 * (i + 0.5) * f, 1080)-(475 + 365 * (i + 1) * f, 1080), QBColor(12)
    Picture1.Line (475 + 365 * (i + 1) * f, 1080)-(475 + 365 * (i + 1) * f, 540), QBColor(12)
Next i

'Señal Codificada

For d = 1 To Len(Text1.Text)
```

```
   Picture1.DrawStyle = 6
   If vec1(d) = 0 Then
        Picture1.Line (475 + 365 * d * f, 2700)-(475 + 365 * (d - 1) * f, 2700), QBColor(12)
   ElseIf vec1(d) = "+" Then
        Picture1.Line (475 + 365 * (d - 1) * f, 2700)-(475 + 365 * (d - 1) * f, 2160), QBColor(12)
        Picture1.Line (475 + 365 * (d - 1) * f, 2160)-(475 + 365 * d * f, 2160), QBColor(12)
        Picture1.Line (475 + 365 * d * f, 2160)-(475 + 365 * d * f, 2700), QBColor(12)
   ElseIf vec1(d) = "-" Then
        Picture1.Line (475 + 365 * (d - 1) * f, 2700)-(475 + 365 * (d - 1) * f, 3240), QBColor(12)
        Picture1.Line (475 + 365 * (d - 1) * f, 3240)-(475 + 365 * d * f, 3240), QBColor(12)
        Picture1.Line (475 + 365 * d * f, 3240)-(475 + 365 * d * f, 2700), QBColor(12)
   End If
Next d

'Señal Decodificada

aux = 2
For r = 1 To Len(Text1.Text)
   Picture1.DrawStyle = 6

   If vec2(r) = 0 And aux <> 0 Then
        Picture1.Line (475 + 365 * r * f, 4320)-(475 + 365 * (r - 1) * f, 4320), QBColor(12)
        Picture1.Line (475 + 365 * (r - 1) * f, 4320)-(475 + 365 * (r - 1) * f, 3780), QBColor(12)
   ElseIf vec2(r) = 1 And aux <> 1 Then
        Picture1.Line (475 + 365 * (r - 1) * f, 4320)-(475 + 365 * (r - 1) * f, 3780), QBColor(12)
        Picture1.Line (475 + 365 * r * f, 3780)-(475 + 365 * (r - 1) * f, 3780), QBColor(12)
   ElseIf vec2(r) = 0 And aux = 0 Then
        Picture1.Line (475 + 365 * r * f, 4320)-(475 + 365 * (r - 1) * f, 4320), QBColor(12)
   ElseIf vec2(r) = 1 And aux = 1 Then
        Picture1.Line (475 + 365 * r * f, 3780)-(475 + 365 * (r - 1) * f, 3780), QBColor(12)
   End If
        aux = vec2(r)
 Next r

'Visualizo los valores de los niveles superior e inferior de las señales
Label6.Visible = True
Label11.Visible = True
Label12.Visible = True
Label13.Visible = True

'Uniformidad del color de los ejes de las señales
Picture1.Line (475, 3240)-(475, 2160), RGB(78, 123, 234)
Picture1.Line (475, 4320)-(475, 3780), RGB(78, 123, 234)
1
End Sub

Private Sub Command3_Click()
'Habilito los componentes para proceder a la decodificación
   Text1.Enabled = True
   Command2.Enabled = True
   Combo1.Enabled = True
   Combo2.Enabled = True
   Command3.Enabled = False
End Sub

Private Sub Command4_Click()

End Sub
```

```
Private Sub Command5_Click()
'Permite iniciar una nueva codificación
   l = 0
   Text1.Text = ""
   Text2.Text = ""
   Picture1.Cls
End Sub

Private Sub mnuApliCodificar_Click()
'Visualizo el formulario de la aplicación para codificar
   Unload Me
   CodAMI.Show
End Sub

Private Sub mnuApliEsp_Click()
'llana al formulario para visualizar los espectros de potencia
   Unload Me
   Espectros.Show
End Sub

Private Sub mnuArchivoSalir_Click()
'Termina el programa cerrando el puerto si es necesario
   If MSComm1.PortOpen = True Then
        Unload Me
        End
   Else
        End
   End If
End Sub

Private Sub mnuVerInfo_Click()
'Visualizo el formulario que presenta la información del código de línea
   Unload Me
   InfAMI.Show
End Sub

Private Sub Text1_KeyPress(KeyAscii As Integer)
'Chequeo que los valores ingresados sean solo los permitidos
   If (KeyAscii = &H30 Or KeyAscii = &H2B Or KeyAscii = &H2D Or KeyAscii = &H8) Then
      If (KeyAscii = &H30 Or KeyAscii = &H2B Or KeyAscii = &H2D) Then
         l = l + 1
      Else
         l = l - 1
      End If

      If l < 40 Then
      Else
      'Chequea que el numero de caracteres ingresados sean los permitidos
         l = l - 1
         p = MsgBox("El número de caracteres permitidos, de la cadena ingresada es 39", vbInformation,
"Información")
         KeyAscii = 0
      End If
   Else
      p = MsgBox("Solo se ingresan los niveles de los datos (+, 0, -)", vbCritical, "Error")
      KeyAscii = 0
   End If
End Sub
```

```
Private Sub Command1_Click()
'Regresar al formulario para seleccionar el código de línea
   Unload Me
   Acerca.Show
End Sub

Private Sub Combo1_Click()
'Graficara las señales de: reloj, codificada y decodificada (datos) a la escala seleccionada
   Call GRAFICA 'Llama al procedimiento Sub Grafica
End Sub

Private Sub Combo2_Click()
'Graficara las señales de: reloj, codificada y decodificada (datos) a la escala seleccionada
   Call GRAFICA 'Llama al procedimiento Sub Grafica
End Sub

Private Sub Form_Load()
'Permite habilitar la opcion de configurar el dispositivo una vez que el codec este conectado
   If CODEC = 1 Then
      mnuDispositivoDetectado.Enabled = True
      'Configura y abre el puerto para la comunicación con el dispositivo
      MSComm1.CommPort = COM
      MSComm1.PortOpen = True ' ABRO EL PUERTO
   Else
      mnuDispositivoDetectado.Enabled = False
      Timer1.Enabled = False              'Detiene el timer
   End If
'Inicializamos el contador de caracteres ingresados
   l = 0
End Sub

Private Sub Form_Unload(Cancel As Integer)
'Cierra el puerto al momento de descargar el formulario
   If CODEC = 1 Then
      MSComm1.PortOpen = False
   Else
   End If
End Sub

Private Sub MSComm1_OnComm()
'SE EJECUTA SIEMPRE HAY UN DATO EN PUERTO SERIAL
'Permite la recepción de un dato para detectar si el codec esta o no conectado
   Timer1.Enabled = False 'Detiene el timer
   letra = MSComm1.Input
   If letra = "C" Then
      CODEC = 1

   Else
   End If
   Timer1.Enabled = True   'Inicializa el timer
End Sub

Private Sub mnuDispositivoDetectado_Click()
'Llama al formulario para configurar el dispositivo
   Unload Me
   Main.Show
End Sub

Private Sub Timer1_Timer()
```

```
'Timer que permite detectar si el codec portátil fue desconectado
   CODEC = 0
   p = MsgBox("El CODEC ha sido Desconectado", vbInformation, "Información")
   MSComm1.PortOpen = False ' CIERRO EL PUERTO
   Timer1.Enabled = False
   mnuDispositivoDetectado.Enabled = False
End Sub

Private Sub SysInfo1_DeviceArrival(ByVal tipoDispositivo As Long, ByVal IdDispositivo As Long, ByVal
nombreDispositivo As String, ByVal datosDispositivo As Long)
'Se produce cuando se agrega un nuevo dispositivo al sistema
   p = MsgBox("Nuevo Hardware encontrado", vbInformation, "Información")
   p = MsgBox("El CODEC esta listo para ser usado", vbInformation, "Información")
   mnuDispositivoDetectado.Enabled = True
 'Configura y abre el puerto para la comunicación con el dispositivo
   MSComm1.CommPort = COM
   MSComm1.PortOpen = True ' ABRO EL PUERTO
   MSComm1.Output = "Z"
   CODEC = 1
End Sub

Private Sub Toolbar1_ButtonClick(ByVal Button As MSComctlLib.Button)
'Permite seleccionar la accion a realizar
   Select Case Button.Key
      Case "Configurar"
         'Llama al formulario para configurar el dispositivo
         If CODEC = 1 Then
            Unload Me
            Main.Show
         Else
            p = MsgBox("No se puede configurar el CODEC porque aun no esta conectado", vbInformation,
"Información")
         End If
      Case "Información"
         ' Permite Visualizar el formulario de Información del código seleccionado
         Unload Me
         InfAMI.Show
      Case "Codificar"
         ' Permite visualizar el formulario de la aplicación para la codificacion con el código seleccionado
         Unload Me
         CodAMI.Show
      Case "D.E.P"
         'llama al formulario para visualizar los espectros de potencia
         Unload Me
         Espectros.Show
      Case "Salir"
         'Termina el programa cerrando el puerto si es necesario
         If MSComm1.PortOpen = True Then
            Unload Me
            End
         Else
            End
         End If

   End Select
End Sub
```

FORMULARIO DECODIFICAR CON EL CÓDIGO DE LÍNEA B3ZS

FORMULARIO DECODIFICAR CON EL CÓDIGO DE LÍNEA BIFASE-M
FORMULARIO DECODIFICAR CON EL CÓDIGO DE LÍNEA CMI
FORMULARIO DECODIFICAR CON EL CÓDIGO DE LÍNEA HDB3
FORMULARIO DECODIFICAR CON EL CÓDIGO DE LÍNEA MANCHESTER DIFERENCIAL
FORMULARIO DECODIFICAR CON EL CÓDIGO DE LÍNEA MILLER
FORMULARIO DECODIFICAR CON EL CÓDIGO DE LÍNEA MLT-3
FORMULARIO DECODIFICAR CON EL CÓDIGO DE LÍNEA MS43
FORMULARIO DECODIFICAR CON EL CÓDIGO DE LÍNEA NRZ POLAR
FORMULARIO DECODIFICAR CON EL CÓDIGO DE LÍNEA RZ POLAR
FORMULARIO INFORMACIÓN DEL CÓDIGO DE LÍNEA 4B3T
FORMULARIO DIAGRAMA DE ESTADOS 4B3T
FORMULARIO REGLA DE CODIFICACIÓN 4B3T
FORMULARIO INFORMACIÓN DEL CÓDIGO DE LÍNEA AMI
FORMULARIO INFORMACIÓN DEL CÓDIGO DE LÍNEA B3ZS
FORMULARIO REGLA DE CODIFICACIÓN B3ZS
FORMULARIO INFORMACIÓN DEL CÓDIGO DE LÍNEA BIFASE-M
FORMULARIO INFORMACIÓN DEL CÓDIGO DE LÍNEA CMI
FORMULARIO INFORMACIÓN DEL CÓDIGO DE LÍNEA HDB3
FORMULARIO REGLA DE CODIFICACIÓN HDB3
FORMULARIO INFORMACIÓN DEL CÓDIGO DE LÍNEA MANCHESTER DIFERENCIAL
FORMULARIO INFORMACIÓN DEL CÓDIGO DE LÍNEA MILLER
FORMULARIO INFORMACIÓN DEL CÓDIGO DE LÍNEA MLT-3
FORMULARIO INFORMACIÓN DEL CÓDIGO DE LÍNEA MS43
FORMULARIO DIAGRAMA DE ESTADOS MS43
FORMULARIO REGLA DE CODIFICACIÓN MS43
FORMULARIO INFORMACIÓN DEL CÓDIGO DE LÍNEA NRZ POLAR
FORMULARIO INFORMACIÓN DEL CÓDIGO DE LÍNEA RZ POLAR
FORMULARIO PRESENTACIÓN DEL PROGRAMA
FORMULARIO CONFIGURACIÓN DEL CODEC

```
' Declaración de variables
Dim cod As String
Dim vel As String
Dim nsel As String

Dim p As Variant
Dim i As Variant
Dim letra As Variant
```

```
Option Explicit

Private Sub Combo1_Click()
'Permite seleccionar las velocidades dependiendo del código seleccionado
   If Combo1.Text = "NRZ Polar" Then
      Combo2.Clear
      Combo2.Text = "2400"
      Combo2.AddItem "300"
      Combo2.AddItem "1200"
      Combo2.AddItem "2400"
      Combo2.AddItem "4800"
      Combo2.AddItem "9600"
      Combo2.AddItem "19200"
   ElseIf Combo1.Text = "AMI" Then
      Combo2.Clear
      Combo2.Text = "2400"
      Combo2.AddItem "300"
      Combo2.AddItem "1200"
      Combo2.AddItem "2400"
      Combo2.AddItem "4800"
      Combo2.AddItem "9600"
      Combo2.AddItem "19200"
   ElseIf Combo1.Text = "RZ Polar" Then
      Combo2.Clear
      Combo2.Text = "2400"
      Combo2.AddItem "300"
      Combo2.AddItem "1200"
      Combo2.AddItem "2400"
      Combo2.AddItem "4800"
   ElseIf Combo1.Text = "Manchester Diferencial" Then
      Combo2.Clear
      Combo2.Text = "2400"
      Combo2.AddItem "300"
      Combo2.AddItem "1200"
      Combo2.AddItem "2400"
      Combo2.AddItem "4800"
   ElseIf Combo1.Text = "Bifase M" Then
      Combo2.Clear
      Combo2.Text = "2400"
      Combo2.AddItem "300"
      Combo2.AddItem "1200"
      Combo2.AddItem "2400"
      Combo2.AddItem "4800"
   ElseIf Combo1.Text = "Miller" Then
      Combo2.Clear
      Combo2.Text = "2400"
      Combo2.AddItem "300"
      Combo2.AddItem "1200"
      Combo2.AddItem "2400"
      Combo2.AddItem "4800"
   ElseIf Combo1.Text = "4B3T" Then
      Combo2.Clear
      Combo2.Text = "300"
      Combo2.AddItem "300"
   ElseIf Combo1.Text = "MS43" Then
      Combo2.Clear
      Combo2.Text = "300"
      Combo2.AddItem "300"
   ElseIf Combo1.Text = "B3ZS" Then
```

```
    Combo2.Clear
    Combo2.Text = "2400"
    Combo2.AddItem "300"
    Combo2.AddItem "1200"
    Combo2.AddItem "2400"
    Combo2.AddItem "4800"
  ElseIf Combo1.Text = "HDB3" Then
    Combo2.Clear
    Combo2.Text = "2400"
    Combo2.AddItem "300"
    Combo2.AddItem "1200"
    Combo2.AddItem "2400"
    Combo2.AddItem "4800"
  ElseIf Combo1.Text = "MLT3" Then
    Combo2.Clear
    Combo2.Text = "2400"
    Combo2.AddItem "300"
    Combo2.AddItem "1200"
    Combo2.AddItem "2400"
    Combo2.AddItem "4800"
  ElseIf Combo1.Text = "CMI" Then
    Combo2.Clear
    Combo2.Text = "2400"
    Combo2.AddItem "300"
    Combo2.AddItem "1200"
    Combo2.AddItem "2400"
    Combo2.AddItem "4800"
  End If

End Sub

Private Sub Command1_Click()
'Permite llamar al formulario para graficar la señal de datos y la señal codificada mediante la recepción de datos enviados por el dispositivo
  Unload Me
  MSComm1.Output = "X"
  Timer1.Enabled = False
  MSComm1.PortOpen = False
  Graficar.Show
End Sub

Private Sub Command2_Click()

'Permite enviar información al dispositivo deacuerdo al código de línea y velocidad de transmición seleccionada
If Command2.Caption = "&Enviar Configuración " Then

  ' Asigno una letra a cada codigo de linea para facilitar el envío al dispositivo el código de línea que fue seleccionado
  If Combo1.Text = "" Then 'Si no se ha seleccionado un código de línea, se informará de tal suceso
    p = MsgBox("Por favor escoja un CODIGO DE LINEA", vbCritical, "Error")
  ElseIf Combo1.Text = "NRZ Polar" Then
    cod = "A"
    CODLINE = "R"
  ElseIf Combo1.Text = "AMI" Then
    cod = "B"
    CODLINE = "A"
  ElseIf Combo1.Text = "RZ Polar" Then
    cod = "C"
```

```
    CODLINE = "P"
  ElseIf Combo1.Text = "Manchester Diferencial" Then
    cod = "D"
    CODLINE = "F"
  ElseIf Combo1.Text = "Bifase M" Then
    cod = "E"
    CODLINE = "B"
  ElseIf Combo1.Text = "Miller" Then
    cod = "F"
    CODLINE = "M"
  ElseIf Combo1.Text = "4B3T" Then
    cod = "G"
    CODLINE = "T"
  ElseIf Combo1.Text = "MS43" Then
    cod = "H"
    CODLINE = "S"
  ElseIf Combo1.Text = "B3ZS" Then
    cod = "I"
    CODLINE = "Z"
  ElseIf Combo1.Text = "HDB3" Then
    cod = "J"
    CODLINE = "D"
  ElseIf Combo1.Text = "MLT3" Then
    cod = "K"
    CODLINE = "E"
  ElseIf Combo1.Text = "CMI" Then
    cod = "L"
    CODLINE = "C"
  End If

  ' Asigno una letra a cada velocidad  para facilitar el envío al dispositivo la velocidad de transmición que
fue seleccionada
  If Combo2.Text = "" Then   'Si no se ha seleccionado una velocidad de transmición, se informará de tal
suceso
    p = MsgBox("Por favor escoja una VELOCIDAD DE TRANSMICION", vbCritical, "Error")
  ElseIf Combo2.Text = 300 Then
    vel = "M"
    VELTRAN = 300
  ElseIf Combo2.Text = 600 Then
    vel = "N"
    VELTRAN = 600
  ElseIf Combo2.Text = 1200 Then
    vel = "O"
    VELTRAN = 1200
  ElseIf Combo2.Text = 2400 Then
    vel = "P"
    VELTRAN = 2400
  ElseIf Combo2.Text = 4800 Then
    vel = "Q"
    VELTRAN = 4800
  ElseIf Combo2.Text = 9600 Then
    vel = "R"
    VELTRAN = 9600
  ElseIf Combo2.Text = 19200 Then
    vel = "S"
    VELTRAN = 19200
  End If

  ' Asigno una letra a cada nivel de señal
```

```
   If Combo3.Text = "" Then   'Si no se ha seleccionado una velocidad de transmición, se informará de tal
suceso
      p = MsgBox("Por favor escoja un NIVEL DE SEÑAL", vbCritical, "Error")
   ElseIf Combo3.Text = "TTL" Then
      nsel = "V"
   ElseIf Combo3.Text = "RS-232" Then
      nsel = "W"
   End If

   'Envío los caracteres que indicarán al dispositivo (uC) el código de línea y la velocidad de transmición
seleccionados
   If MSComm1.PortOpen = True Then
   Else
      MSComm1.PortOpen = True
   End If
   MSComm1.Output = nsel   'Envio el nivel de señal
   MSComm1.Output = vel   'Envio la velocidad de transmición
   MSComm1.Output = cod   'Envio el código de línea

   'Permite visualizar una etiqueta de "COMPLETADO", una vez que se haya enviado todos los parametros
de configuración del dispositivo
   If Combo1.Text = "" Or Combo2.Text = "" Then
   Else
      Label10.Caption = "COMPLETADO"
      Combo1.Enabled = False
      Combo2.Enabled = False
      Combo3.Enabled = False
      Command2.Caption = "C&ambiar Parámetros"   'Cambia la etiqueta del boton "Enviar Configuración"
por "Cambiar Configuración" para que el mismo boton desempeñe otra función con esta etiqueta
      Command1.Enabled = True
   End If

'Permite cambiar la información a enviarse al dispositivo deacuerdo al código de línea y velocidad de
transmición seleccionada
ElseIf Command2.Caption = "C&ambiar Parámetros" Then
      Command2.Caption = "&Enviar Configuración "  'Cambia la etiqueta del boton "Cambiar
Configuración" por la etiqueta inicial para poder enviar los parametros que se han modificado
      Combo1.Enabled = True
      Combo2.Enabled = True
      Combo3.Enabled = True
      Command1.Enabled = False
End If


End Sub

Private Sub Command4_Click()
' Termina el programa cerrando el puerto si es necesario
   If MSComm1.PortOpen = True Then
         Unload Me
         End
   Else
         End
   End If

End Sub

Private Sub mnuArchivoSalir_Click()
'Termina el programa cerrando el puerto si es necesario
```

```
    If MSComm1.PortOpen = True Then
        Unload Me
        End
    Else
        End
    End If
End Sub

Private Sub Form_Load()
'Permite detener el timer si el codec esta desconectado
    If CODEC = 1 Then
        'Configura y abre el puerto para la comunicación con el dispositivo
        MSComm1.CommPort = COM
        MSComm1.PortOpen = True ' ABRO EL PUERTO
        MSComm1.Output = "Y"
    Else
        Timer1.Enabled = False
    End If

 'Determina las velocidades para el codigo AMI
        Combo2.Clear
        Combo2.Text = "2400"
        Combo2.AddItem "300"
        Combo2.AddItem "1200"
        Combo2.AddItem "2400"
        Combo2.AddItem "4800"
        Combo2.AddItem "9600"
        Combo2.AddItem "19200"
End Sub

Private Sub Form_Unload(Cancel As Integer)
'Cierra el puerto al momento de descargar el formulario
    If CODEC = 1 Then
        MSComm1.PortOpen = False
    Else
    End If
End Sub

Private Sub mnuVerInfor_Click()
'Visualizo el formulario que presenta la información de los códigos de línea
    Unload Me
    Acerca.Show
End Sub
Private Sub MSComm1_OnComm()
'SE EJECUTA SIEMPRE HAY UN DATO EN PUERTO SERIAL
'Permite la recepción de un dato para detectar si el codec esta o no conectado
    Timer1.Enabled = False 'Detiene el timer
    letra = MSComm1.Input
    If letra = "C" Then
        CODEC = 1
    Else
    End If
    Timer1.Enabled = True   'Inicializa el timer
End Sub

Private Sub Timer1_Timer()
'Timer que permite detectar si el codec portátil fue desconectado
    CODEC = 0
```

```
   p = MsgBox("El CODEC ha sido Desconectado, el programa cerrará la aplicación actual y regresá a ver la
Información de los Códigos de Línea", vbInformation, "Información")
   Timer1.Enabled = False
   Unload Me
   Acerca.Show
End Sub

Private Sub Timer2_Timer()
   If Combo1.Text = "MS43" Then
      nsel = "W"
      cod = "H"
      vel = "M"
      MSComm1.Output = nsel    'Envio el nivel de señal
      MSComm1.Output = vel    'Envio la velocidad de transmición
      MSComm1.Output = cod    'Envio el código de línea
   ElseIf Combo1.Text = "4B3T" Then
      nsel = "W"
      cod = "G"
      vel = "M"
      MSComm1.Output = nsel    'Envio el nivel de señal
      MSComm1.Output = vel    'Envio la velocidad de transmición
      MSComm1.Output = cod    'Envio el código de línea
   End If
End Sub

Private Sub Toolbar1_ButtonClick(ByVal Button As MSComctlLib.Button)
'Permite seleccionar la accion a realizar
   Select Case Button.Key
      Case "Información"
         ' Permite Visualizar el formulario de Información del código seleccionado
         Unload Me
         Acerca.Show
      Case "D.E.P"
         'llama al formulario para visualizar los espectros de potencia
         Unload Me
         Espectros.Show
      Case "Salir"
         'Termina el programa cerrando el puerto si es necesario
         If MSComm1.PortOpen = True Then
            Unload Me
            End
         Else
            End
         End If

   End Select
End Sub
```

FORMULARIO SELECCIÓN DEL VCP

```
Dim fs As Variant
Dim A As Variant

Option Explicit
Private Sub Combo1_Click()
'Permite editar el archivo CodecPortatil que permite identificar el VCP asignado por la PC
   Set fs = CreateObject("Scripting.FileSystemObject")
   Set A = fs.CreateTextFile("c:\windows\system32\CodecPortatil.txt", True)
   If Combo1.Text = "COM 3" Then
```

```
      A.WriteLine ("3")
      A.Close
      COM = 3
   ElseIf Combo1.Text = "COM 4" Then
      A.WriteLine ("4")
      A.Close
      COM = 4
   ElseIf Combo1.Text = "COM 5" Then
      A.WriteLine ("5")
      A.Close
      COM = 5
   ElseIf Combo1.Text = "COM 6" Then
      A.WriteLine ("6")
      A.Close
      COM = 6
   ElseIf Combo1.Text = "COM 7" Then
      A.WriteLine ("7")
      A.Close
      COM = 7
   ElseIf Combo1.Text = "COM 8" Then
      A.WriteLine ("8")
      A.Close
      COM = 8
   End If
'Habilitamos Timer para cerrar el formulario
Timer1.Enabled = True
End Sub

Private Sub Timer1_Timer()
'Cierra el formulario actual y llama al siguiente
   Unload Me
   Form5.Show
End Sub
```

FORMULARIO CODEC PORTÁTIL

```
Dim letra As Variant
Dim p As Variant
Dim fs As Variant
Dim A As Variant
Option Explicit


Private Sub Command1_Click()
' Termina el programa cerrando el puerto si es necesario
   If MSComm1.PortOpen = True Then
         Unload Me
         End
   Else
         End
   End If
End Sub

Private Sub mnuArchivoSalir_Click()
' Termina el programa cerrando el puerto si es necesario
   If MSComm1.PortOpen = True Then
         Unload Me
         End
   Else
```

```
        End
    End If
End Sub

Private Sub Option1_Click()
'Llama al formulario para configurar el dispositivo

    Unload Me
    Main.Show
End Sub

Private Sub Option2_Click()
'Llama al formulario que presenta la información de los códigos de línea
    Unload Me
    Acerca.Show
End Sub

Private Sub Form_Load()
'Permite detener el timer si el codec esta desconectado
    If CODEC = 1 Then
        'Configura y abre el puerto para la comunicación con el dispositivo
        MSComm1.CommPort = COM
        MSComm1.PortOpen = True ' ABRO EL PUERTO
    Else
        Timer1.Enabled = False
    End If
End Sub

Private Sub Form_Unload(Cancel As Integer)
'Cierra el puerto al momento de descargar el formulario
    If CODEC = 1 Then
        MSComm1.PortOpen = False
    Else
    End If
End Sub

Private Sub MSComm1_OnComm()
'SE EJECUTA SIEMPRE HAY UN DATO EN PUERTO SERIAL
'Permite la recepción de un dato para detectar si el codec esta o no conectado
    Timer1.Enabled = False 'Detiene el timer
    letra = MSComm1.Input
    If letra = "C" Then
        CODEC = 1

    Else
    End If
    Timer1.Enabled = True   'Inicializa el timer
End Sub

Private Sub Timer1_Timer()
'Timer que permite detectar si el codec portátil fue desconectado
    CODEC = 0
    p = MsgBox("El CODEC ha sido Desconectado", vbInformation, "Información")
    MSComm1.PortOpen = False ' CIERRO EL PUERTO
    Timer1.Enabled = False
    Option1.Enabled = False
    Label5.Enabled = False

End Sub
```